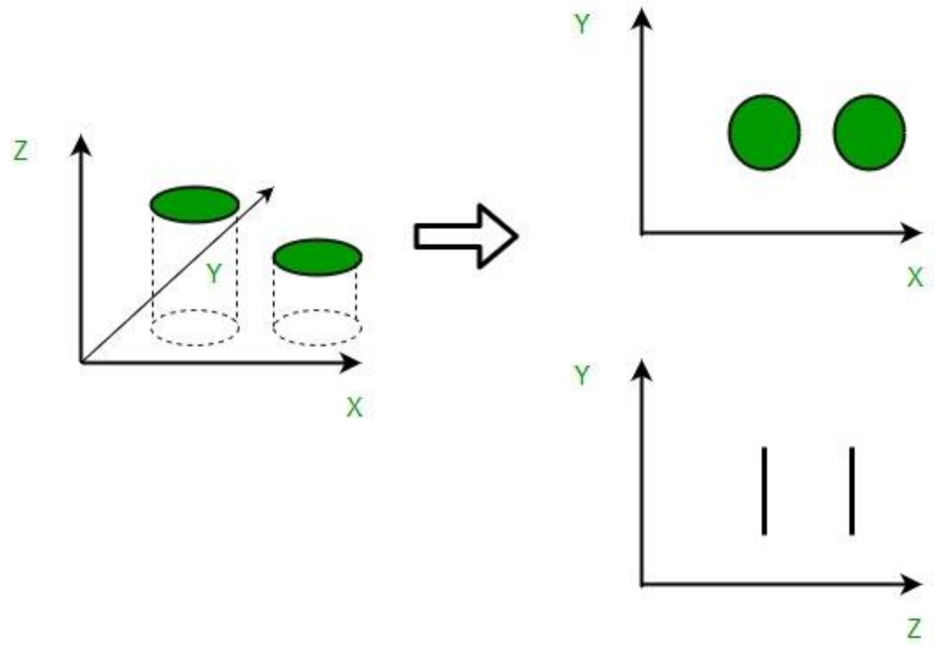
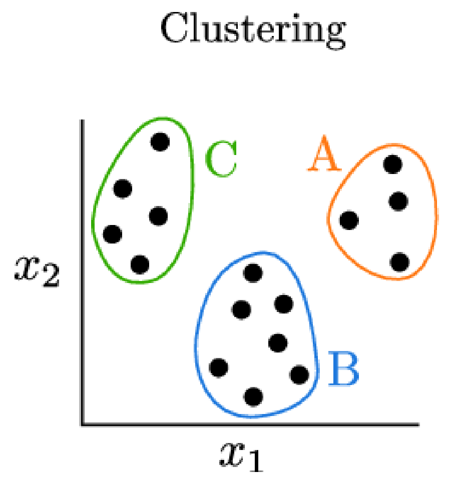
Dimensionality Reduction

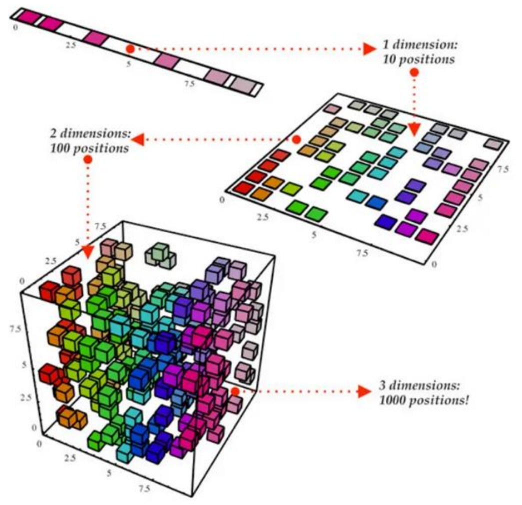


Shuwen Yue
Assistant Professor, Cornell University
May 1, 2025

What do we use it for?

- Visualizing structure in high-dimensional data
- Identifying patterns or clusters
- Removing noise or redundant features
- Reconstruction or generation
- PCs are often used as features in ML models

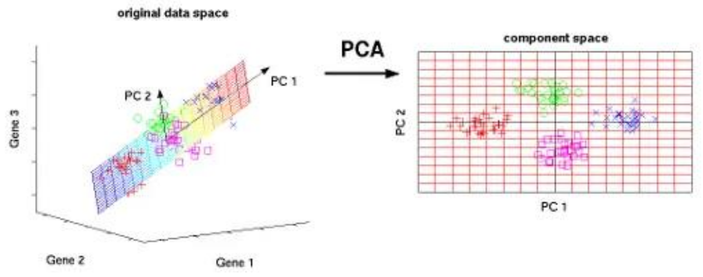



1 dimension: 10 positions
2 dimensions: 100 positions
3 dimensions: 1000 positions!

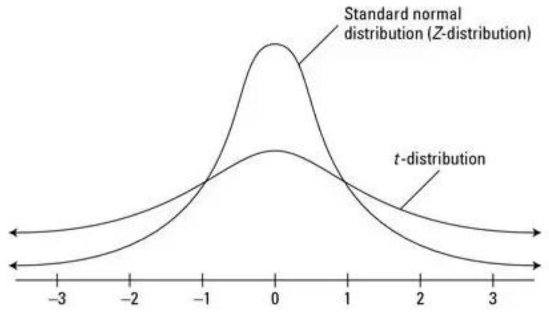
https://corinwagen.github.io/public/blog/20230417_dimensionality_reduction.html

Approaches

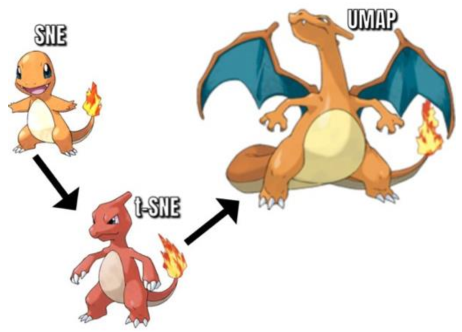
PCA



t-SNE

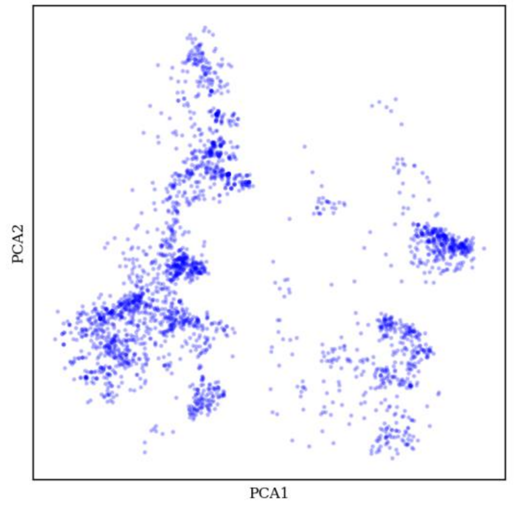


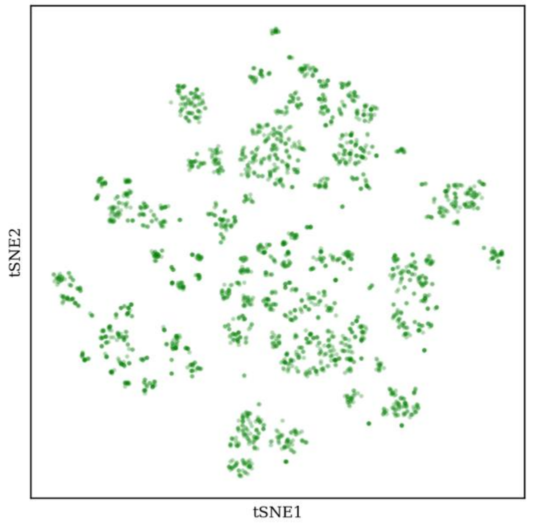
UMAP

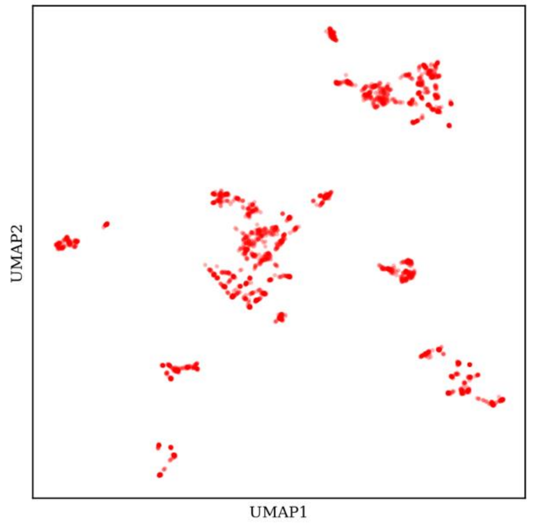


https://arize.com/blog-course/reduction-of-dimensionality-top-techniques/

For an example dataset of 2683 aryl bromides from Reaxys, using RDKIT fingerprints

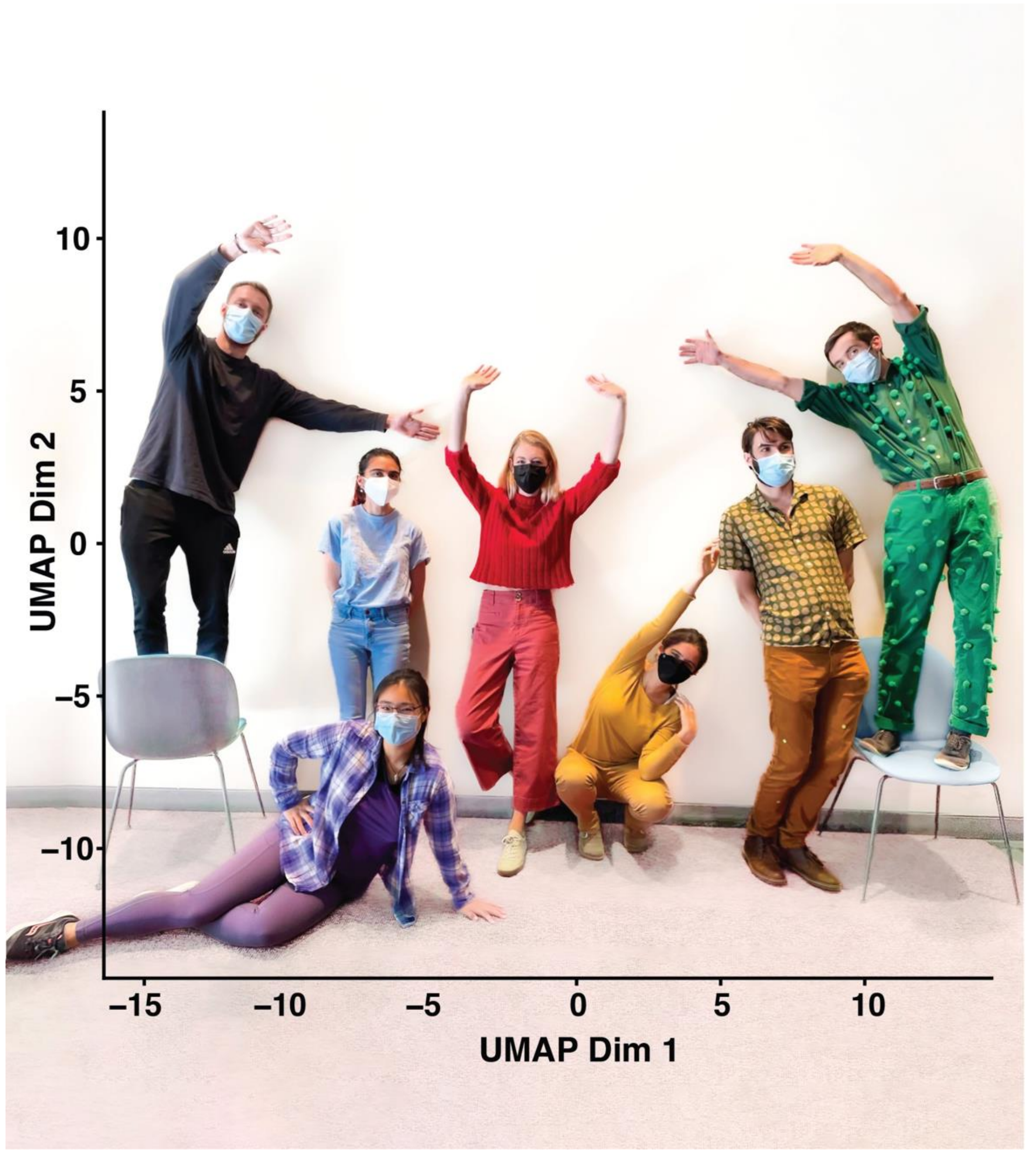


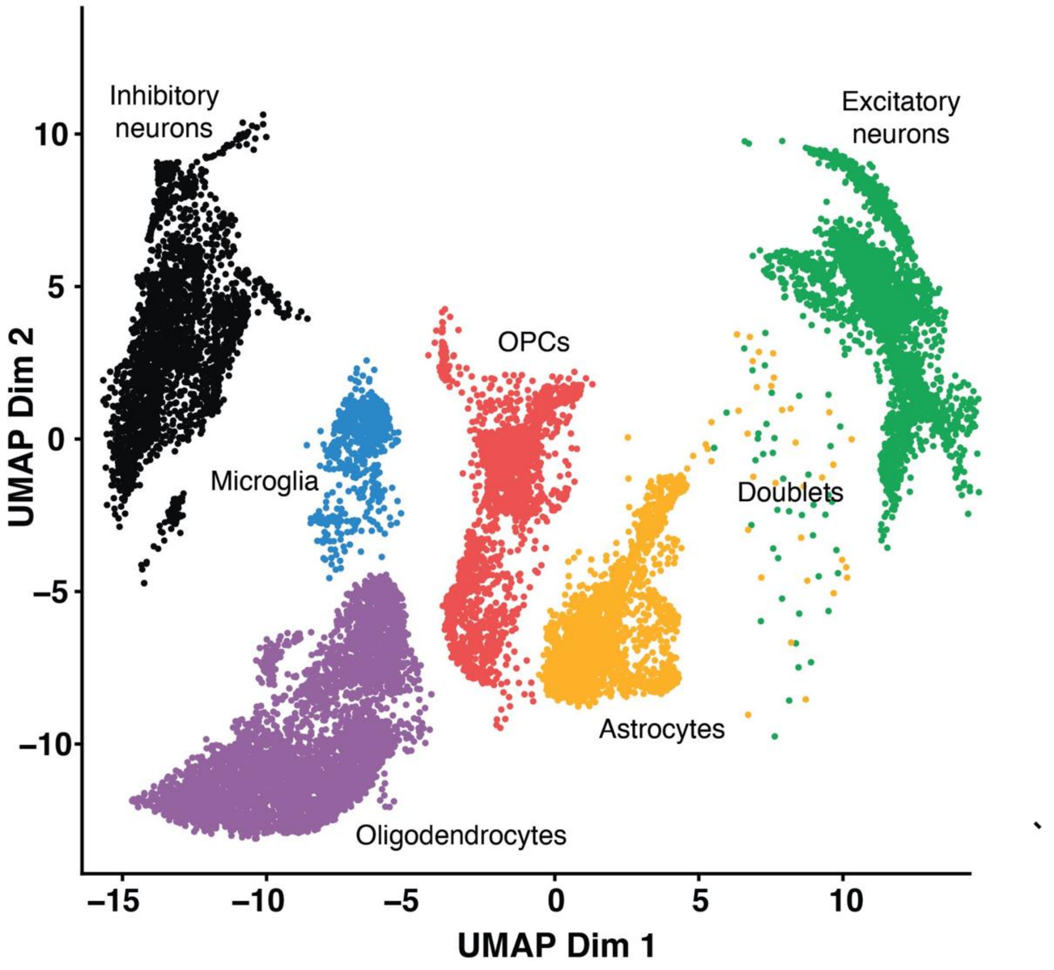




https://corinwagen.github.io/public/blog/20230417_dimensionality_reduction.html

Research group Halloween costume idea…

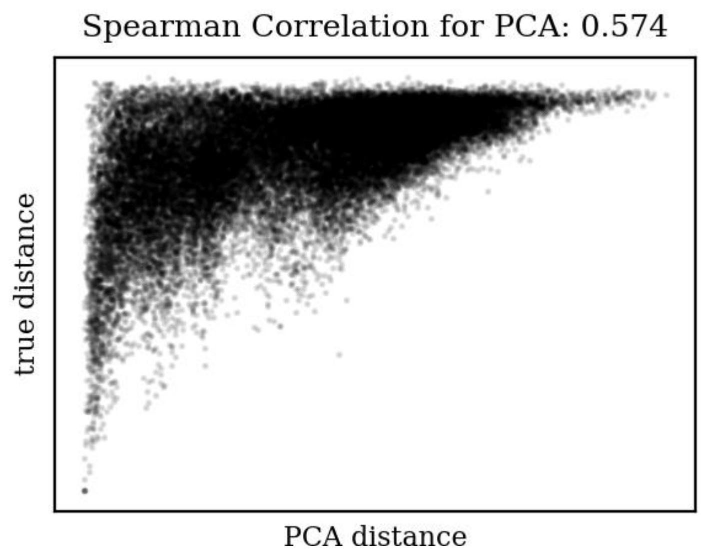




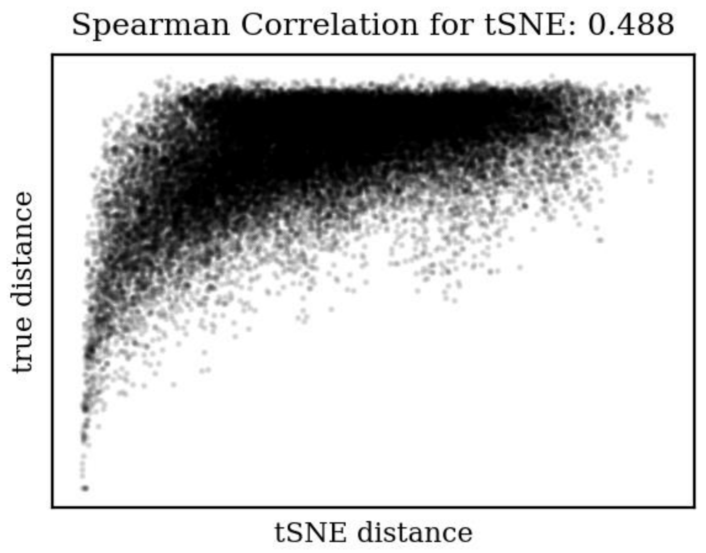
Retaining information – no free lunch!

Distanced preservation – ideally, we want things that are close together in high dimensional space to also be close together in low dimensional space

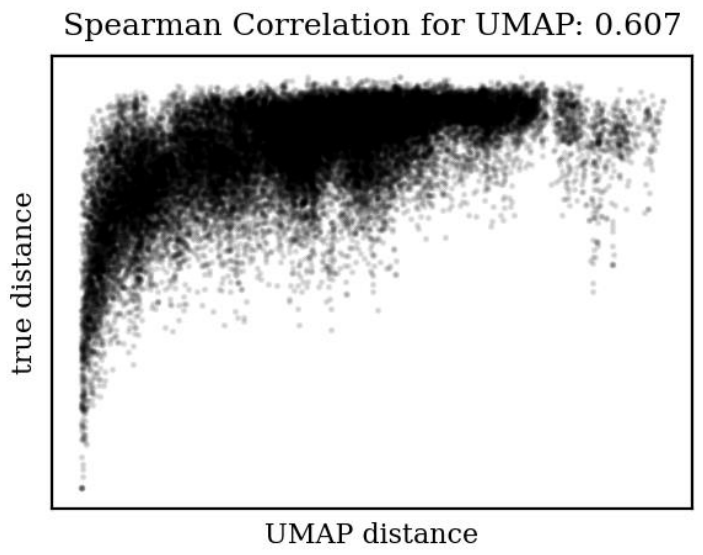
PCA



t-SNE



UMAP

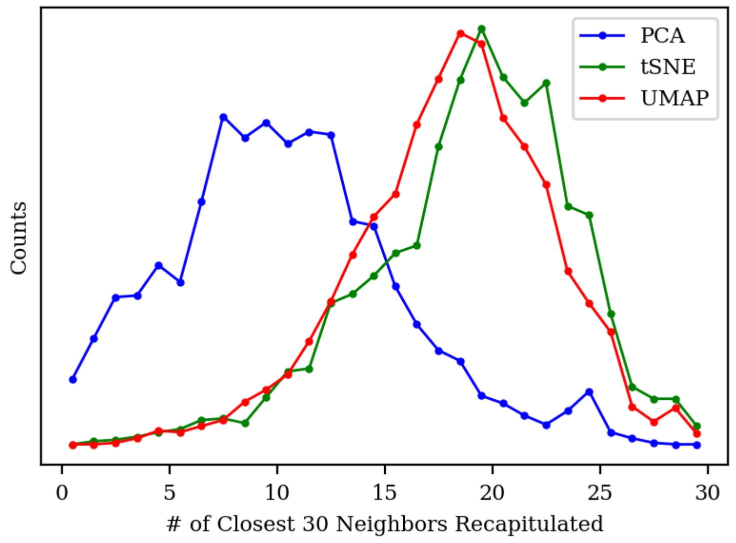


https://corinwagen.github.io/public/blog/20230417_dimensionality_reduction.html

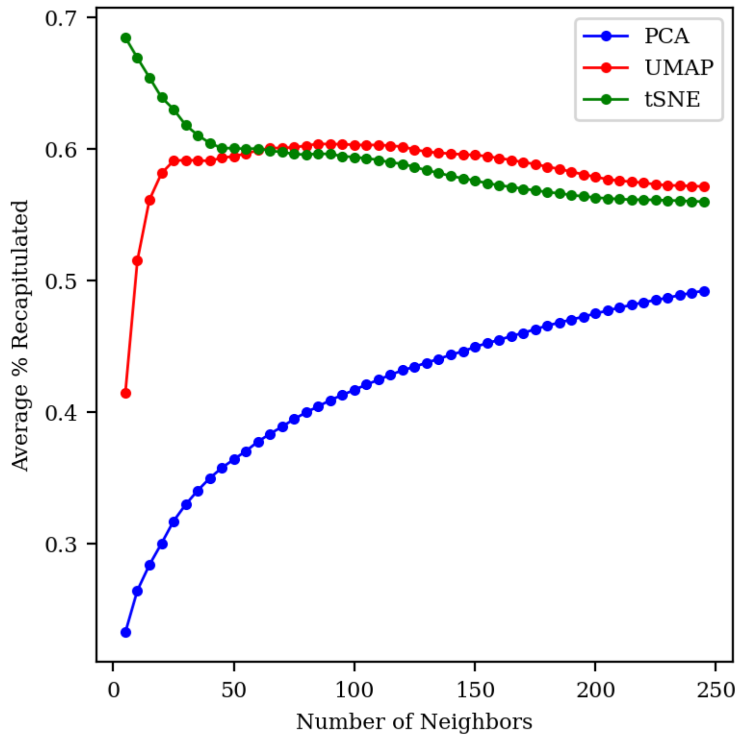
Retaining information – no free lunch!

While dimensionality reduction can help uncover patterns and importance metric, there is still a fundamental trade off and cost in information loss

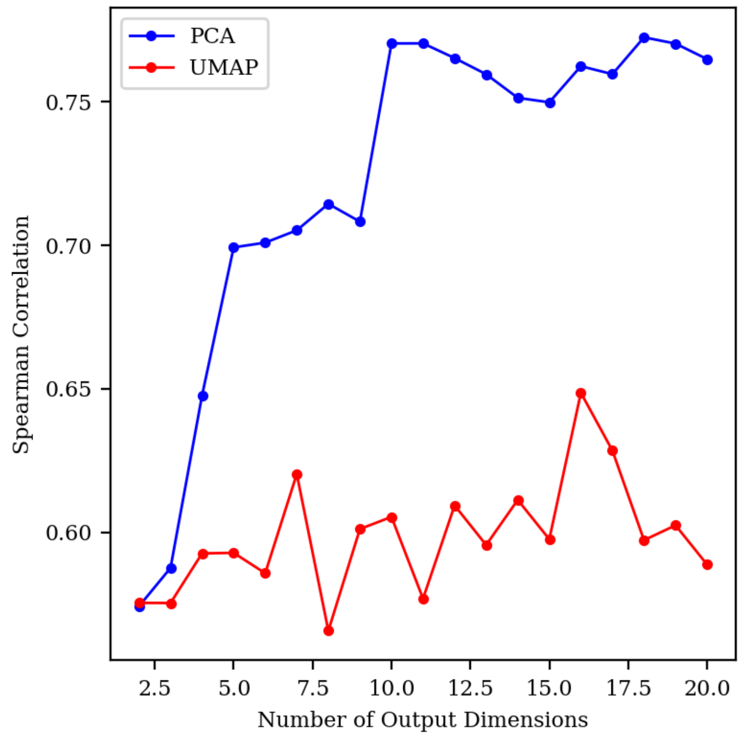
30 nearest neighbors



X nearest neighbors



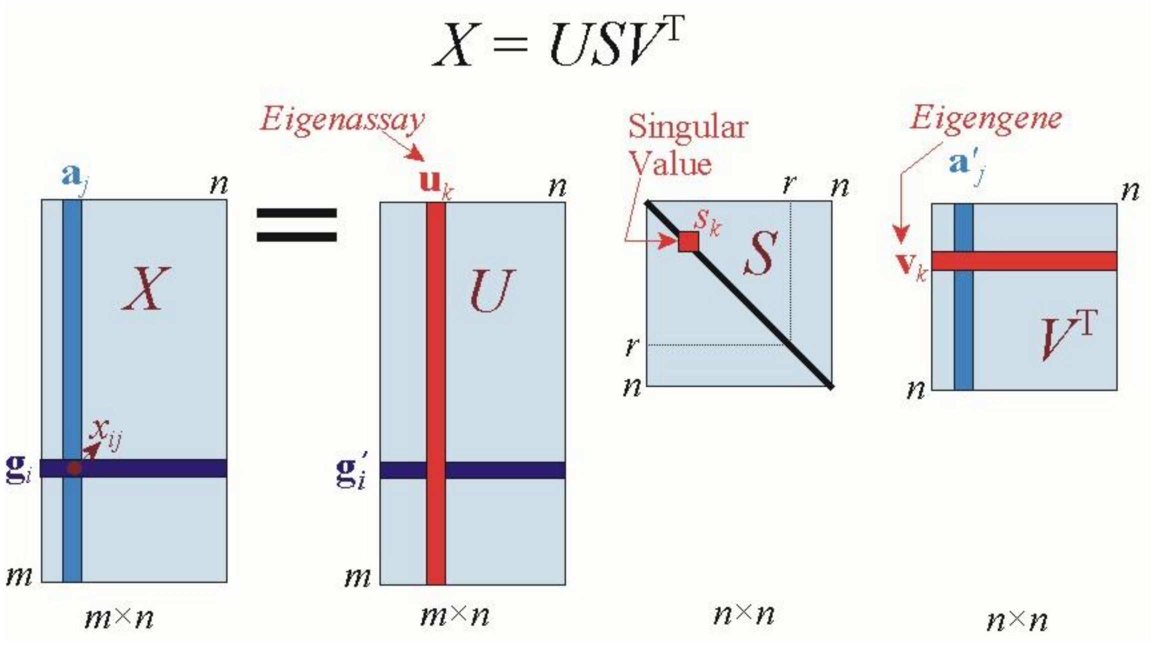
Increasing dimensions



https://corinwagen.github.io/public/blog/20230417_dimensionality_reduction.html

Principle Component Analysis (PCA)

- PCA transforms data linearly into new properties that are not correlated with each other
- Based on Single Value Decomposition (SVD)

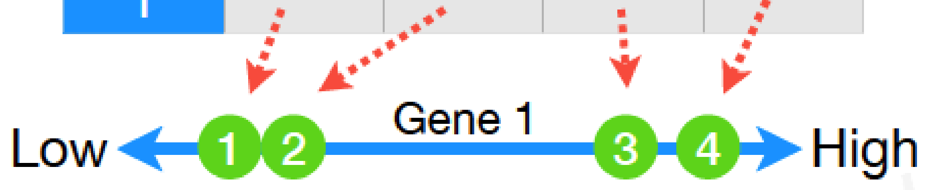


https://jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491

The Problem

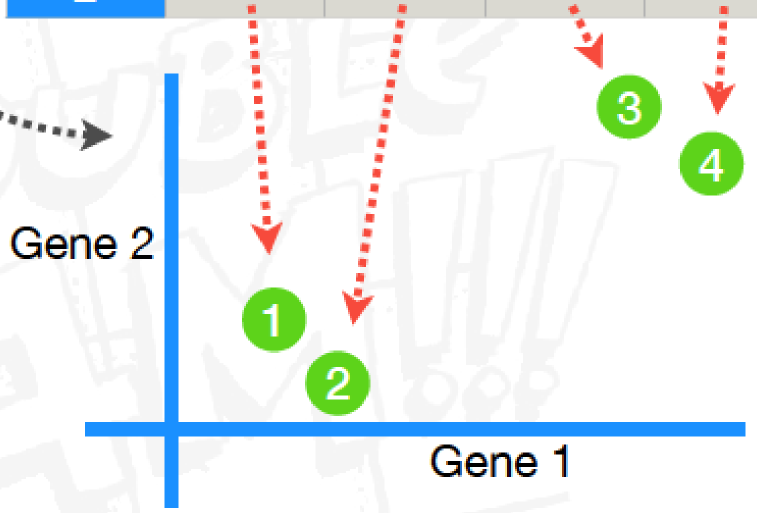
When we have low-dimensional data, like one or two measurements per subject (in this case subject = mouse)…

	Mouse 1	Mouse 2	Mouse 3	Mouse 4
Gene 1	2	3	8	9



…graphing each subject is easy…

	Mouse 1	Mouse 2	Mouse 3	Mouse 4
Gene 1	2	3	8	9
Gene 2	2	1	7	6



	Mouse 1	Mouse 2	…
Gene 1	2	3	…
Gene 2	2	1	…
…	…	…	…
Gene 20000	32	10	…

??

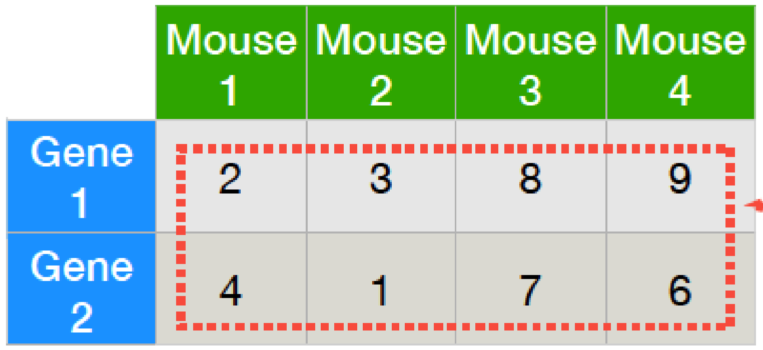
…but when we have high-dimensional data, like measuring 20,000 genes per mouse, graphing each mouse is not so easy.

https://www.youtube.com/watch?v=FgakZw6K1QQ

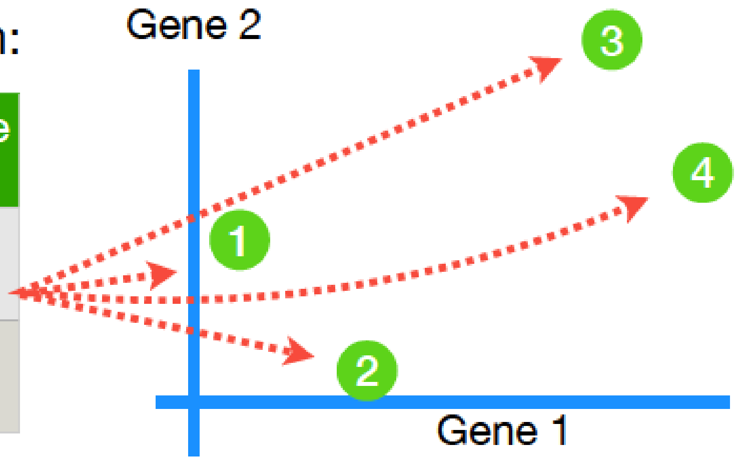
PCA, Step-by-Step

First we will demonstrate the concepts with 2-Dimensional data (2 measurements, Gene 1 and Gene 2, per subject).

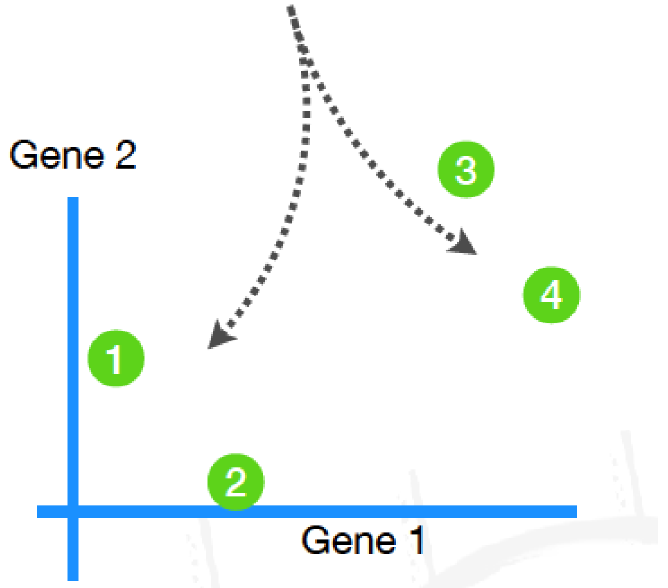
The data and related graph:



	Mouse 1	Mouse 2	Mouse 3	Mouse 4
Gene 1	2	3	8	9
Gene 2	4	1	7	6

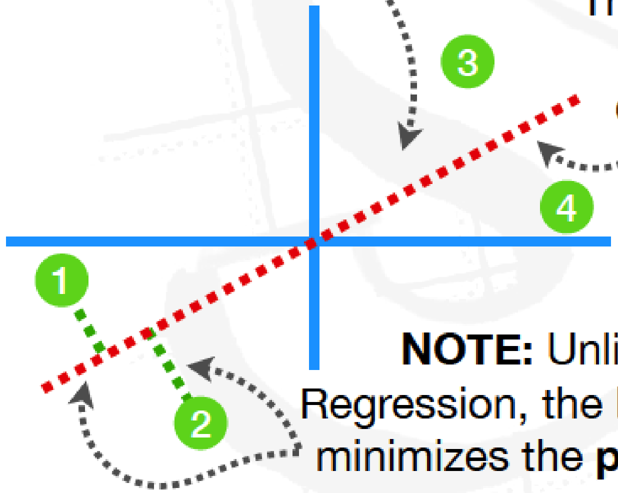


Step 1: Center the Data...



https://www.youtube.com/watch?v=FgakZw6K1QQ

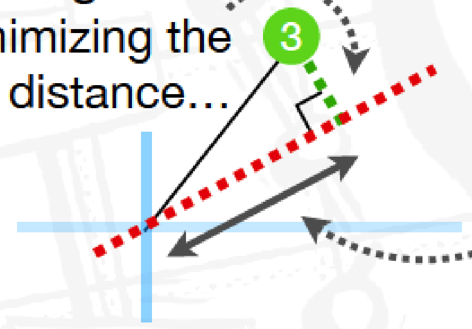
Step 2: Fit a line to the data that goes through the origin…



The best fitting line is the first principal component, **PC1**.

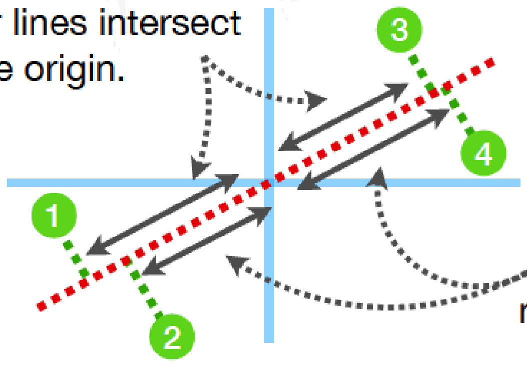
NOTE: Unlike Linear Regression, the best fitting line minimizes the **perpendicular distance** to the data.

Due to the **Pythagorean Theorem**, minimizing the perpendicular distance…



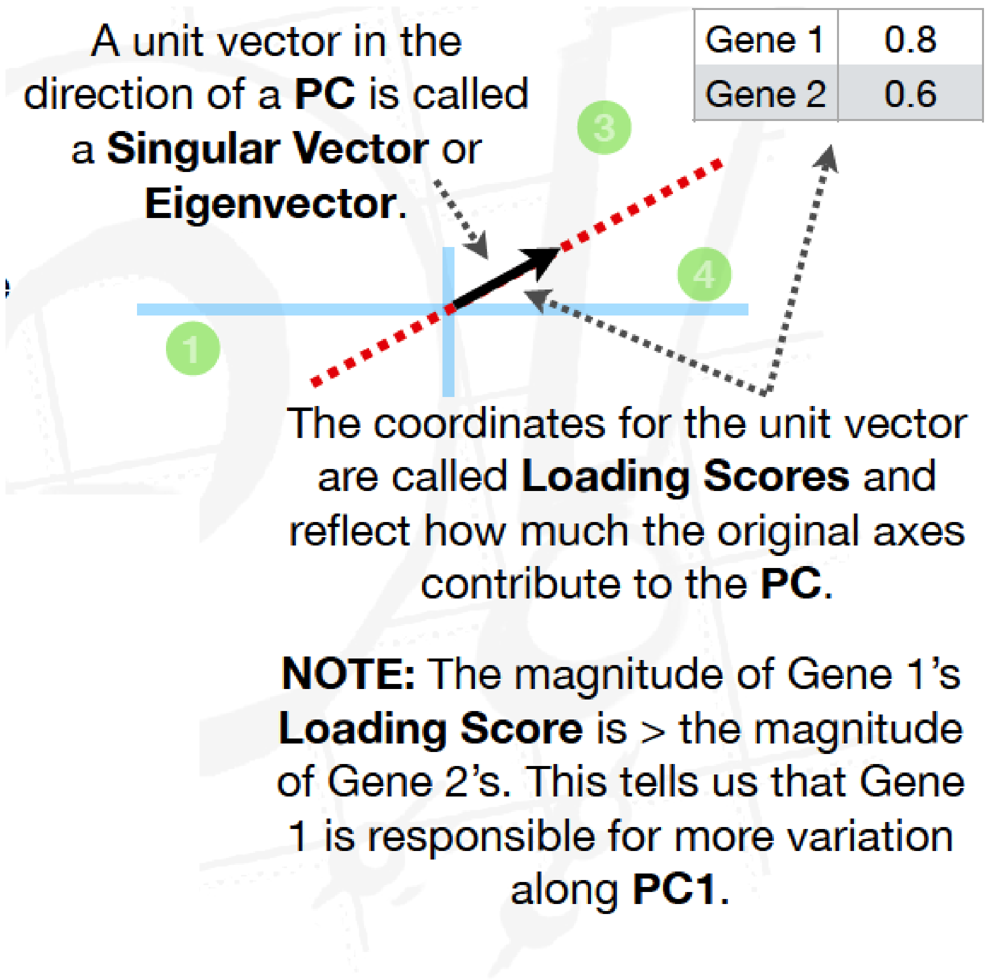
…maximizes the distance along the **PC**.

The **PC** maximizes the variation between where the perpendicular lines intersect and the origin.



The sum of the squared distances along the **PC** is called the **Eigenvalue** for the **PC** and tells us the relative amount of variance along that **PC**.

https://www.youtube.com/watch?v=FgakZw6K1QQ



A unit vector in the direction of a **PC** is called a **Singular Vector** or **Eigenvector**.

Gene 1	0.8
Gene 2	0.6

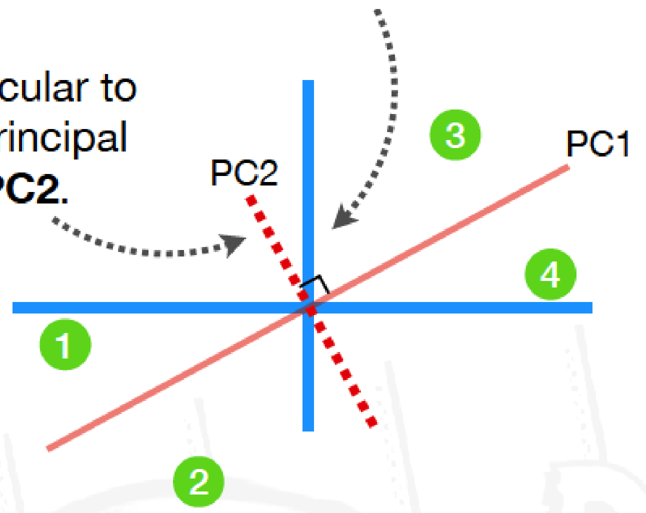
The coordinates for the unit vector are called **Loading Scores** and reflect how much the original axes contribute to the **PC**.

NOTE: The magnitude of Gene 1's **Loading Score** is > the magnitude of Gene 2's. This tells us that Gene 1 is responsible for more variation along **PC1**.

https://www.youtube.com/watch?v=FgakZw6K1QQ

Step 3: Fit a line to the data that is perpendicular to PC1…

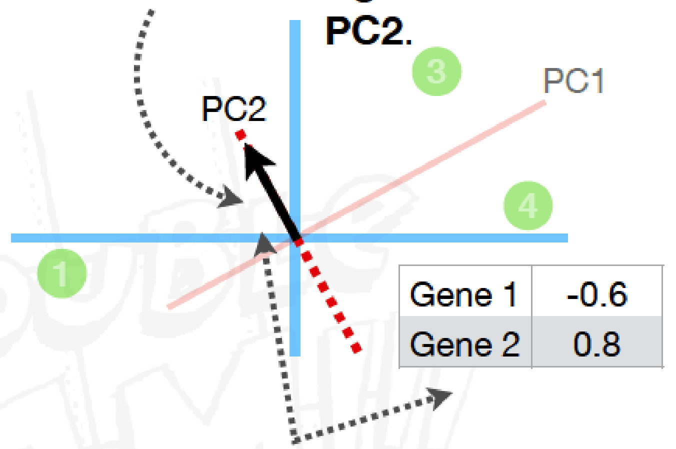
The line perpendicular to **PC1** is second principal component, **PC2**.



NOTE: Because the original data only has 2-Dimensions, there are only 2 principal components.

In general, the number of PCs is determined by whichever is smaller, the number of samples (samples = mice in this example) or the number of variables (variables = genes in this example)

A unit vector in the direction of **PC2** is called a **Singular Vector** or **Eigenvector** for **PC2**.

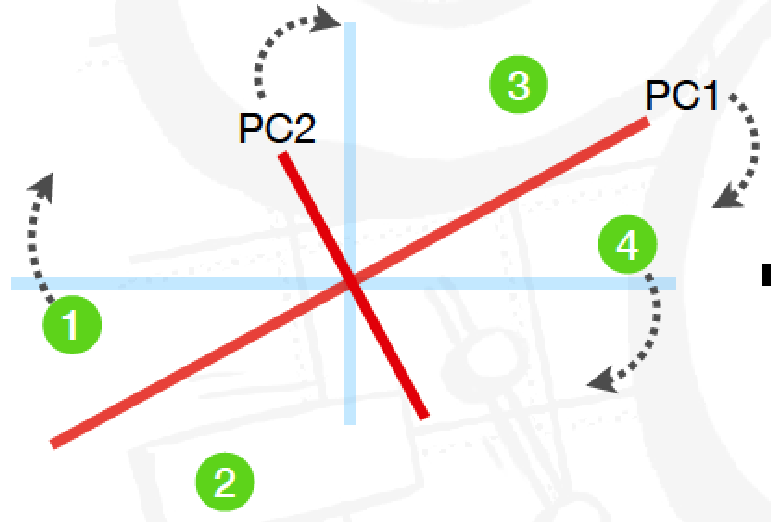


Gene 1	-0.6
Gene 2	0.8

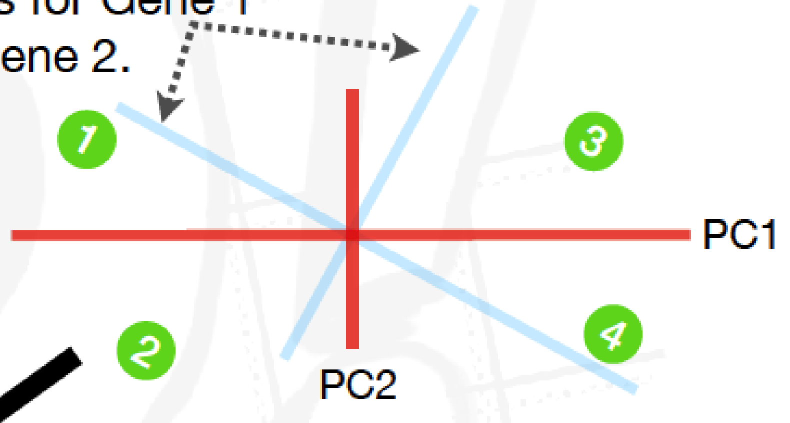
The magnitudes of **Loading Scores** (the coordinates for the unit vector) show that Gene 2 is responsible for more variation along **PC2** than Gene 1.

https://www.youtube.com/watch?v=FgakZw6K1QQ

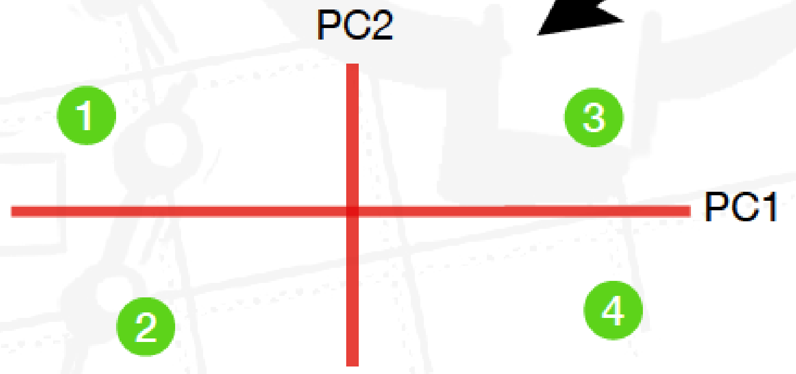
Step 4: Rotate the data and PCs…



The **light blue lines** are the original axes for Gene 1 and Gene 2.



The way the data points are spread out along **PC1** tells us that samples (mice) 1 and 2 are more similar to each than they are to samples 3 and 4.



This is the final **PCA** graph of the data.

Eigenvalue = Sum of Squared Distances along a **PC**

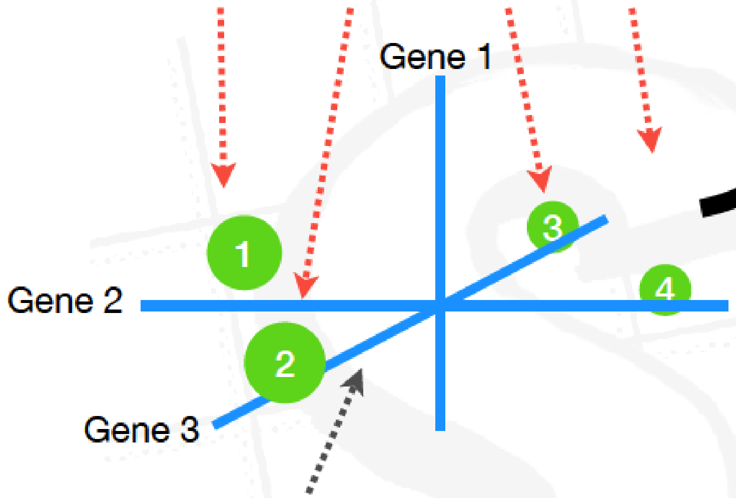
$$\textbf{Variance} = \frac{\text{Sum of Squared Distances along a } \textbf{PC}}{n - 1}$$

…where ***n*** is the number of data points.

https://www.youtube.com/watch?v=FgakZw6K1QQ

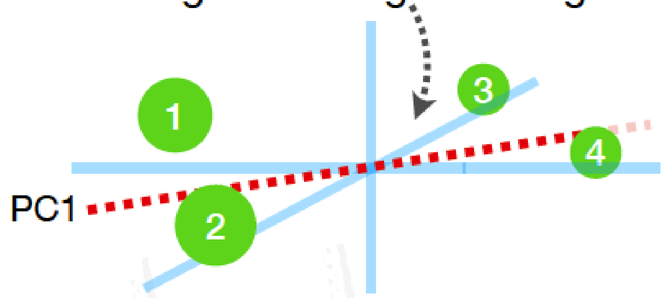
If we have 3-dimensions...

	Mouse 1	Mouse 2	Mouse 3	Mouse 4
Gene 1	2	3	8	9
Gene 2	4	1	7	6
Gene 3	8	10	-3	-4

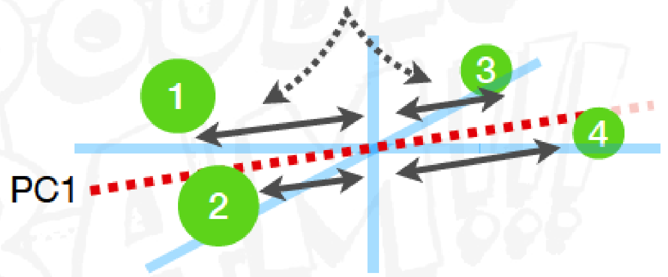


NOTE: The axis for Gene 3 is supposed to represent the 3rd dimension, which is hard to draw on a 2-D piece of paper. Just try to imagine it sticking out of the page.

In 3-D, **PC1** is still the best fitting line that goes through the origin…

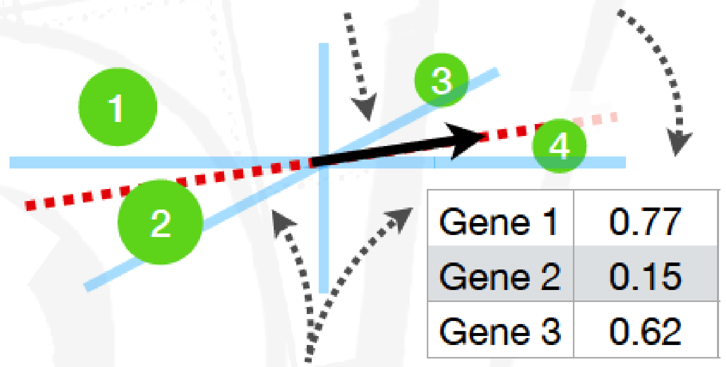


…and it accounts for the largest **Eigenvalue**, the sum of the squared distances.



https://www.youtube.com/watch?v=FgakZw6K1QQ

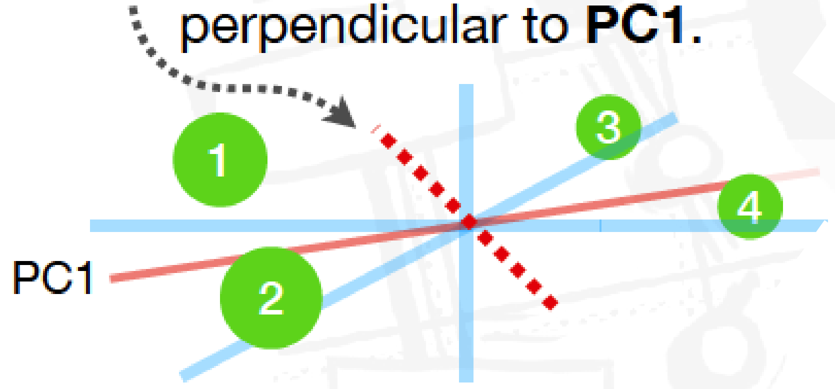
…however, now the **Eigenvector**, the unit vector in the direction of the **PC**, has 3 coordinates.



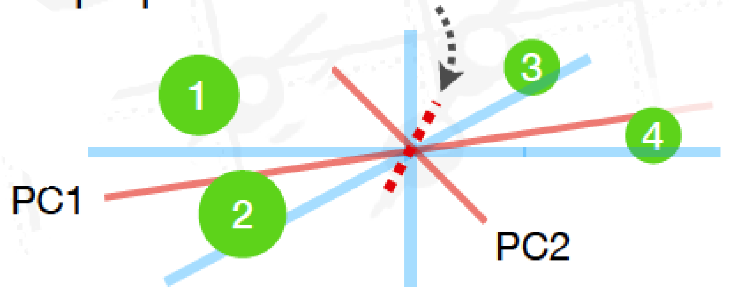
Gene 1	0.77
Gene 2	0.15
Gene 3	0.62

Because Gene 1's coordinate (**Loading Score**) has the largest magnitude, Gene 1 plays the largest role in the direction of **PC1**.

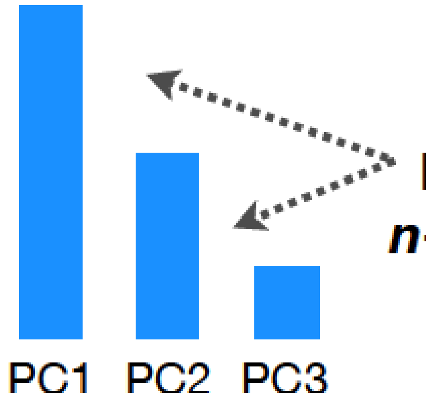
PC2 is the next best fitting line, given that it goes through the origin and is perpendicular to **PC1**.



PC3 is the next best fitting line, given that it goes through the origin and is perpendicular to **PC1** and **PC2**.



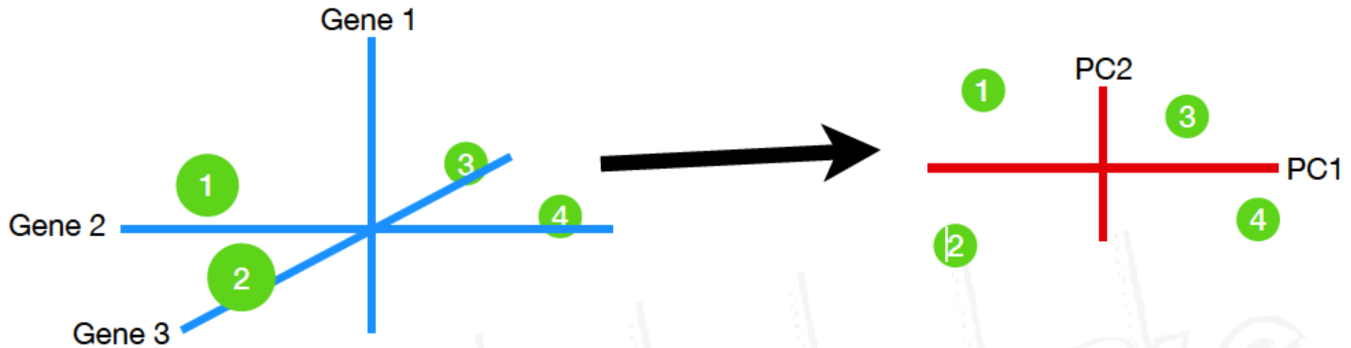
https://www.youtube.com/watch?v=FgakZw6K1QQ



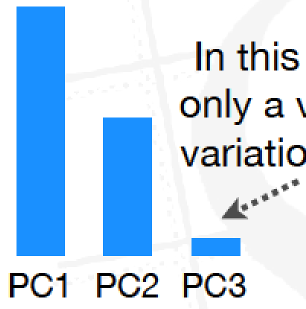
A **Scree Plot** shows the **Eigenvalues**, or if divided by ***n*-1**, the variances, for the **PCs**.

Scree Plots help us evaluate how many **PCs** we need to accurately represent the original data.

To convert the 3-D graph into a 2-D PCA plot…

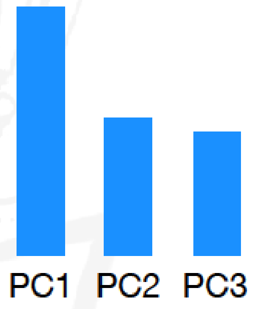


Step 1: Look at the Scree Plot: The **Scree Plot** tells us how much variation each **PC** accounts for.



In this case **PC3** accounts for only a very small amount of the variation and it makes sense to exclude it.

In contrast, if this were the **Scree Plot**, we might hesitate to exclude **PC3** since it accounts for almost as much variation as **PC2**

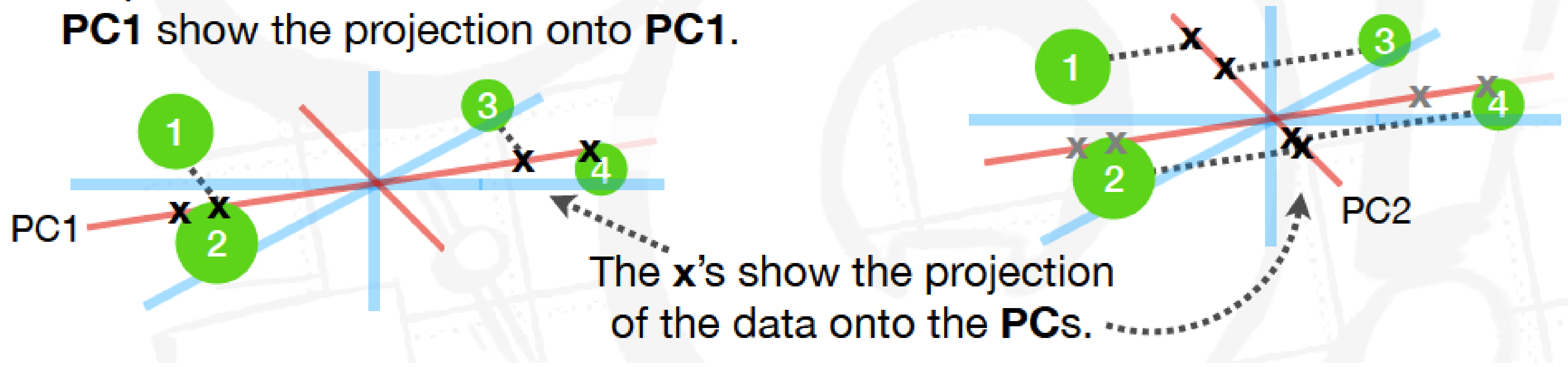


https://www.youtube.com/watch?v=FgakZw6K1QQ

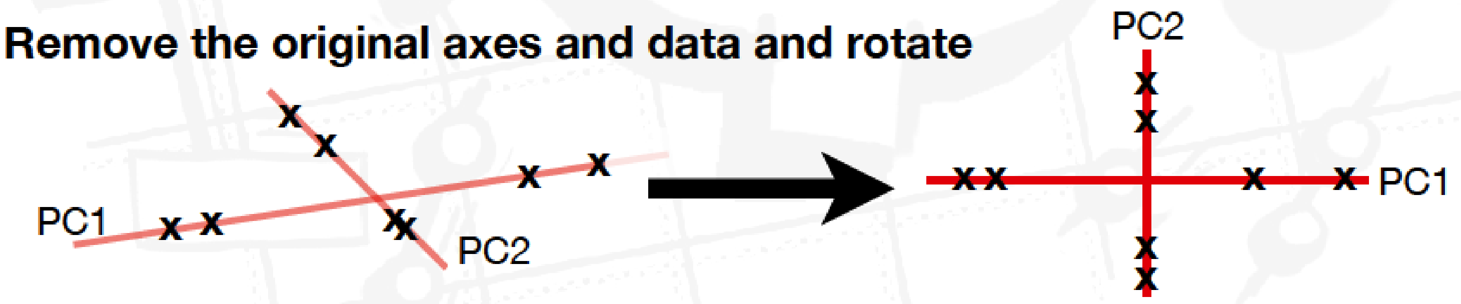
Step 2: Project the data onto PC1 and PC2

Perpendicular lines from the data to **PC1** show the projection onto **PC1**.

Perpendicular lines from the data to **PC2** show the projection onto **PC2**.

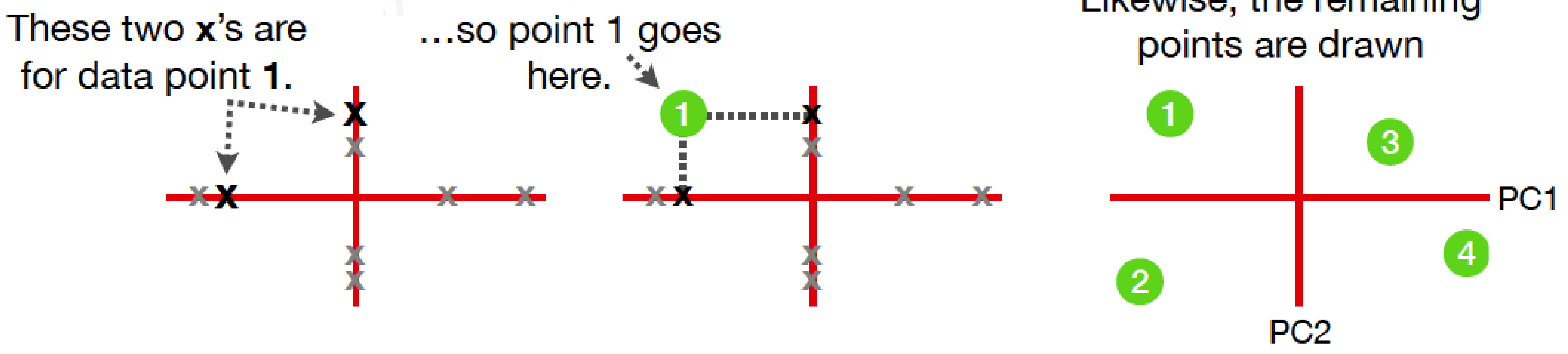

1
2
3
4
PC1
PC2
The **x**'s show the projection of the data onto the **PC**s.

Step 3: Remove the original axes and data and rotate



https://www.youtube.com/watch?v=FgakZw6K1QQ

Step 4: Add the data back


These two x's are for data point 1.
...so point 1 goes here.
1
Likewise, the remaining points are drawn
1
3
PC1
4
2
PC2

https://www.youtube.com/watch?v=FgakZw6K1QQ

Colab notebook