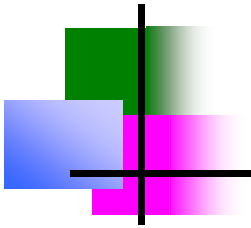


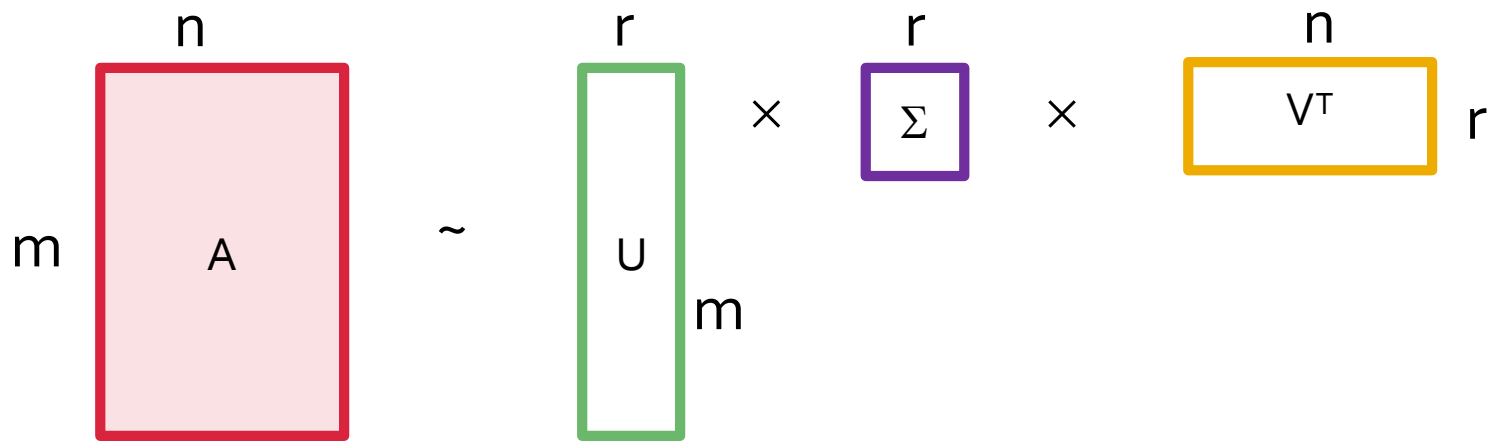
Machine Learning and Computational Intelligence
Lecture 3

Sanjeeb Prasad Panday, PhD
Associate Professor
Dept. of Electronics and Computer Engineering
Director (ICTC)
IOE, TU

SVD: Singular Value Decomposition

Reducing Matrix Dimension

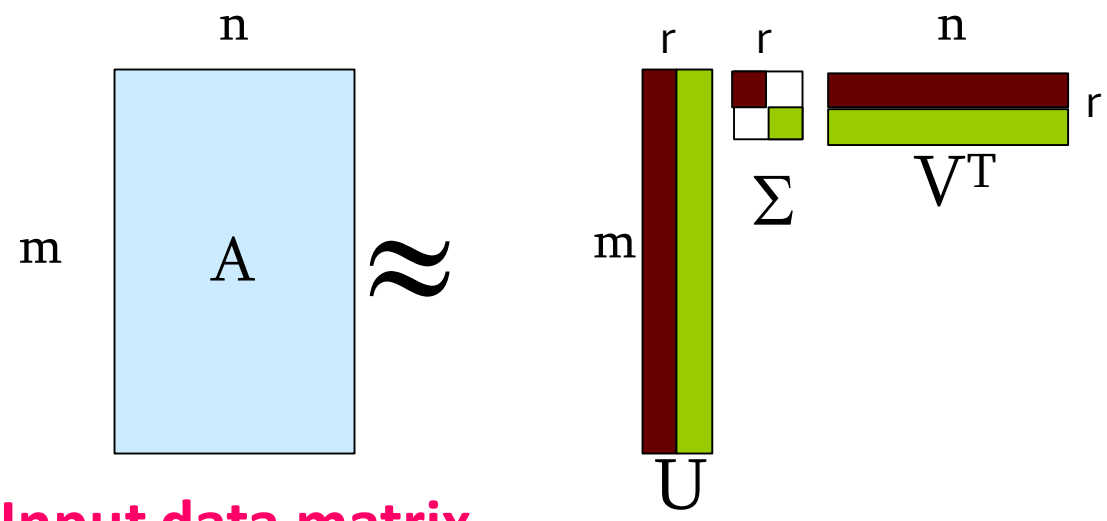
- **Gives a decomposition of any matrix into a product of three matrices:**



- There are strong constraints on the form of each of these matrices
  - Results in a unique decomposition
- From this decomposition, you can choose any number r of intermediate concepts (latent factors) in a way that minimizes the reconstruction error

SVD – Definition

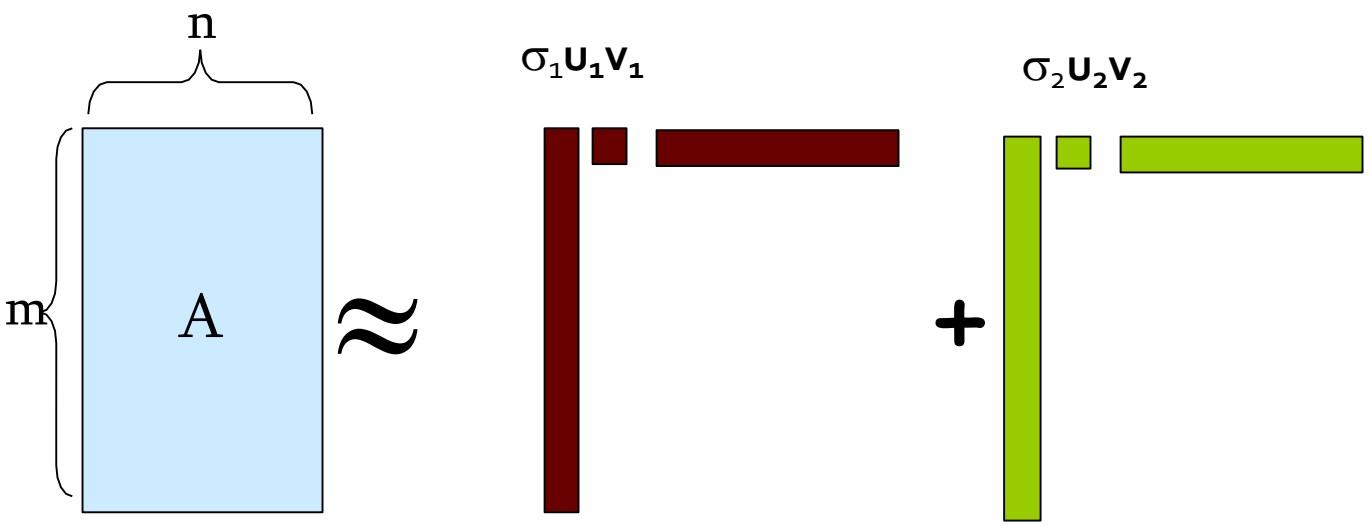
$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$



- **A**: **Input data matrix**
  - *m* x *n* matrix (e.g., *m* documents, *n* terms)
- **U**: **Left singular vectors**
  - *m* x *r* matrix (*m* documents, *r* concepts)
- Σ: **Singular values**
  - *r* x *r* diagonal matrix (strength of each 'concept') (*r* : rank of the matrix **A**)
- **V**: **Right singular vectors**
  - *n* x *r* matrix (*n* terms, *r* concepts)

SVD

$$\mathbf{A} \approx \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$



If we set $\sigma_2 = 0$, then the green columns may as well not exist.

σ_i … scalar

u_i … vector

v_i … vector

SVD – Properties

It is **always** possible to decompose a real matrix A into $A = U \Sigma V^T$, where

- U, Σ, V: unique
- U, V: column orthonormal
  - $U^T U = I$; $V^T V = I$ (I: identity matrix)
  - (Columns are orthogonal unit vectors)
- Σ: diagonal
  - Entries (**singular values**) are non-negative, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$)

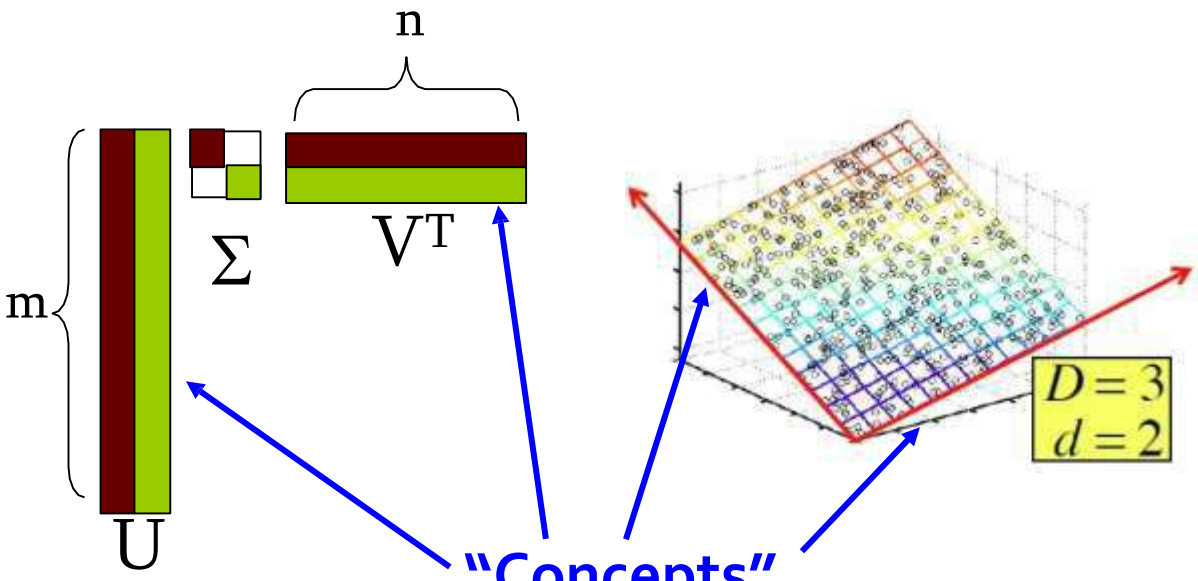
Nice proof of uniqueness: https://www.cs.cornell.edu/courses/cs322/2008sp/stuff/TrefethenBau_Lec4_SVD.pdf

SVD – Example: Users-to-Movies

Consider a matrix. What does SVD do?

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi ↑	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
↓	5	5	5	0	0
Romance ↑	0	2	0	4	4
	0	0	0	5	5
↓	0	1	0	2	2

=



Ratings matrix where each column corresponds to a movie and each row to a user. First 4 users prefer SciFi, while others prefer Romance.

"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD – Example: Users-to-Movies

- **A = U Σ V^T - example: Users to Movies**

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi ↑	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romance ↑	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

=

0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
0.68	0.11	-0.05
0.15	**-0.59**	**0.65**
0.07	**-0.73**	**-0.67**
0.07	**-0.29**	**0.32**

x

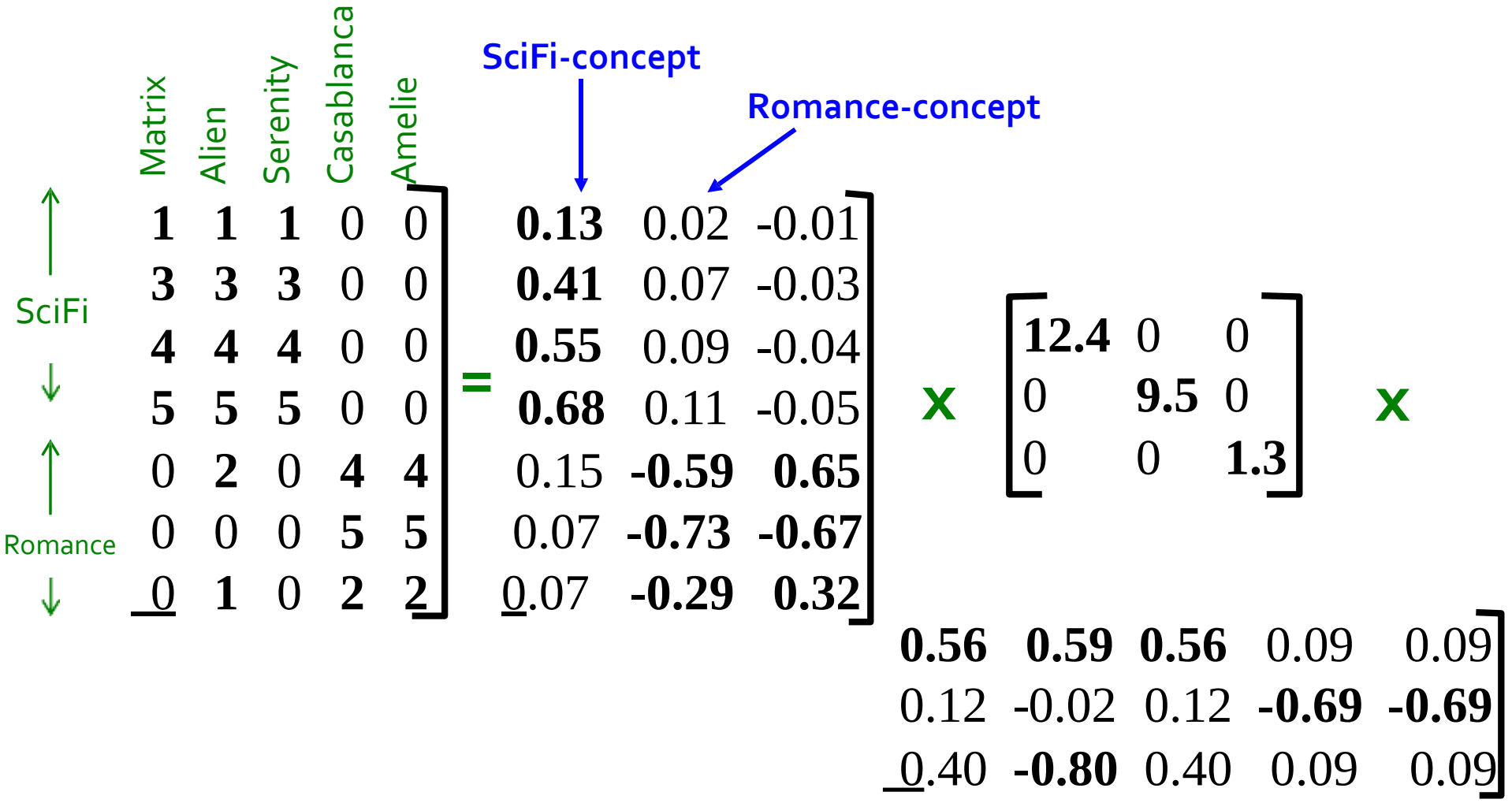
12.4	0	0
0	**9.5**	0
0	0	**1.3**

x

0.56	**0.59**	**0.56**	0.09	0.09
0.12	-0.02	0.12	**-0.69**	**-0.69**
0.40	**-0.80**	0.40	0.09	0.09

SVD – Example: Users-to-Movies

A = U Σ V^T - example: Users to Movies



SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$ - example:

U is "user-to-concept" factor matrix

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romance	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

=

SciFi-concept	Romance-concept	
0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
0.68	0.11	-0.05
0.15	**-0.59**	**0.65**
0.07	**-0.73**	**-0.67**
0.07	**-0.29**	**0.32**

x

12.4	0	0
0	**9.5**	0
0	0	**1.3**

x

0.56	**0.59**	**0.56**	0.09	0.09
0.12	-0.02	0.12	**-0.69**	**-0.69**
0.40	**-0.80**	0.40	0.09	0.09

SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$ - example:

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romance	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

=

SciFi-concept		
0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
0.68	0.11	-0.05
0.15	**-0.59**	**0.65**
0.07	**-0.73**	**-0.67**
0.07	**-0.29**	**0.32**

x

"strength" of the SciFi-concept

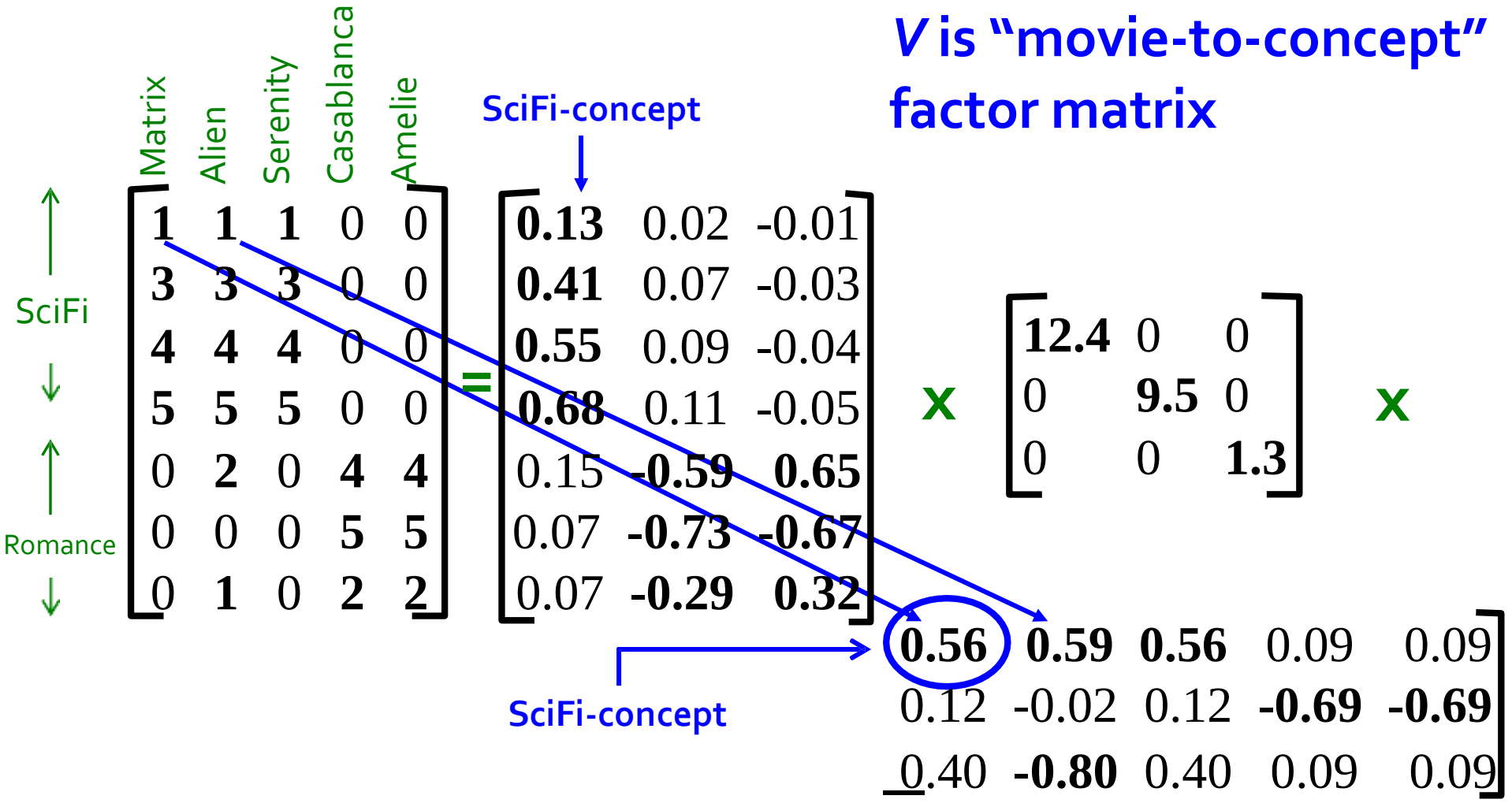
12.4	0	0
0	**9.5**	0
0	0	**1.3**

x

0.56	**0.59**	**0.56**	0.09	0.09
0.12	-0.02	0.12	**-0.69**	**-0.69**
0.40	**-0.80**	0.40	0.09	0.09

SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$ - example:



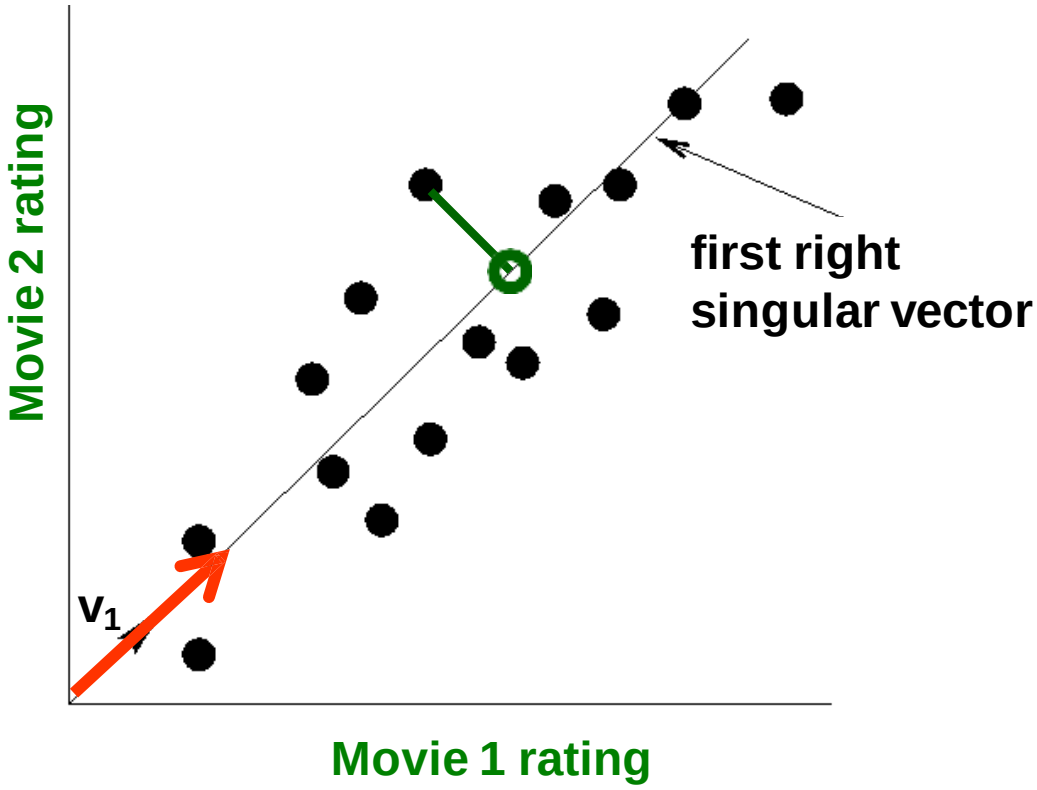
SVD – Interpretation #1

Movies, **users** and **concepts**:

- ***U***: user-to-concept matrix
- ***V***: movie-to-concept matrix
- Σ: its diagonal elements:
  'strength' of each concept

Dimensionality Reduction with SVD

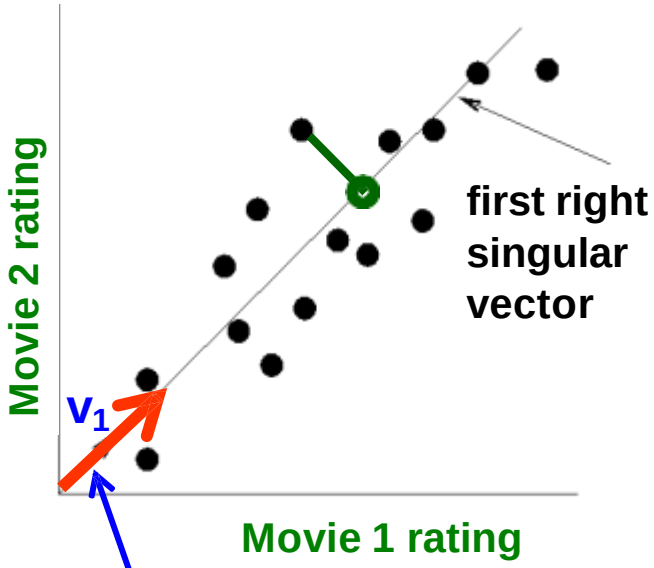
SVD – Dimensionality Reduction



- Instead of using two coordinates (x, y) to describe point positions, let's use only one coordinate
- Point's position is its location along vector v_1

SVD – Dimensionality Reduction

- **A = U Σ V^T - example:**
  - **U**: "user-to-concept" matrix
  - **V**: "movie-to-concept" matrix

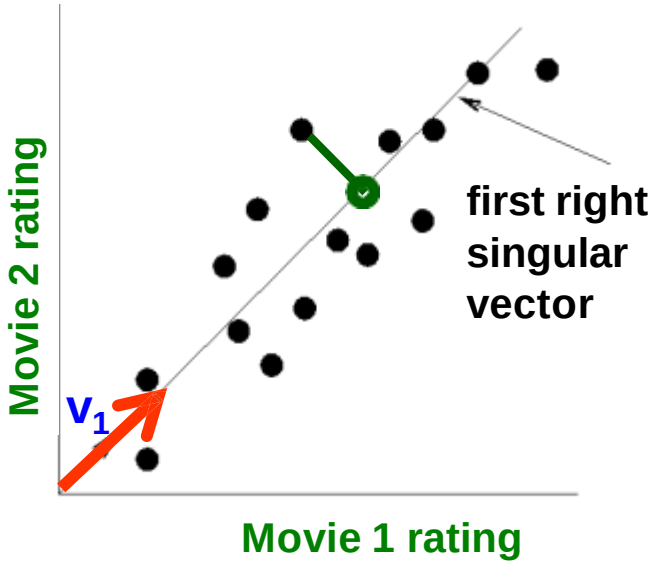


$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times$$

$$\begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Dimensionality Reduction

- **$A = U \Sigma V^T$ - example:**



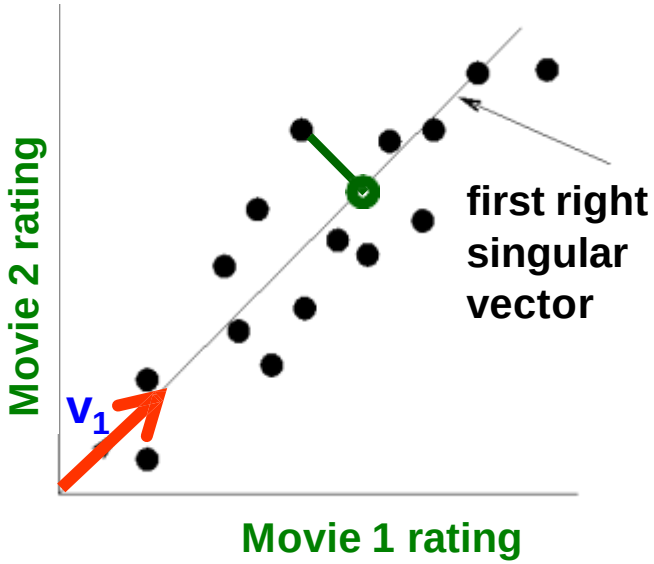
variance ('spread') on the v_1 axis

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Dimensionality Reduction

$A = U \Sigma V^T$ - example:

- $U \Sigma$: Gives the coordinates of the points in the projection axis



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}$$

Projection of users on the "Sci-Fi" axis

$U \Sigma$:

$$\begin{bmatrix} 1.61 & 0.19 & -0.01 \\ 5.08 & 0.66 & -0.03 \\ 6.82 & 0.85 & -0.05 \\ 8.43 & 1.04 & -0.06 \\ 1.86 & -5.60 & 0.84 \\ 0.86 & -6.93 & -0.87 \\ 0.86 & -2.75 & 0.41 \end{bmatrix}$$

SVD – Interpretation #2

More details

- Q: How is dim. reduction done?

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Interpretation #2

More details

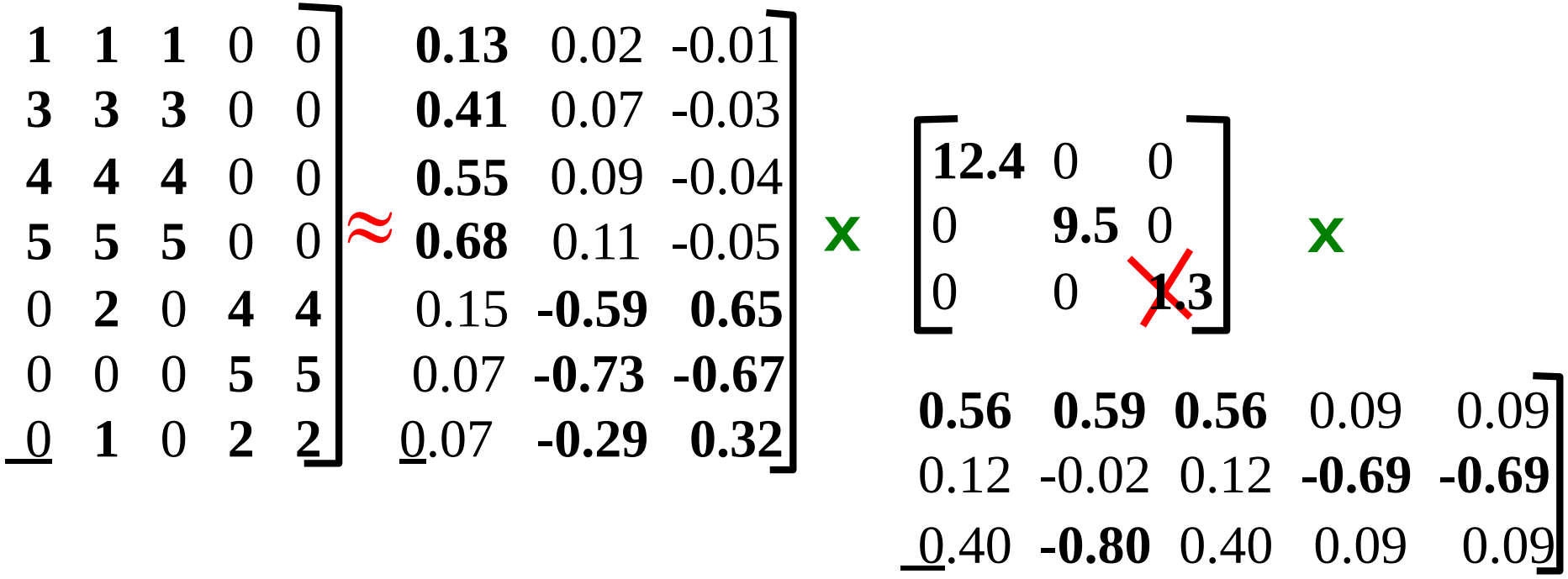
- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Interpretation #2

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero



SVD – Interpretation #2

This is Rank 2 approximation to A. We could also do Rank 1 approx. The larger the rank the more accurate the approximation.

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Interpretation #2

This is Rank 2 approximation to A. We could also do Rank 1 approx. The larger the rank the more accurate the approximation.

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

SVD – Interpretation #2

This is Rank 2 approximation to A. We could also do Rank 1 approx. The larger the rank the more accurate the approximation

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\ 2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\ 3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\ 4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\ 0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\ -0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\ 0.32 & 0.23 & 0.32 & 2.01 & 2.01 \end{bmatrix}$$

Reconstructed data matrix B

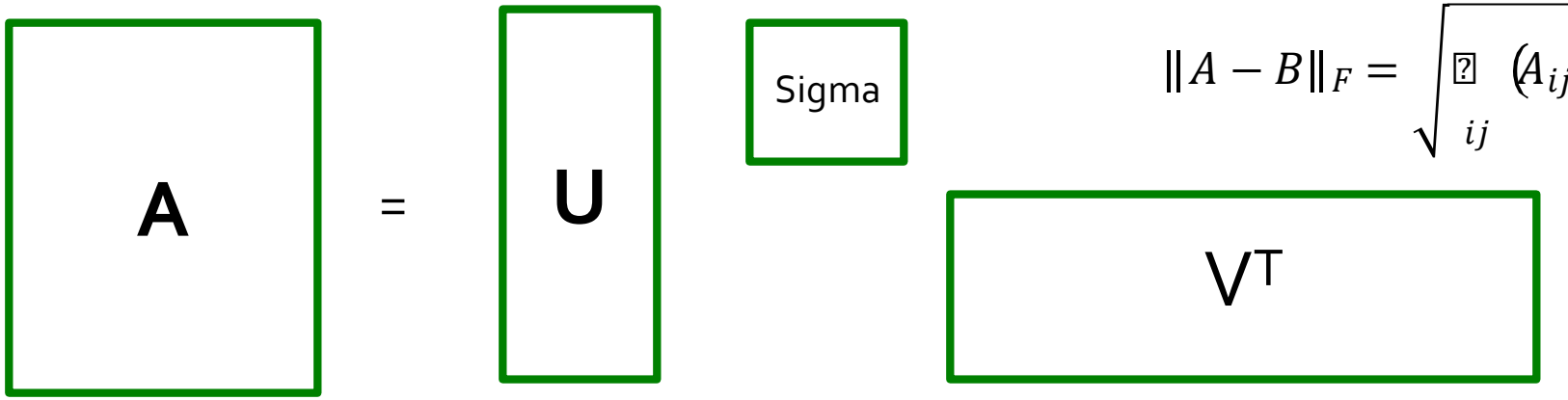
Reconstruction Error is quantified by the Frobenius norm:

$\|M\|_F = \sqrt{\Sigma_{ij} M_{ij}^2}$

$\|A-B\|_F = \sqrt{\Sigma_{ij} (A_{ij}-B_{ij})^2}$ is "small"

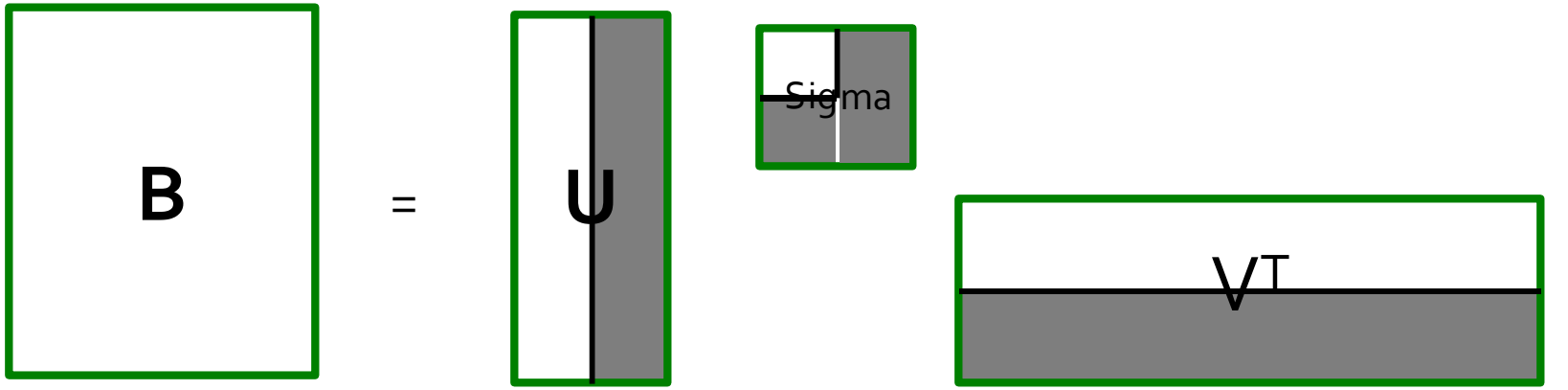
SVD – Best Low Rank Approx.

- **Fact: SVD gives 'best' axis to project on:**
  - '**best**' = minimizing the sum of reconstruction errors



$$\|A - B\|_F = \sqrt{\sum_{ij} \left(A_{ij} - B_{ij}\right)^2}$$

B is best approximation of A:



SVD – Best Low Rank Approx.

- **Theorem:**
  Let **A = U Σ V^T** and **B = U S V^T** where
  S = diagonal *r*x*r* matrix with $s_i=\sigma_i$ ($i=1\ldots k$) else $s_i=0$
  then **B** is a **best** rank(**B**)=***k*** approx. to ***A***

What do we mean by "best":

- B **is a solution to** $\min_B \|A-B\|_F$ **where** $\text{rank}(B)=k$

$$\underset{m\times n}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}} = \underset{m\times r}{\begin{pmatrix} u_{11} & \cdots & \\ \vdots & \ddots & \\ u_{m1} & & \end{pmatrix}} \underset{r\times r}{\begin{pmatrix} \sigma_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & \end{pmatrix}} \underset{r\times n}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ & & \end{pmatrix}}$$

$$\|A-B\|_F = \sqrt{\sum_{ij} \left(A_{ij}-B_{ij}\right)^2}$$

SVD – Conclusions so far

- **SVD: A= U Σ V^T: unique**
  - **U**: user-to-concept factors
  - **V**: movie-to-concept factors
  - Σ : strength of each concept
- **Q: So what's a good value for r (# of latent factors)?**
- Let the *energy* of a set of singular values be the sum of their squares.
- Pick r so the retained singular values have at least 90% of the total energy.
- **Back to our example:**
  - With singular values 12.4, 9.5, and 1.3, total energy = 245.7
  - If we drop 1.3, whose square is only 1.7, we are left with energy 244, or over 99% of the total

How to Compute SVD

How to Compute the SVD

- **Start by supposing $\boldsymbol{A = U\Sigma V^T}$**
- $A^T = (U\Sigma V^T)^T = (V^T)^T\Sigma^T U^T = V\Sigma U^T$
  - **Why?** (1) Rule for transpose of a product; (2) the transpose of the transpose and the transpose of a diagonal matrix are both the identity functions
- $\boldsymbol{A^T A = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T}$
  - **Why?** U is orthonormal, so $U^T U$ is an identity matrix
  - Also note that Σ^2 is a diagonal matrix whose i-th element is the square of the i-th element of Σ
- $\boldsymbol{A^T AV = V\Sigma^2 V^T V = V\Sigma^2}$
  - **Why?** V is also orthonormal

Computing the SVD –(2)

- Since $\boldsymbol{A^T A = V\Sigma^2 V^T \rightarrow ATAV = V\Sigma^2}$
  - **Note** that therefore the i-th column of V is an eigenvector of $A^T A$, and its eigenvalue is the i-th element of Σ^2
- Thus, we can find V and Σ by finding the eigenpairs for $A^T A$
  - Once we have the eigenvalues in Σ^2, we can find the singular values by taking the square root of these eigenvalues
- Symmetric argument, AA^T gives us U

SVD – Complexity

- **To compute the full SVD using specialized methods:**
  - **$O(nm^2)$** or **$O(n^2m)$** (whichever is less)
- **But:**
  - Less work, if we just want singular values
  - or if we want the first *k* singular vectors
  - or if the matrix is sparse
- **Implemented in** linear algebra packages like
  - LINPACK, Matlab, SPlus, Mathematica …

Example of SVD

Case study: How to query?

- Q: Find users that like 'Matrix'
- A: Map query into a 'concept space' – how?

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi ↑	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romance ↑	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

=

0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
0.68	0.11	-0.05
0.15	**-0.59**	**0.65**
0.07	**-0.73**	**-0.67**
0.07	**-0.29**	**0.32**

x

12.4	0	0
0	**9.5**	0
0	0	**1.3**

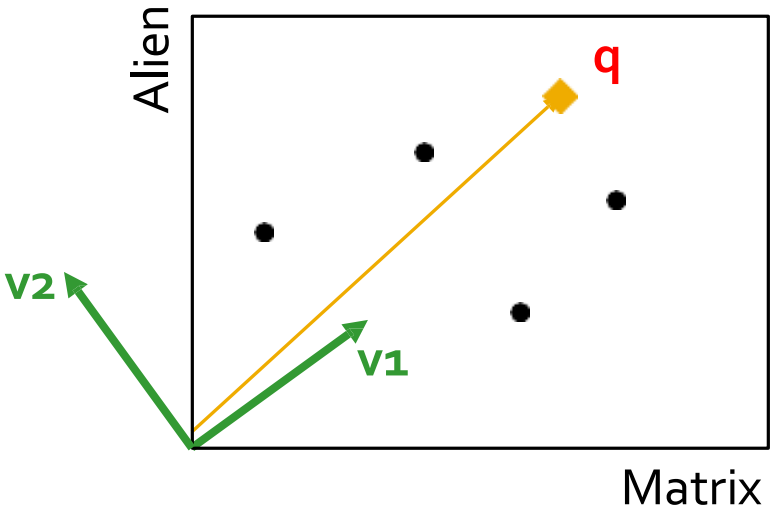
x

0.56	**0.59**	**0.56**	0.09	0.09
0.12	-0.02	0.12	**-0.69**	**-0.69**
0.40	**-0.80**	0.40	0.09	0.09

Case study: How to query?

- Q: Find users that like 'Matrix'
- A: Map query into a 'concept space' – how?

	Matrix	Alien	Serenity	Casablanca	Amelie
q =	5	0	0	0	0



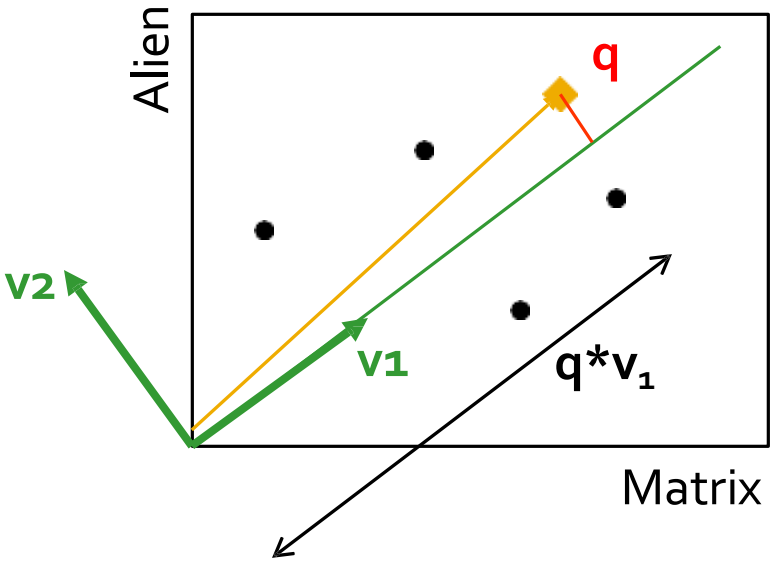
Project into concept space:
Inner product with each 'concept' vector $\mathbf{v}_i$

Case study: How to query?

- Q: Find users that like 'Matrix'
- A: Map query into a 'concept space' – how?

$$q = \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Project into concept space:
Inner product with each 'concept' vector $\mathbf{v_i}$

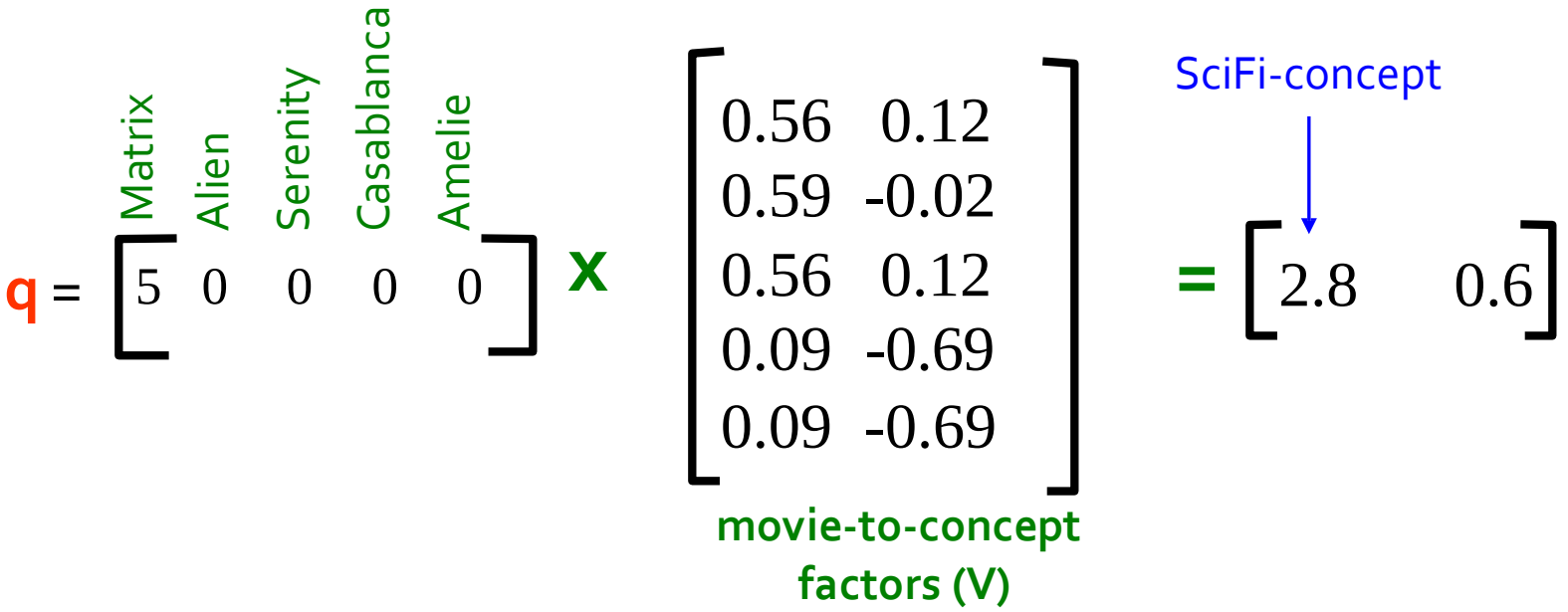


Case study: How to query?

Compactly, we have:

$$q_{concept} = q\ V$$

E.g.:

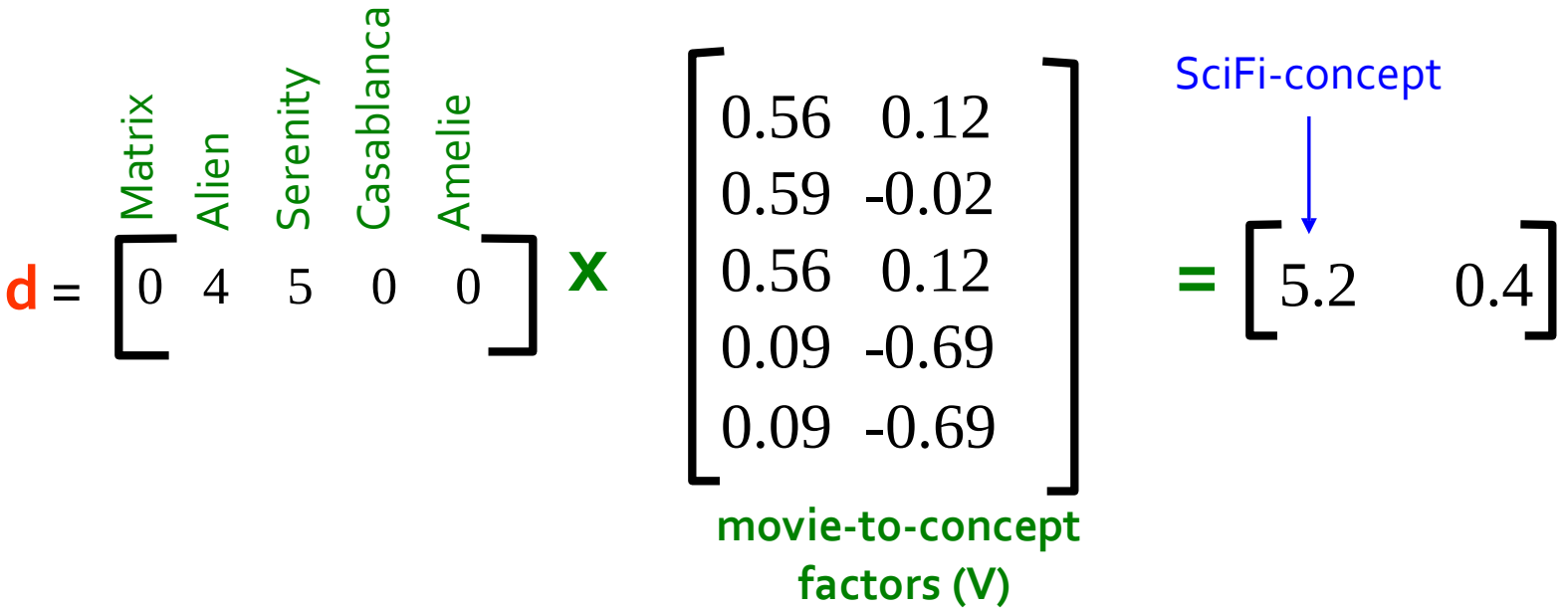


Case study: How to query?

- How would the user *d* that rated ('Alien', 'Serenity') be handled?

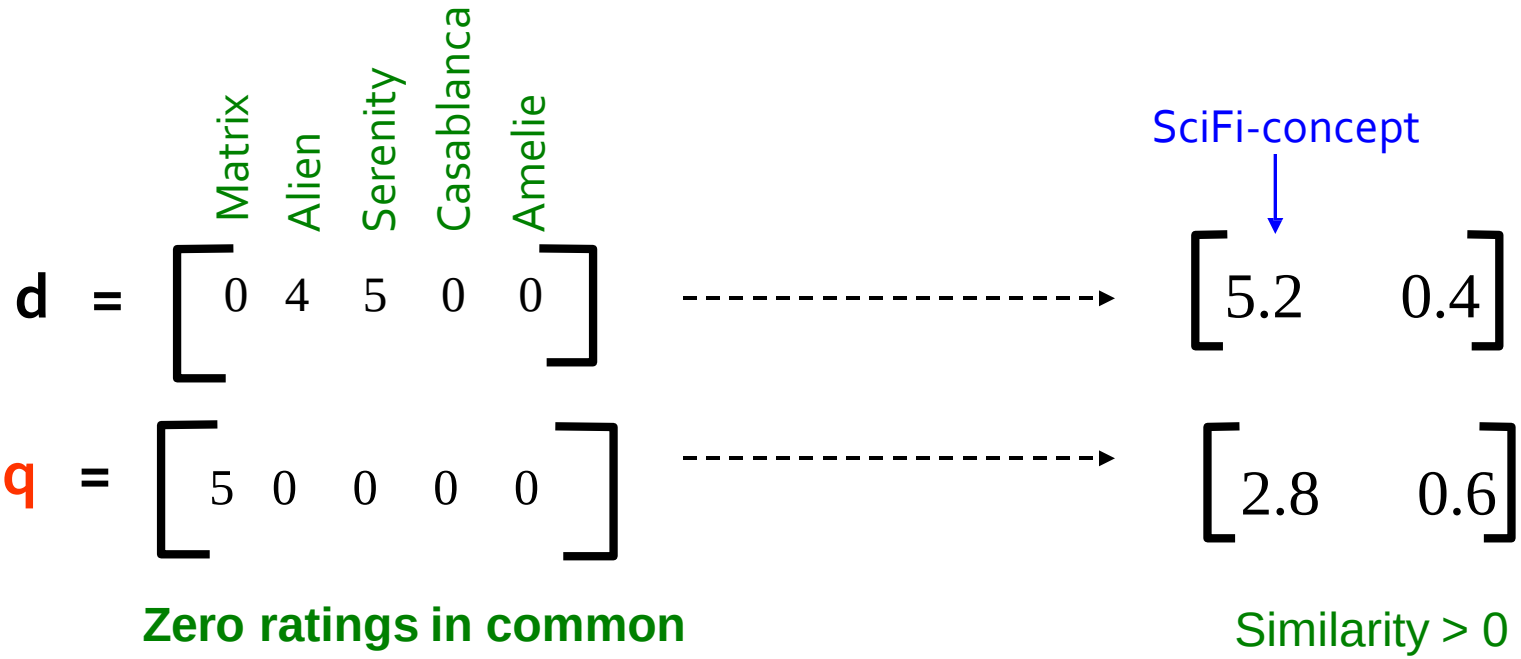
  $d_{concept} = d\ V$

E.g.:



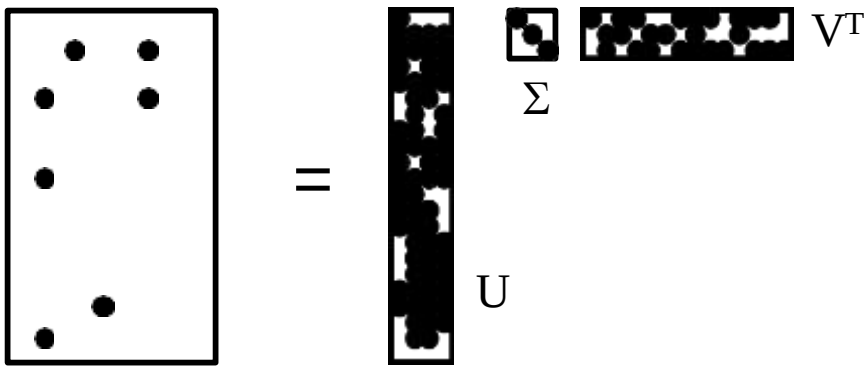
Case study: How to query?

- **Observation:** User ***d*** that rated (‘*Alien*’, ‘*Serenity*’) will be **similar** to user **q** that rated (‘*Matrix*’), although ***d*** and **q** have **zero ratings in common**!



SVD: Drawbacks

- **Optimal low-rank approximation** in terms of Frobenius norm
- **Interpretability problem:**
  - A singular vector specifies a linear combination of all input columns or rows
- **Lack of sparsity:**
  - Singular vectors are **dense!**



What is SVD?

The SVD is the Swiss Army knife of matrix decompositions

—Diane O'Leary, 2006

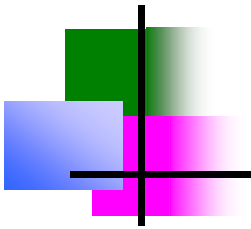
Matrix Decomposition

- Eigen Decomposition
- A (non-zero) vector **v** of dimension *N* is an **eigenvector** of a square ($N \times N$) matrix **A** if it satisfies the linear equation

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- Where λ is **eigenvalue** corresponding to **v**

- What if it is not a square matrix?



What is SVD?

Singular Value Decomposition is based on a theorem from linear algebra which says the following:
a rectangular matrix A can be broken down into the product of three matrices:

- an orthogonal matrix U,
- a diagonal matrix S,
- the transpose of an orthogonal matrix V.

What is SVD?

Let A be an $m \times n$ real matrix. Then there exist orthogonal matrices U of size $m \times m$ and V of size $n \times n$ such that

$$A = USV^T$$

where S is an $m \times n$ matrix with nondiagonal entries all zero and $s_{11} \geq s_{12} \geq \cdots \geq s_{pp} \geq 0, \quad p = \min\{m, n\}$.

- the diagonal entries of S are called the *singular values* of A,
- the columns of U are called the *left singular vectors* of A,
- the columns of V are called the *right singular vectors* of A,
- the factorization USV^T is called the *singular value decomposition* of A.

Example

Let's find the singular value decomposition of $A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$.

In order to find U, we start with AA^T.

$$AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

- Eigenvalues: $\lambda_1 = 12$ and $\lambda_2 = 10$.
- Eigenvectors: $\vec{u_1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\vec{u_2} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

Gram-Schmidt Orthonormalization Process

More generally, if we have an orthonormal set of vectors $\vec{u_1}, .., \vec{u_{k-1}}$, then $\vec{w_k}$ is expressed as

$$\vec{w_k} = \vec{v_k} - \sum_{i=1}^{k-1} \vec{u_t} \cdot \vec{v_k} * \vec{u_t}$$

Scalar Multiplication

$1.5 * [3, 6, 8, 4] = [4.5, 9, 12, 6]$. More generally, *scalar multiplication* means if d is a real number and $\vec{v}$ is a vector $[v_1, v_2, ..., v_n]$, then $d * \vec{v} = [dv_1, dv_2, ..., dv_n]$.

Inner Product

$$(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

For example, if $\vec{x} = [1, 6, 7, 4]$ and $\vec{y} = [3, 2, 8, 3]$, then

$$\vec{x} \cdot \vec{y} = 1(3) + 6(2) + 7(8) + 3(4) = 83$$

$$\vec{u_1} = \frac{\vec{v_1}}{|\vec{v_1}|} = \frac{[1,1]}{\sqrt{1^2+1^2}} = \frac{[1,1]}{\sqrt{2}} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$$

Compute

$$\vec{w_2} = \vec{v_2} - \vec{u_1} \cdot \vec{v_2} * \vec{u_1} =$$

$$[1,-1] - [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \cdot [1,-1] * [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] =$$

$$[1,-1] - 0 * [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] = [1,-1] - [0,0] = [1,-1]$$

and normalize

$$\vec{u_2} = \frac{\vec{w_2}}{|\vec{w_2}|} = [\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]$$

to give

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Example

The calculation of V is similar. V is based on A^TA, so we have

$$A^TA = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

We find the eigenvalues of A^TA

- Eigenvalues: $\lambda_1 = 12$, $\lambda_2 = 10$ and $\lambda_3 = 0$.
- Eigenvectors: $\vec{u}_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$, $\vec{u}_2 = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$ and $\vec{u}_3 = \begin{bmatrix} 1 \\ 2 \\ -5 \end{bmatrix}$.

and use the Gram-Schmidt orthonormalization process to convert that to an orthonormal matrix.

$$\vec{u_1} = \frac{\vec{v_1}}{|\vec{v_1}|} = [\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}]$$

$$\vec{w_2} = \vec{v_2} - \vec{u_1} \cdot \vec{v_2} * \vec{u_1} = [2, -1, 0]$$

$$\vec{u_2} = \frac{\vec{w_2}}{|\vec{w_2}|} = [\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, 0]$$

$$\vec{w_3} = \vec{v_3} - \vec{u_1} \cdot \vec{v_3} * \vec{u_1} - \vec{u_2} \cdot \vec{v_3} * \vec{u_2} = [\frac{-2}{3}, \frac{-4}{3}, \frac{10}{3}]$$

$$\vec{u_3} = \frac{\vec{w_3}}{|\vec{w_3}|} = [\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{-5}{\sqrt{30}}]$$

All this to give us

$$V = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

when we really want its transpose

$$V^T = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

Example

$$S = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

Now we have all the pieces of the puzzle

$$A_{mn} = U_{mm} S_{mn} V^T_{nn} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} =$$

$$\begin{bmatrix} \frac{\sqrt{12}}{\sqrt{2}} & \frac{\sqrt{10}}{\sqrt{2}} & 0 \\ \frac{\sqrt{12}}{\sqrt{2}} & \frac{-\sqrt{10}}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

Exercise

Example: Find the SVD of A, $U\Sigma V^T$, where $A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$.

First we compute the singular values σ_i by finding the eigenvalues of AA^T.

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}.$$

The characteristic polynomial is $det(AA^T - \lambda I) = \lambda^2 - 34\lambda + 225 = (\lambda - 25)(\lambda - 9)$, so the singular values are $\sigma_1 = \sqrt{25} = 5$ and $\sigma_2 = \sqrt{9} = 3$.

Answer to Exercise

So at this point we know that

$$A = U\Sigma V^T = U \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}.$$

Finally, we can compute U by the formula $\sigma u_i = Av_i$, or $u_i = \frac{1}{\sigma} Av_i$. This gives $U = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$. So in its full glory the SVD is:

$$A = U\Sigma V^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}.$$

SVD Algorithm

$$A = USV^T \tag{2.66}$$

Here, A is the given matrix of dimension $m \times n$, U is the orthogonal matrix whose dimension is $m \times n$, S is the diagonal matrix of dimension $n \times n$, and V is the orthogonal matrix. The procedure for finding decomposition matrix is given as follows:

SVD Algorithm

1. For a given matrix, find AA^T
2. Find eigen values of AA^T
3. Sort the eigen values in a descending order. Pack the eigen vectors as a matrix U.
4. Arrange the square root of the eigen values in diagonal. This matrix is diagonal matrix, S.
5. Find eigen values and eigen vectors for A^TA. Find the eigen value and pack the eigen vector as a matrix called V.

SVD Example

Example 2.13: Find SVD of the matrix:

$$A = \begin{pmatrix} 1 & 2 \\ 4 & 9 \end{pmatrix}$$

Solution: The first step is to compute:

$$AA^T = \begin{pmatrix} 1 & 2 \\ 4 & 9 \end{pmatrix}\begin{pmatrix} 1 & 4 \\ 2 & 9 \end{pmatrix} = \begin{pmatrix} 5 & 22 \\ 22 & 97 \end{pmatrix}$$

The eigen value and eigen vector of this matrix can be calculated to get U. The eigen values of this matrix are 0.0098 and 101.9902.

SVD Example

The eigen vectors of this matrix are:

$$u_1 = \begin{pmatrix} 0.2268 \\ 1 \end{pmatrix}$$

$$u_2 = \begin{pmatrix} -4.4086 \\ 1 \end{pmatrix}$$

These vectors are normalized to get the vectors respectively as:

$$u_1 = \begin{pmatrix} 0.2212 \\ 0.9752 \end{pmatrix}$$

$$u_2 = \begin{pmatrix} -0.9752 \\ 0.2212 \end{pmatrix}$$

The matrix U can be obtained by concatenating the above vector as:

$$U = [u_1, u_2] = \begin{pmatrix} 0.2212 & -0.9752 \\ 0.9752 & 0.2212 \end{pmatrix}$$

SVD Example

The matrix V can be obtained by finding A^TA. It is $\begin{pmatrix} 17 & 38 \\ 38 & 85 \end{pmatrix}$. The eigen values are 0.0098 and 101.9902. The eigen vectors can be found as follows:

$$v_1 = \begin{pmatrix} 0.447 \\ 1 \end{pmatrix} \text{ when } \lambda = 101.99$$

$$v_2 = \begin{pmatrix} -2.236 \\ 1 \end{pmatrix} \text{ when } \lambda = 0.0098$$

The above can be normalized as follows:

$$v_1 = \begin{pmatrix} 0.4082 \\ 0.9129 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} -0.9129 \\ 0.4082 \end{pmatrix}$$

The matrix V can be obtained by concatenating the above vector as:

$$V = [v_1 \; v_2] = \begin{pmatrix} 0.4081 & -0.9129 \\ 0.9129 & 0.4082 \end{pmatrix}$$

SVD Example

The matrix S can be found as the diagonal matrix as:

$$S = \begin{pmatrix} \sqrt{101.9902} & 0 \\ 0 & \sqrt{0.0098} \end{pmatrix} = \begin{pmatrix} 10.099 & 0 \\ 0 & 0.099 \end{pmatrix}$$

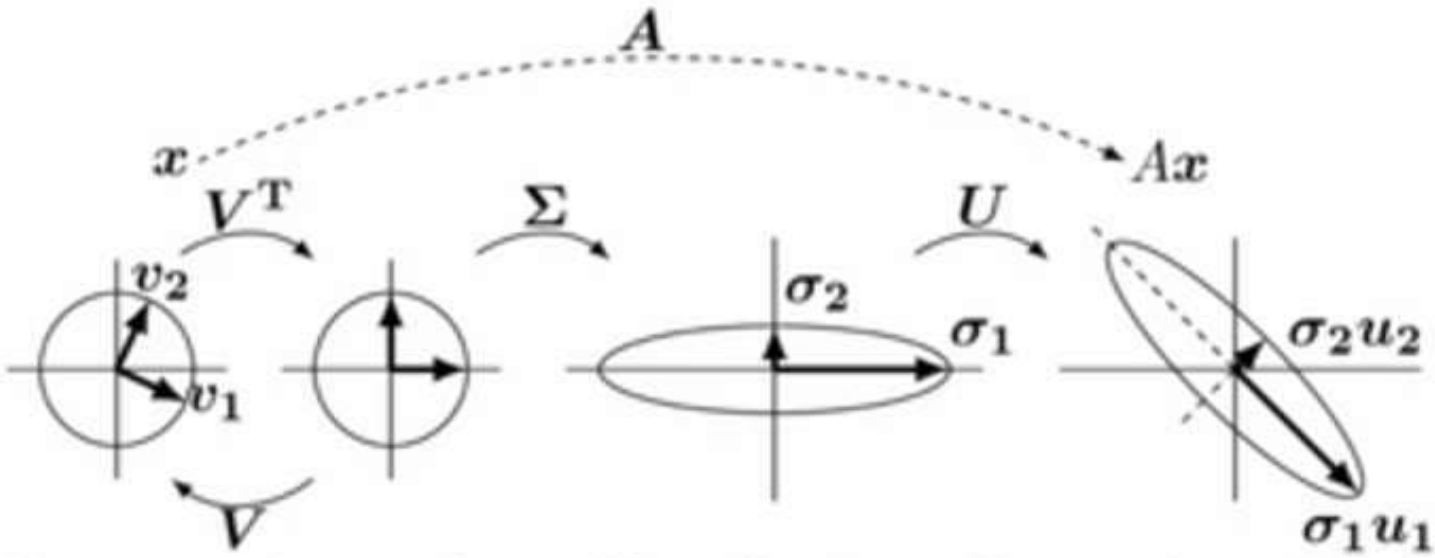
Therefore, the matrix decomposition $A = U\,SV^T$ is complete.

$$A = U\Sigma V^{\mathrm{T}} \text{ with } U^{\mathrm{T}}U = I \text{ and } V^{\mathrm{T}}V = I$$

$AV = U\Sigma$ means

$$A\begin{bmatrix} \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_r \end{bmatrix}\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \text{ and } A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$$

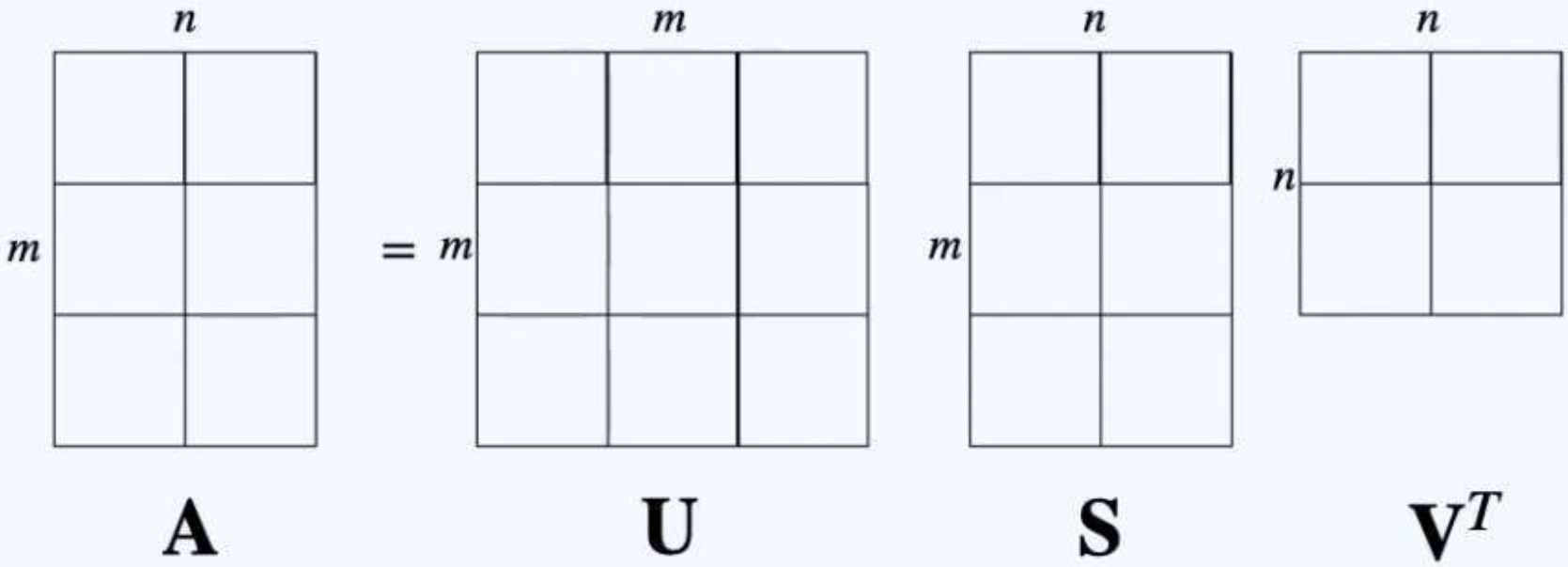
SINGULAR VALUES $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ $\quad r = \text{rank of } A$



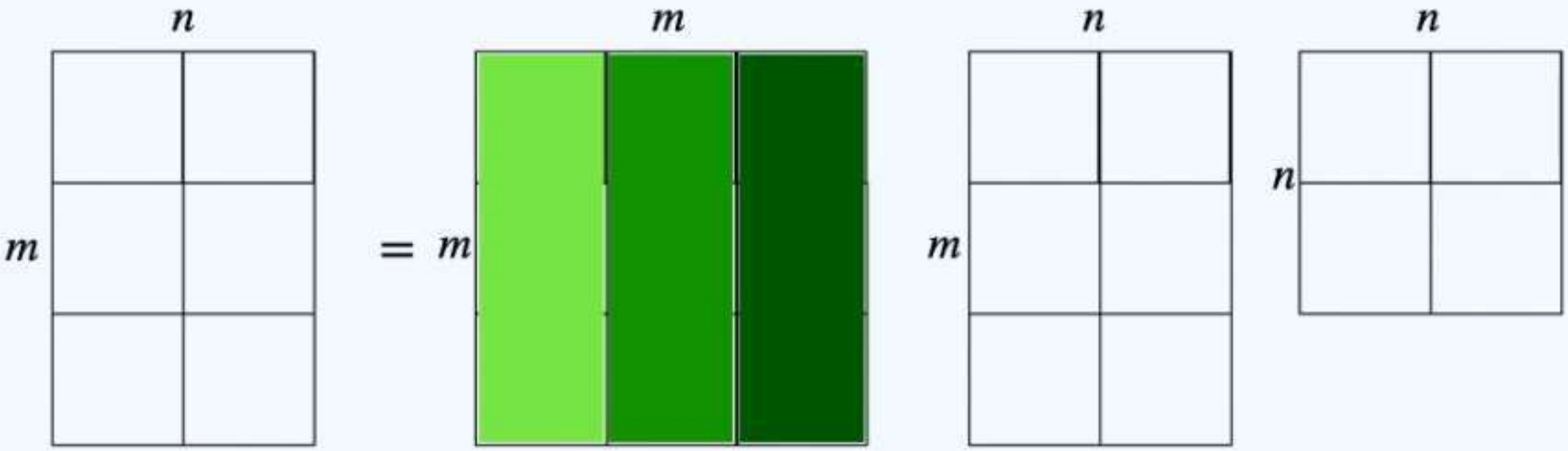
U and V are rotations and possible reflections. Σ stretches circle to ellipse.

SVD Example

- Consider a 2-dimensional array $\mathbf{A} \in \mathbb{R}^{m \times n}, m > n$



SVD Example

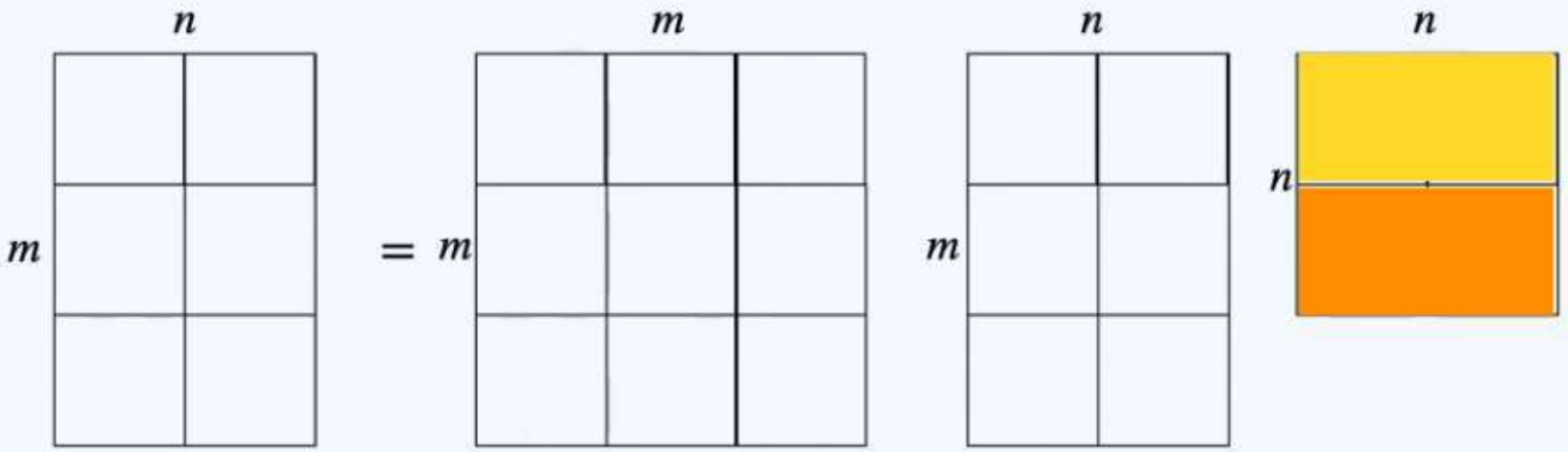


$\mathbf{U}$

Left singular vectors

$$\mathbf{U}^{-1} = \mathbf{U}^{T}$$

SVD Example

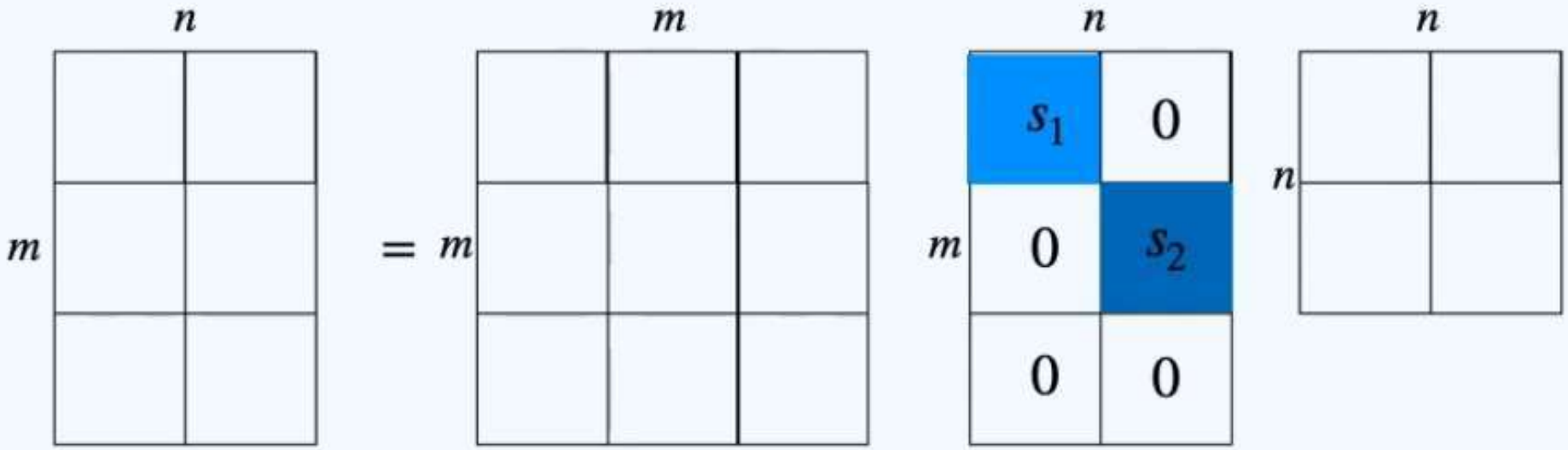


$$\mathbf{V}^T$$

Right singular vectors

$$\mathbf{V}^{-1} = \mathbf{V}^T$$

SVD Example



S

singular values

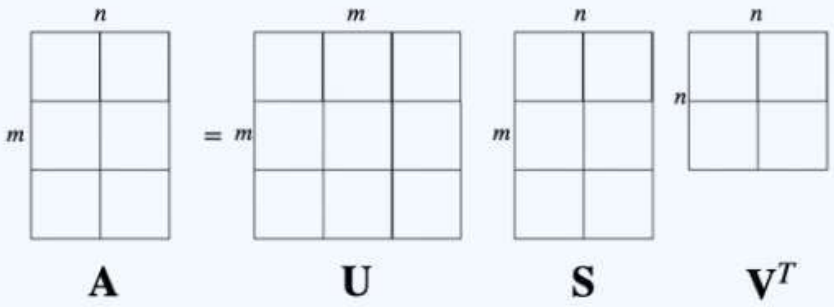
$$s_1 \geq s_2 \geq \ldots$$

SVD Example

```
import numpy as np
from numpy.linalg import svd
```

```
A = np.array([[1,-0.8],
              [0,1],
              [1,0]
])
```

```
U, S, VT = svd(A, full_matrices=True)
```




```
print("Left Singular Vectors:")
print(U)
print("Singular Values:")
print(S)
print("Right Singular Vectors:")
print(np.transpose(VT))
```

```
Left Singular Vectors:
[[-7.88170109e-01  1.87057766e-16 -6.15457455e-01]
 [ 3.84473224e-01 -7.80868809e-01 -4.92365964e-01]
 [-4.80591530e-01 -6.24695048e-01  6.15457455e-01]]
Singular Values:
[1.62480768 1.        ]
Right Singular Vectors:
[[-0.78086881 -0.62469505]
 [ 0.62469505 -0.78086881]]
```

SVD Example

```
smat = np.zeros((3, 2))
smat[:2,:2] = np.diag(S)
print(smat)
```

```
[[1.62480768 0.        ]
 [0.         1.        ]
 [0.         0.        ]]
```

```
from numpy.linalg import multi_dot
print(multi_dot([U,smat,VT]),'\n', A)
```

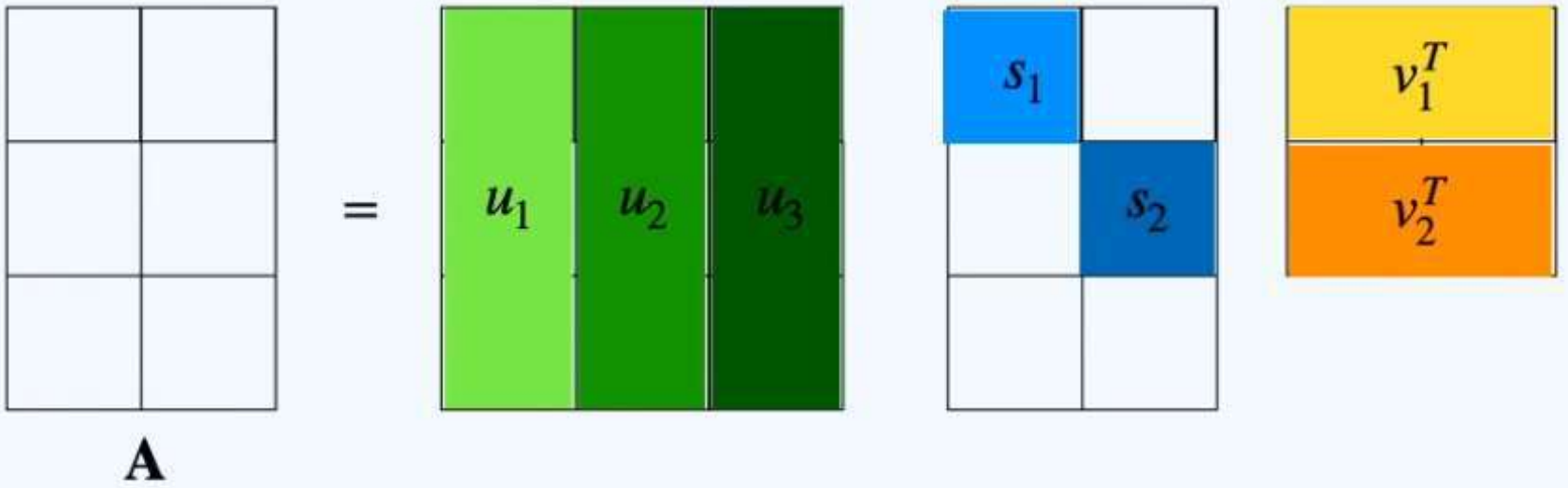
```
[[ 1.00000000e+00 -8.00000000e-01]
 [ 8.93385967e-18  1.00000000e+00]
 [ 1.00000000e+00  1.23451001e-16]]
 [[ 1.  -0.8]
 [ 0.   1. ]
 [ 1.   0. ]]
```

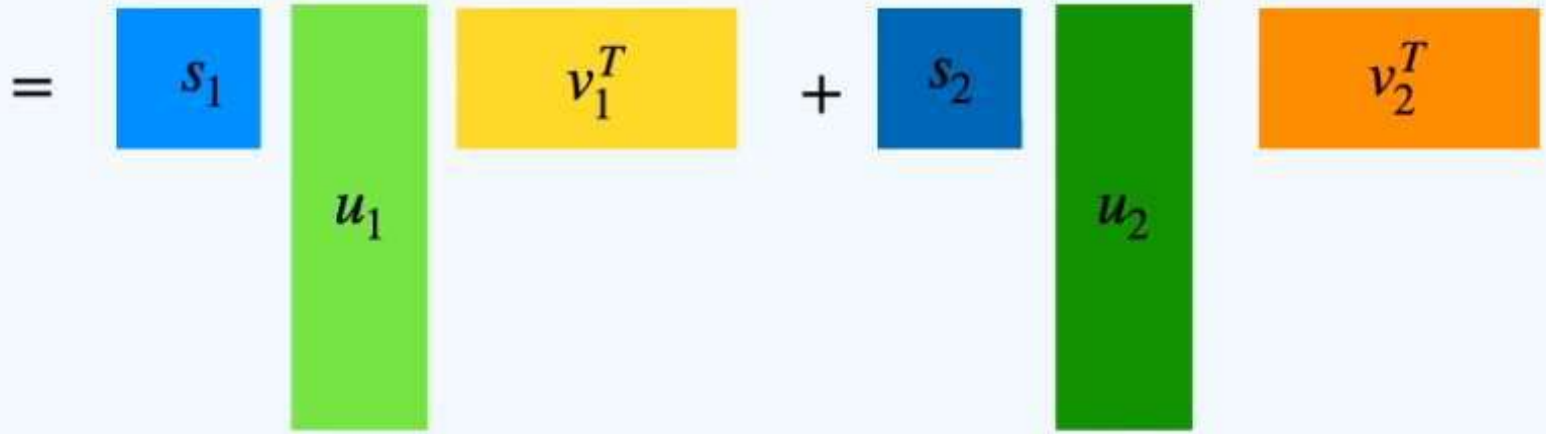
```
from numpy.linalg import norm

print(norm(A - multi_dot([U,smat,VT])))
```

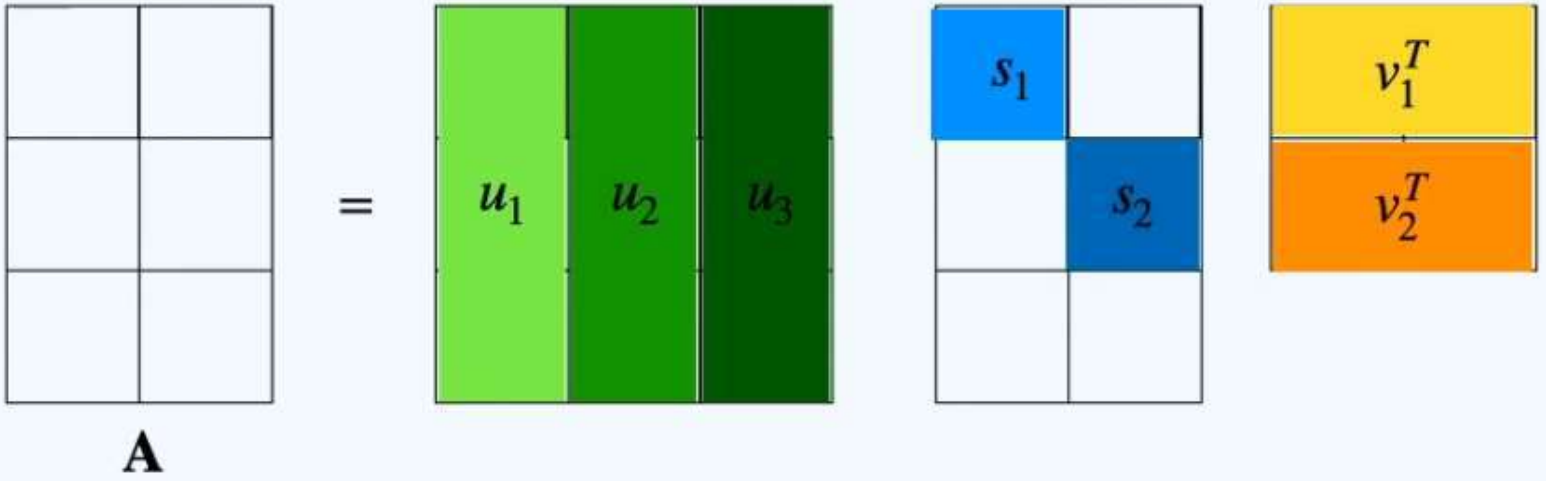
```
4.741952567914139e-16
```

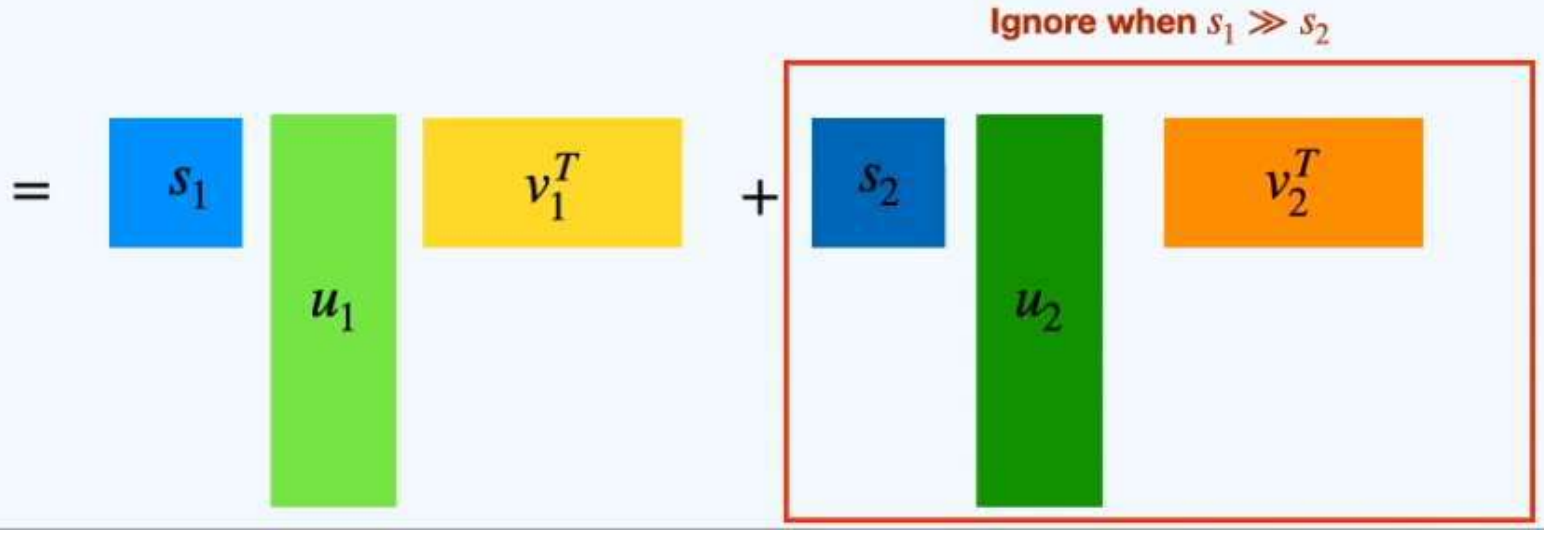
Application of SVD





Application of SVD





Low Rank Approximation

```
print(A)
```

```
[[0.16492833 0.61643322 0.27304301]
 [0.23179766 0.86636283 0.38374687]
 [0.03342004 0.12491015 0.05532772]
 [0.22841804 0.85373123 0.37815182]
 [0.07690245 0.28742925 0.12731395]]
```

```
from numpy.linalg import svd
U, S, VT = svd(A, full_matrices = False)
print(U.shape, S.shape,VT.shape)
```

```
(5, 3) (3,) (3, 3)
```

```
print(S)
```

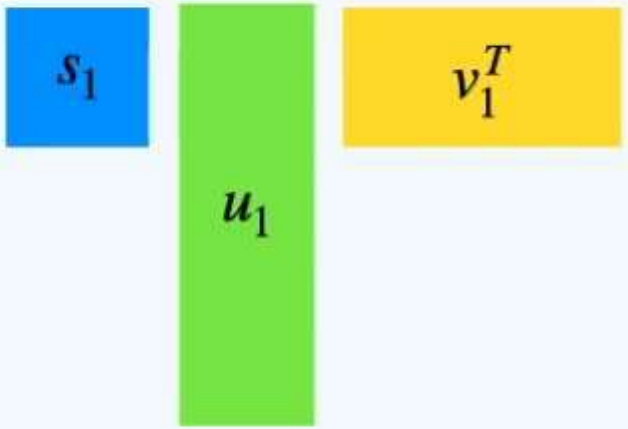
```
[1.57539511e+00 2.99922151e-17 1.96684159e-17]
```

Low Rank Approximation

```
rank = 1
U_sub = U[:,:rank]
VT_sub = VT[:rank,:]
S_sub = np.diag(S[:rank])
A_low_rank = np.dot(np.dot(U_sub,S_sub),VT_sub)

print(A_low_rank)
```

```
[[0.16492833 0.61643322 0.27304301]
 [0.23179766 0.86636283 0.38374687]
 [0.03342004 0.12491015 0.05532772]
 [0.22841804 0.85373123 0.37815182]
 [0.07690245 0.28742925 0.12731395]]
```




```
from numpy.linalg import norm

print(norm(A - A_low_rank))
```

```
3.9836745275876 67e-16
```

Linear Discriminant Analysis (LDA) Algorithm

Remember that PCA is a method to find the linear combination that accounts for as much variability as possible.

$$PC = \alpha_1 X_1 + \alpha_2 X_2$$

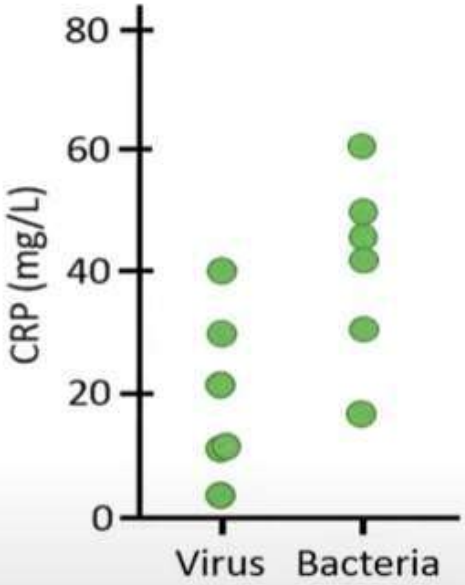
$$\alpha_1^2 + \alpha_2^2 = 1$$

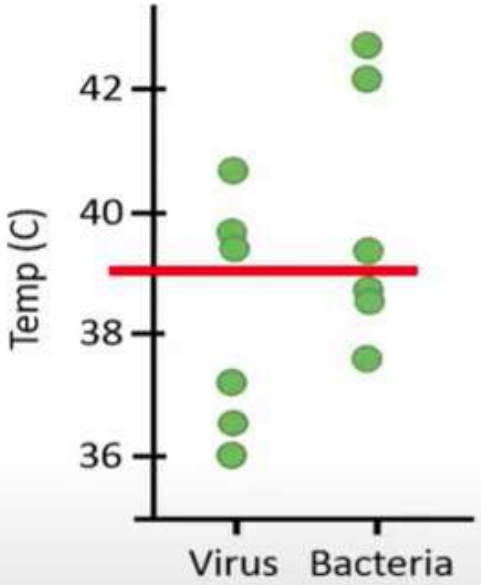
Linear discriminant analysis (LDA) is a similar method, which aims to maximize the separation between two or more groups.

$$LD = \alpha_1 X_1 + \alpha_2 X_2$$

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

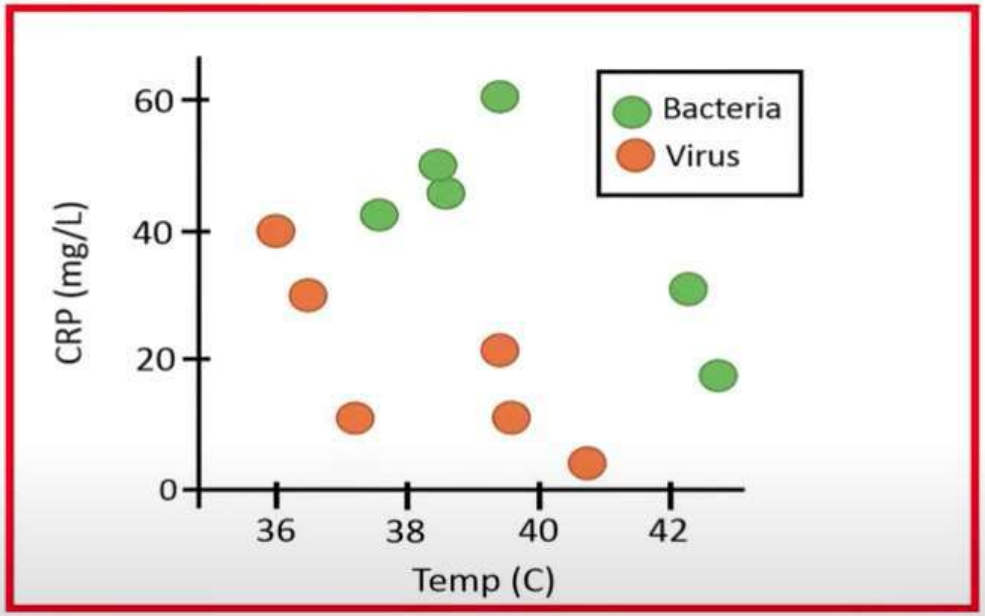




As you see, we have the same problem again. There is no line that can separate the two groups of patients.

Linear Discriminant Analysis (LDA) Algorithm

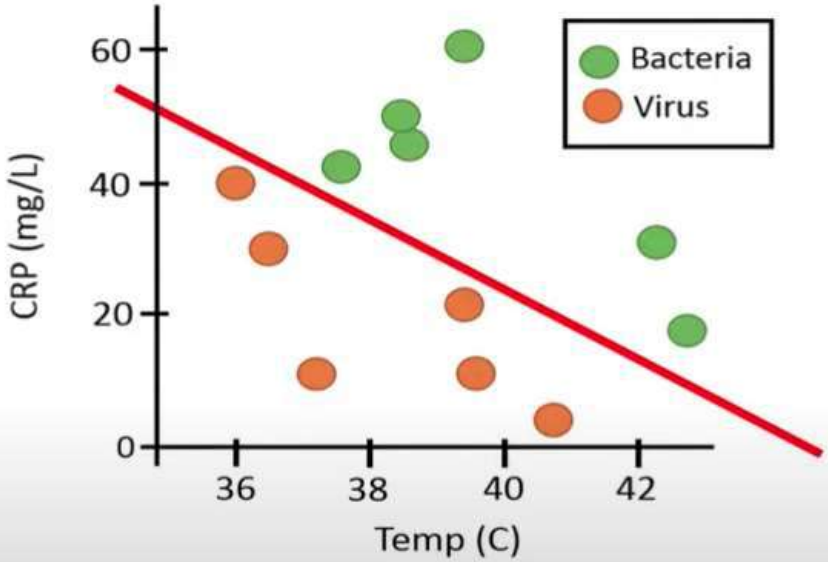
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



However, if we plot the CRP concentration and the body temperature in the same plot and use different colors for patients with a viral infection and bacterial infection,

Linear Discriminant Analysis (LDA) Algorithm

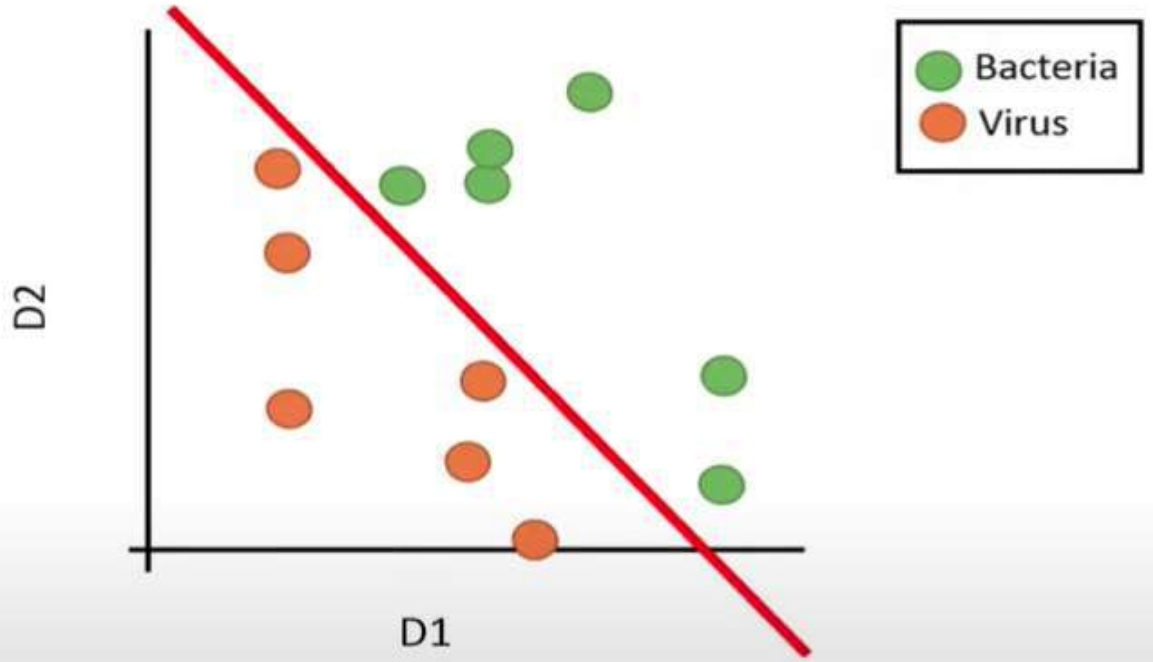
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



we can see that the following line can separate the two groups completely. By combining the two variables, we can get a better separation between the two groups compared to if we use one of the variables alone.

Linear Discriminant Analysis (LDA) Algorithm

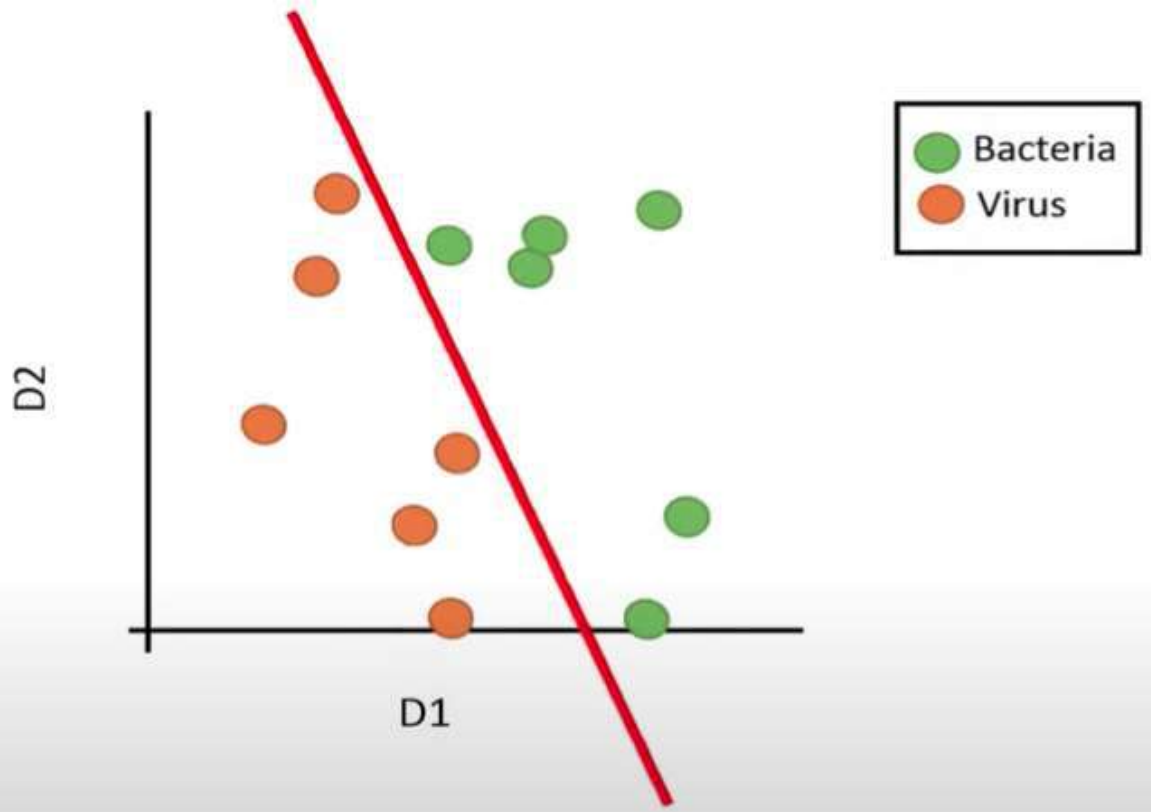
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



Just as PCA, LDA can be seen as we rotate the data into two new dimensions like this.

Linear Discriminant Analysis (LDA) Algorithm

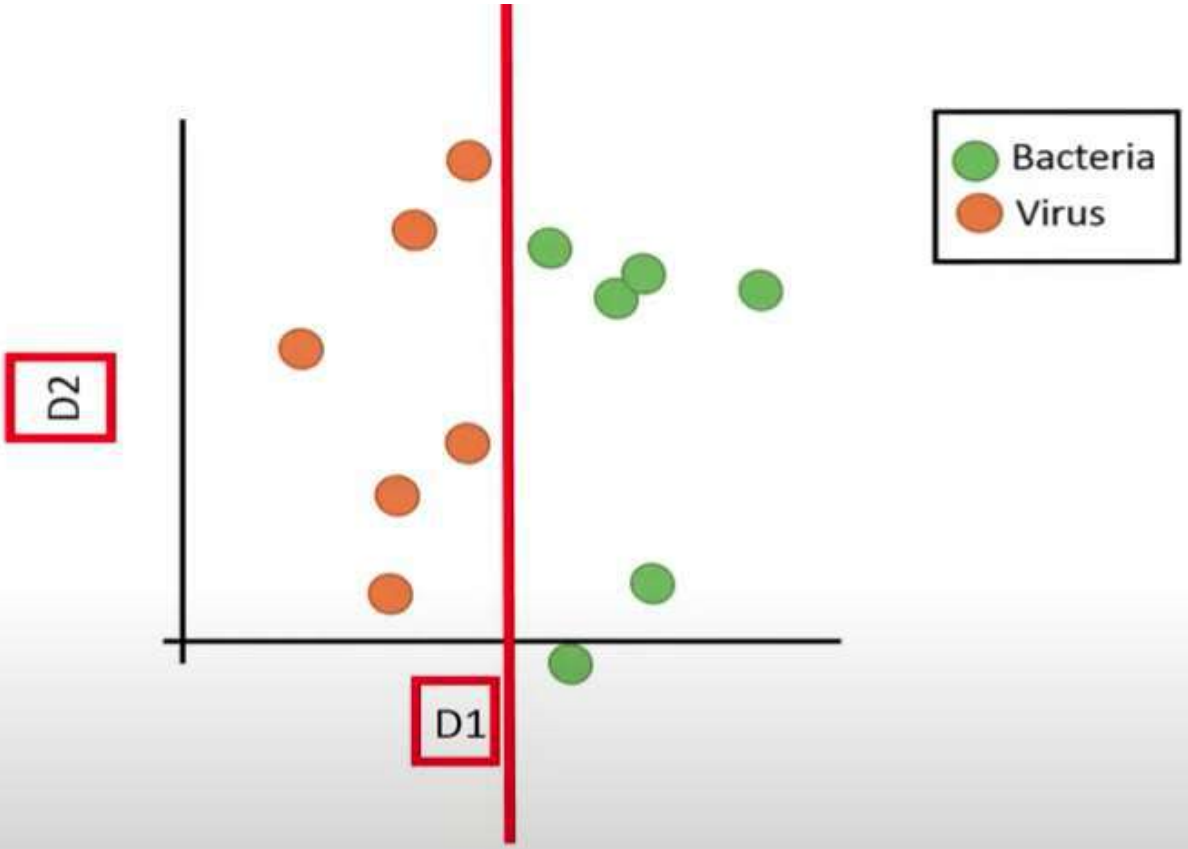
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



Just as PCA, LDA can be seen as we rotate the data into two new dimensions like this.

Linear Discriminant Analysis (LDA) Algorithm

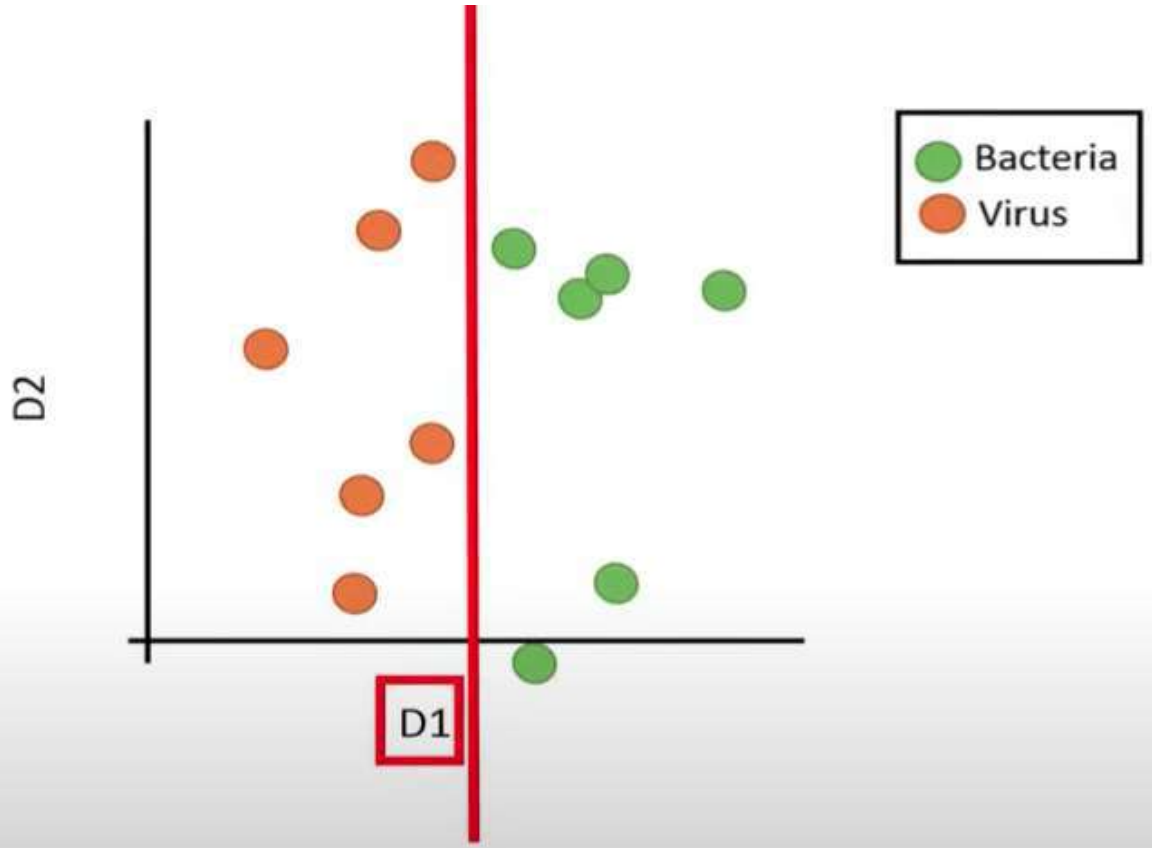
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



The new dimensions are here called discriminant function one and two.

Linear Discriminant Analysis (LDA) Algorithm

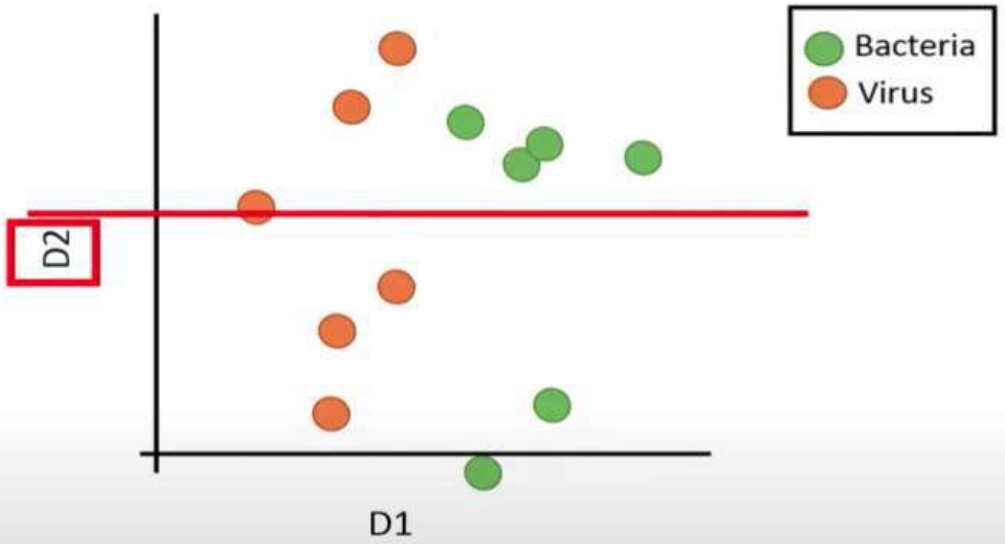
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



We can see that the first discriminant function can be used to separate the two groups perfectly.

Linear Discriminant Analysis (LDA) Algorithm

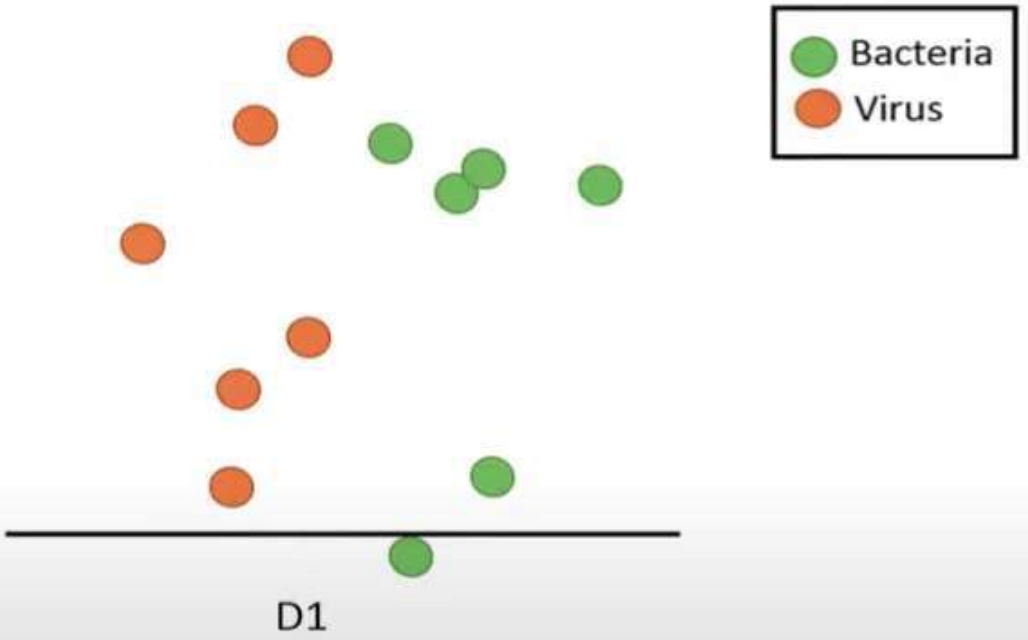
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



In contrast, we cannot separate the groups based on a horizontal line, which means that the second discriminant function is not useful for separation.

Linear Discriminant Analysis (LDA) Algorithm

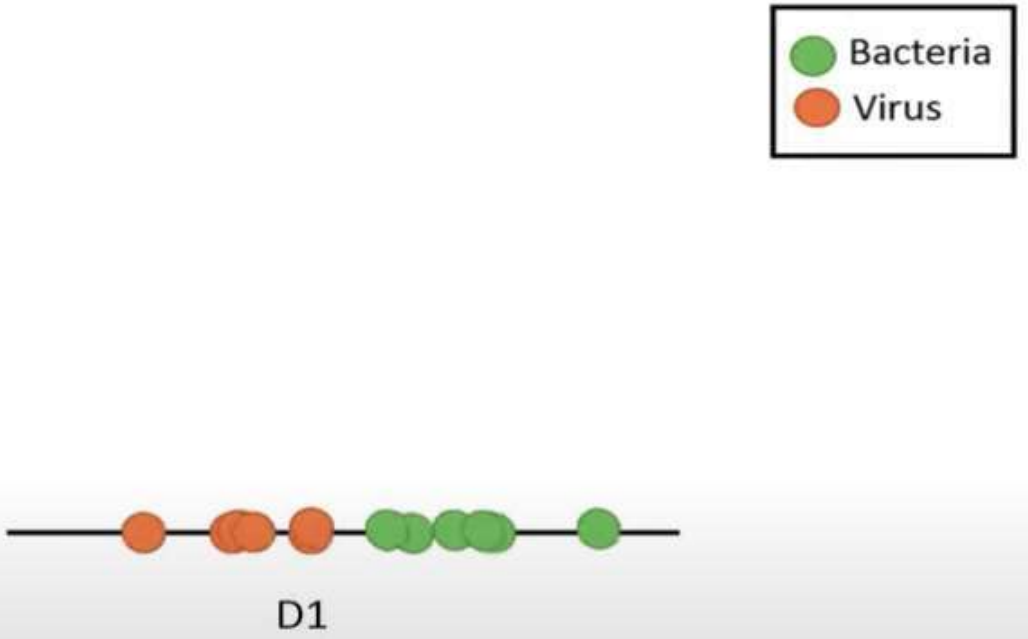
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



We can therefore delete the second discriminant function and place all data points on just one line.

Linear Discriminant Analysis (LDA) Algorithm

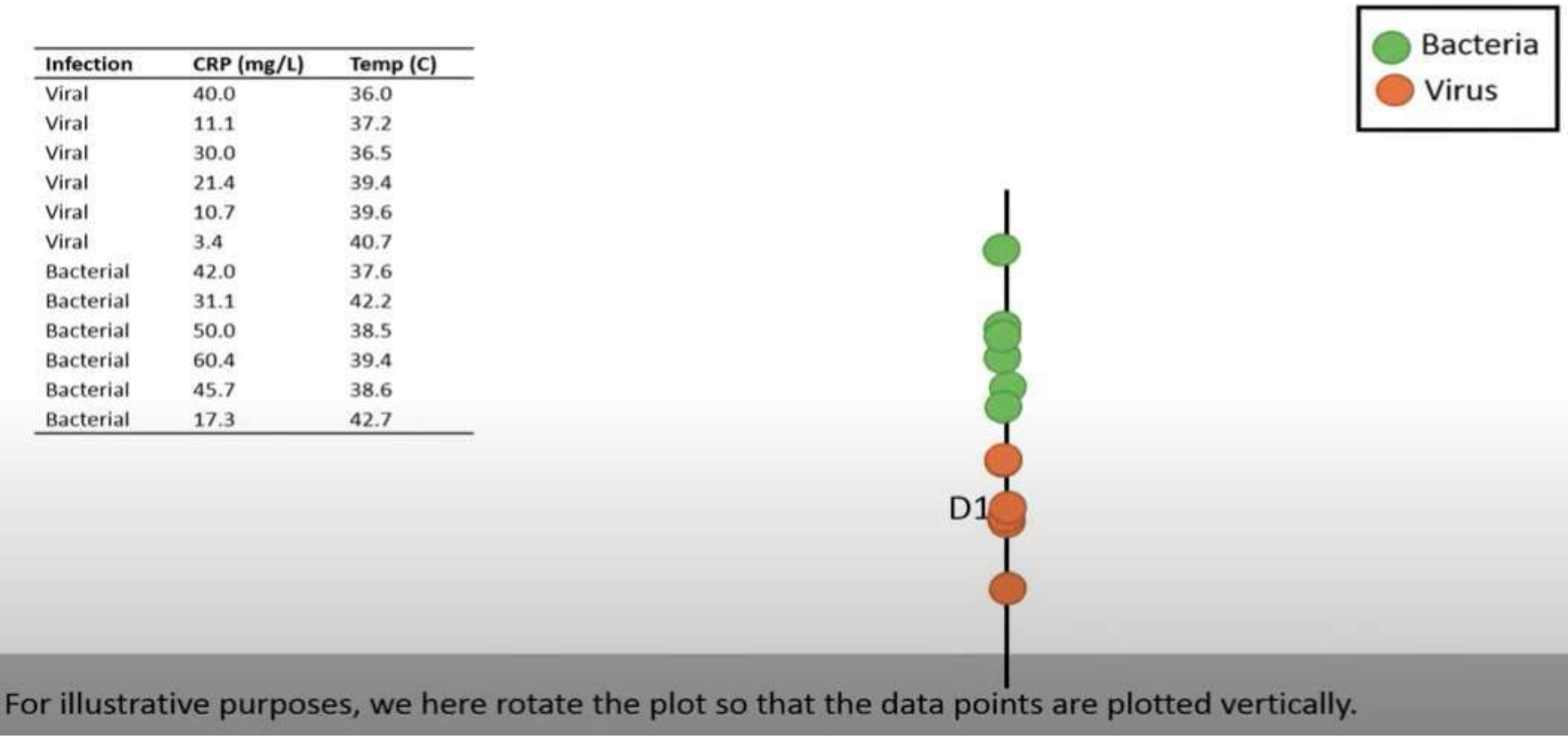
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



We can therefore delete the second discriminant function and place all data points on just one line.

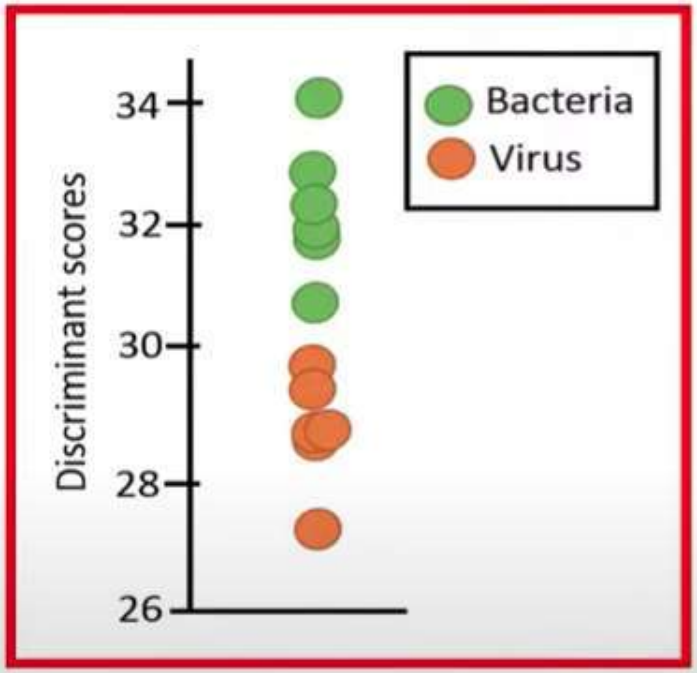
Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



Linear Discriminant Analysis (LDA) Algorithm

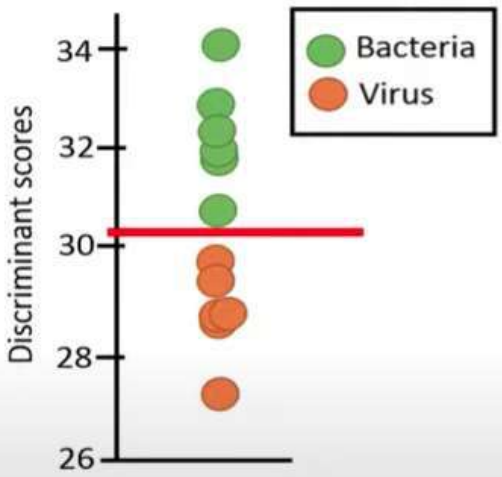
Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7



If we use LDA, we can plot the so-called discriminant scores like this.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

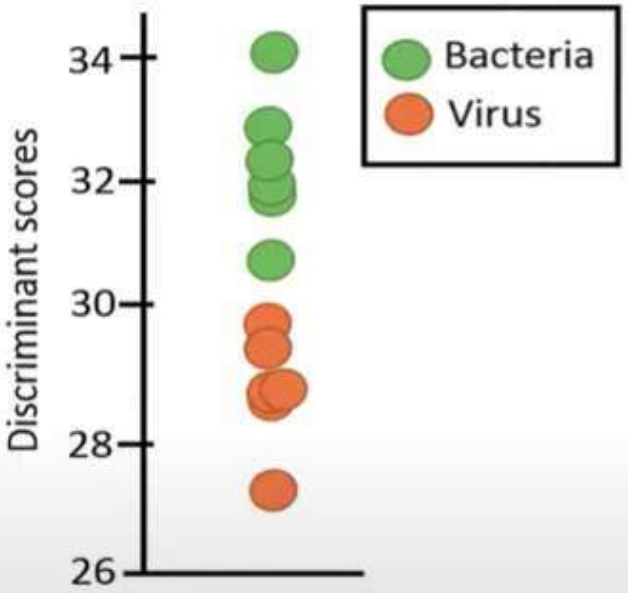


We can now place a line like this that gives us a perfect separation between the two groups. By using LDA, we have combined the variables CRP and body temperature in a way that has maximized the separation between the two groups.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = \alpha_1 X_1 + \alpha_2 X_2$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

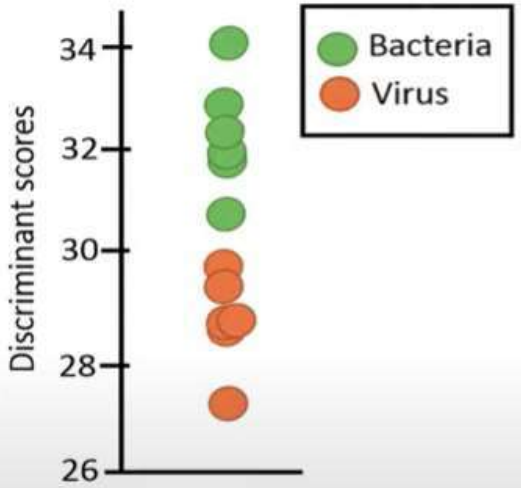


The linear discriminant scores that you see in the plot can be obtained by the following equation.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \text{CRP} + 0.70 \cdot \text{Temp}$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

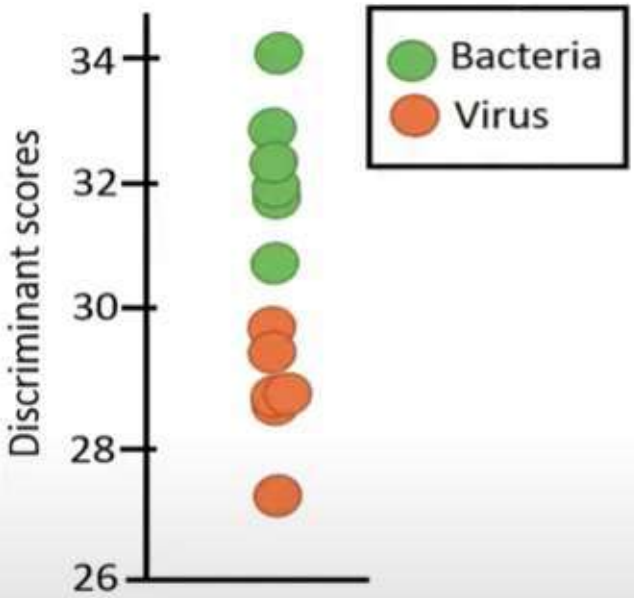


The values of our weights, alpha 1 and alpha 2, were computed to 0.11 and 0.70. These values give us the optimal separation between the groups. We will later see how these weights are calculated.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot 40.0 + 0.70 \cdot 36.0$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	
Viral	11.1	37.2	
Viral	30.0	36.5	
Viral	21.4	39.4	
Viral	10.7	39.6	
Viral	3.4	40.7	
Bacterial	42.0	37.6	
Bacterial	31.1	42.2	
Bacterial	50.0	38.5	
Bacterial	60.4	39.4	
Bacterial	45.7	38.6	
Bacterial	17.3	42.7	

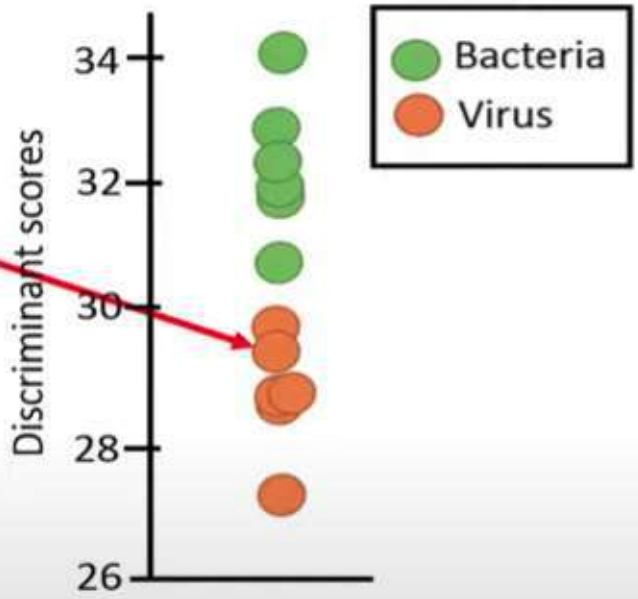


If we plug in the CRP value and the body temperature for the first patient in this equation,

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot 40.0 + 0.70 \cdot 36.0$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	29.5
Viral	11.1	37.2	
Viral	30.0	36.5	
Viral	21.4	39.4	
Viral	10.7	39.6	
Viral	3.4	40.7	
Bacterial	42.0	37.6	
Bacterial	31.1	42.2	
Bacterial	50.0	38.5	
Bacterial	60.4	39.4	
Bacterial	45.7	38.6	
Bacterial	17.3	42.7	

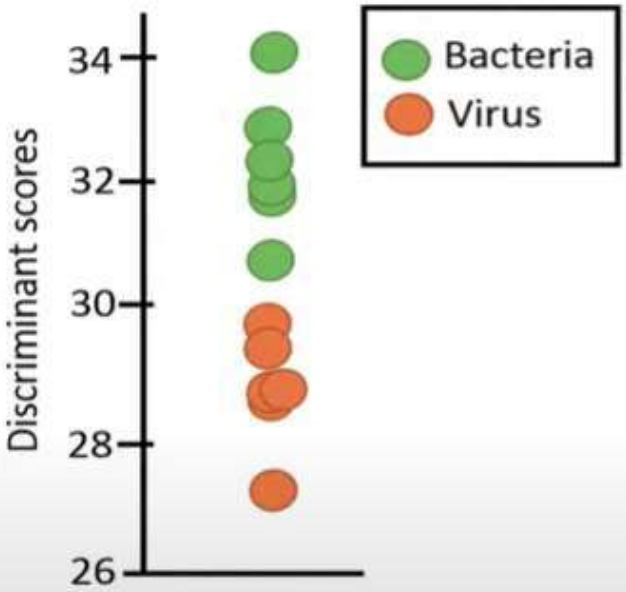


This score is represented by the following point in the plot.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \text{CRP} + 0.70 \cdot \text{Temp}$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	29.5
Viral	11.1	37.2	27.3
Viral	30.0	36.5	28.8
Viral	21.4	39.4	29.9
Viral	10.7	39.6	28.9
Viral	3.4	40.7	28.9
Bacterial	42.0	37.6	30.8
Bacterial	31.1	42.2	32.9
Bacterial	50.0	38.5	32.3
Bacterial	60.4	39.4	34.0
Bacterial	45.7	38.6	31.9
Bacterial	17.3	42.7	31.8

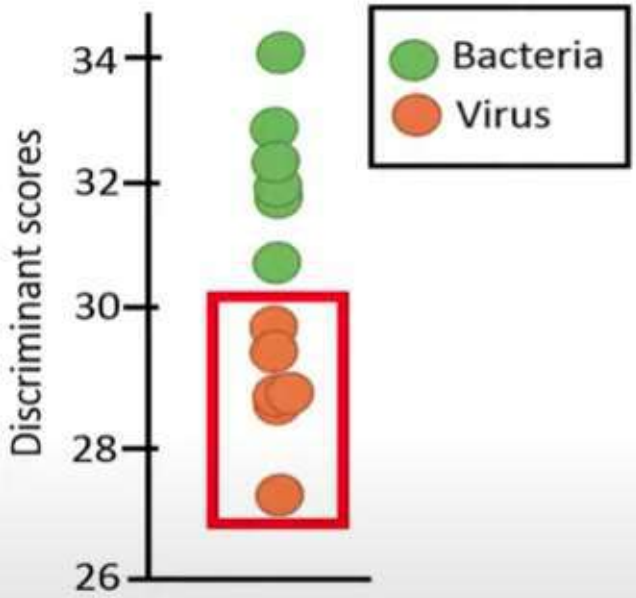


We use the same equation to compute the scores for the other patients as well.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \text{CRP} + 0.70 \cdot \text{Temp}$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	29.5
Viral	11.1	37.2	27.3
Viral	30.0	36.5	28.8
Viral	21.4	39.4	29.9
Viral	10.7	39.6	28.9
Viral	3.4	40.7	28.9
Bacterial	42.0	37.6	30.8
Bacterial	31.1	42.2	32.9
Bacterial	50.0	38.5	32.3
Bacterial	60.4	39.4	34.0
Bacterial	45.7	38.6	31.9
Bacterial	17.3	42.7	31.8

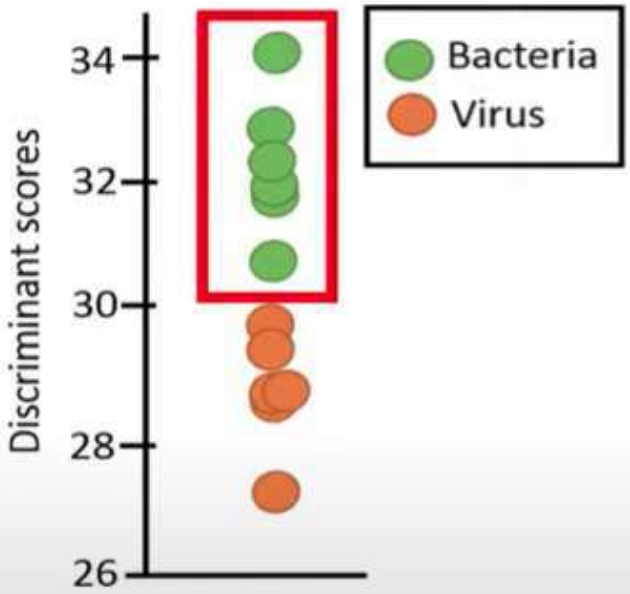


We can see that the scores for the ones with a viral infection are lower,

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \text{CRP} + 0.70 \cdot \text{Temp}$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	29.5
Viral	11.1	37.2	27.3
Viral	30.0	36.5	28.8
Viral	21.4	39.4	29.9
Viral	10.7	39.6	28.9
Viral	3.4	40.7	28.9
Bacterial	42.0	37.6	30.8
Bacterial	31.1	42.2	32.9
Bacterial	50.0	38.5	32.3
Bacterial	60.4	39.4	34.0
Bacterial	45.7	38.6	31.9
Bacterial	17.3	42.7	31.8



compared to the ones who have a bacterial infection.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \left(\text{CRP} - \overline{\text{CRP}}\right) + 0.70 \cdot \left(\text{Temp} - \overline{\text{Temp}}\right)$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	29.5
Viral	11.1	37.2	27.3
Viral	30.0	36.5	28.8
Viral	21.4	39.4	29.9
Viral	10.7	39.6	28.9
Viral	3.4	40.7	28.9
Bacterial	42.0	37.6	30.8
Bacterial	31.1	42.2	32.9
Bacterial	50.0	38.5	32.3
Bacterial	60.4	39.4	34.0
Bacterial	45.7	38.6	31.9
Bacterial	17.3	42.7	31.8

Most software tools report the centered discriminant scores, which means that we should subtract the corresponding means from the original values before we calculate the scores.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \left(\text{CRP} - \overline{\text{CRP}}\right) + 0.70 \cdot \left(\text{Temp} - \overline{\text{Temp}}\right)$$

Infection	CRP (mg/L)	Temp (C)	Scores	Cent. scores
Viral	40.0	36.0	29.5	-1.1
Viral	11.1	37.2	27.3	-3.3
Viral	30.0	36.5	28.8	-1.8
Viral	21.4	39.4	29.9	-0.7
Viral	10.7	39.6	28.9	-1.7
Viral	3.4	40.7	28.9	-1.7
Bacterial	42.0	37.6	30.8	0.2
Bacterial	31.1	42.2	32.9	2.3
Bacterial	50.0	38.5	32.3	1.7
Bacterial	60.4	39.4	34.0	3.5
Bacterial	45.7	38.6	31.9	1.3
Bacterial	17.3	42.7	31.8	1.2

By using this equation, the following centered scores were calculated.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \left(\text{CRP} - \overline{\text{CRP}}\right) + 0.70 \cdot \left(\text{Temp} - \overline{\text{Temp}}\right)$$

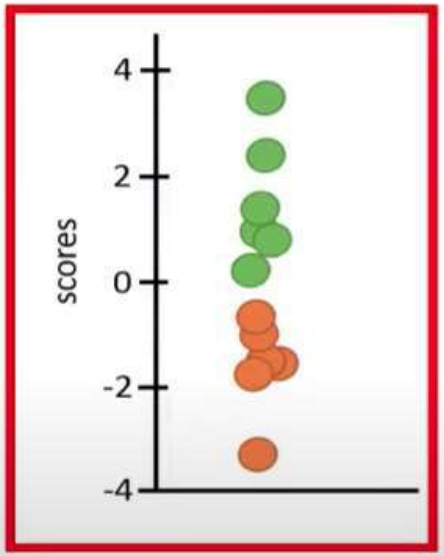
Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	29.5
Viral	11.1	37.2	27.3
Viral	30.0	36.5	28.8
Viral	21.4	39.4	29.9
Viral	10.7	39.6	28.9
Viral	3.4	40.7	28.9
Bacterial	42.0	37.6	30.8
Bacterial	31.1	42.2	32.9
Bacterial	50.0	38.5	32.3
Bacterial	60.4	39.4	34.0
Bacterial	45.7	38.6	31.9
Bacterial	17.3	42.7	31.8

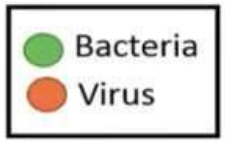
Most software tools report the centered discriminant scores, which means that we should subtract the corresponding means from the original values before we calculate the scores.

Linear Discriminant Analysis (LDA) Algorithm

$$LD = 0.11 \cdot \left(\text{CRP} - \overline{\text{CRP}}\right) + 0.70 \cdot \left(\text{Temp} - \overline{\text{Temp}}\right)$$

Infection	CRP (mg/L)	Temp (C)	Scores	Cent. scores
Viral	40.0	36.0	29.5	-1.1
Viral	11.1	37.2	27.3	-3.3
Viral	30.0	36.5	28.8	-1.8
Viral	21.4	39.4	29.9	-0.7
Viral	10.7	39.6	28.9	-1.7
Viral	3.4	40.7	28.9	-1.7
Bacterial	42.0	37.6	30.8	0.2
Bacterial	31.1	42.2	32.9	2.3
Bacterial	50.0	38.5	32.3	1.7
Bacterial	60.4	39.4	34.0	3.5
Bacterial	45.7	38.6	31.9	1.3
Bacterial	17.3	42.7	31.8	1.2





If we plot the centered scores, we see that the scores centre around zero since the mean of these scores is equal to zero.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)
Viral	0.5	-1.4
Viral	-1.1	-0.9
Viral	0.0	-1.2
Viral	-0.5	0.2
Viral	-1.1	0.3
Viral	-1.5	0.8
Bacterial	0.7	-0.7
Bacterial	0.0	1.5
Bacterial	1.1	-0.3
Bacterial	1.7	0.2
Bacterial	0.9	-0.2
Bacterial	-0.7	1.7
var	**1**	**1**

$$Z_i = \frac{X_i - \overline{X}}{\text{SD}}$$

Remember that we can standardize a variable by the following formula, where we subtract the mean from the values of that variable and divide by its standard deviation.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)
Viral	0.5	-1.4
Viral	-1.1	-0.9
Viral	0.0	-1.2
Viral	-0.5	0.2
Viral	-1.1	0.3
Viral	-1.5	0.8
Bacterial	0.7	-0.7
Bacterial	0.0	1.5
Bacterial	1.1	-0.3
Bacterial	1.7	0.2
Bacterial	0.9	-0.2
Bacterial	-0.7	1.7
var	**1**	**1**

$$Z_i = \frac{40.0 - 30.3}{17.8}$$

For example, this first standardized value has been calculated by subtracting the mean of the variable CRP, 30.3, from the CRP value of the first person, which is 40.0, and divide by the standard deviation, which is 17.8 for the CRP variable.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)
Viral	0.5	-1.4
Viral	-1.1	-0.9
Viral	0.0	-1.2
Viral	-0.5	0.2
Viral	-1.1	0.3
Viral	-1.5	0.8
Bacterial	0.7	-0.7
Bacterial	0.0	1.5
Bacterial	1.1	-0.3
Bacterial	1.7	0.2
Bacterial	0.9	-0.2
Bacterial	-0.7	1.7
var	**1**	**1**

These weights, from the LDA, were also extracted from the first eigenvector. However, this eigenvector is not an eigenvector of the covariance matrix. Instead, it is an eigenvector of another type of matrix that we will discuss later.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**

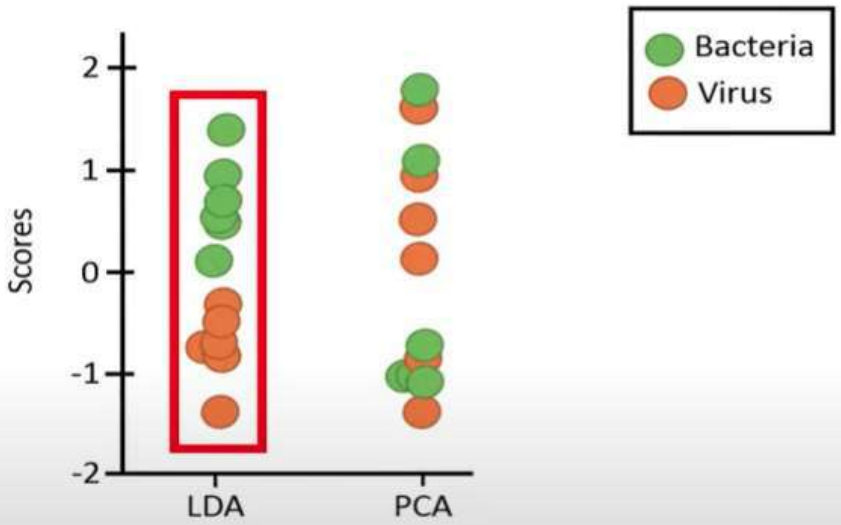
but only 0.71 for the scores based on the first discriminant function. This is no surprise since PCA aims to maximize the variance of the first principal component, whereas LDA instead aims to maximize the separation of the two groups.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**



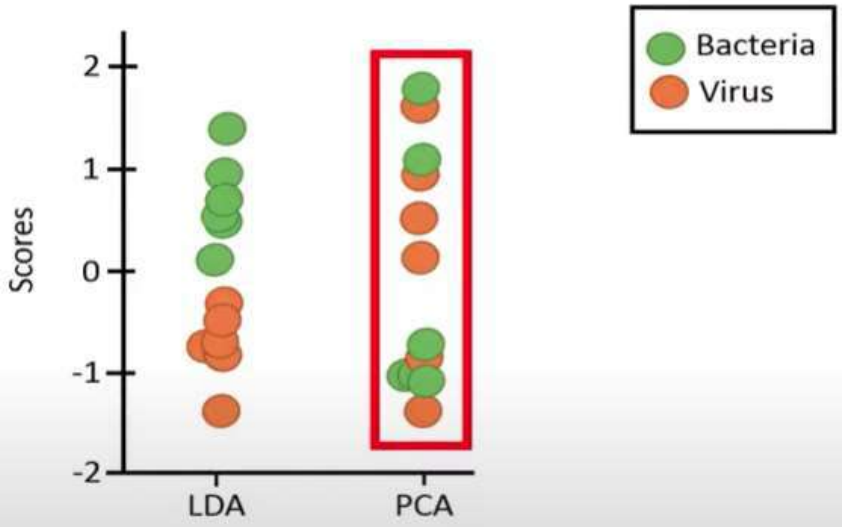
If we plot the scores computed by LDA, we see that the scores for the bacteria group are completely separated from the scores of the virus group.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**



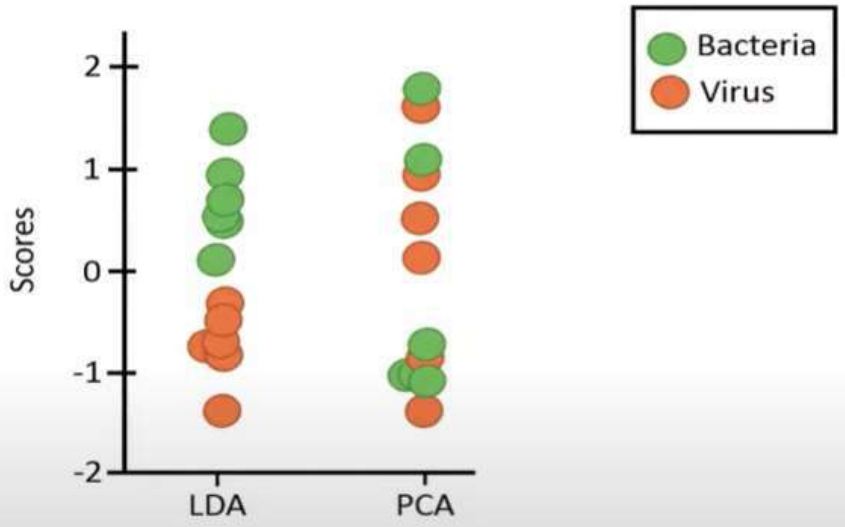
In comparison, the scores computed by the PCA show no clear separation between the groups. Although PCA has captured more variation in the combined variable, it does not separate the groups very well.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**



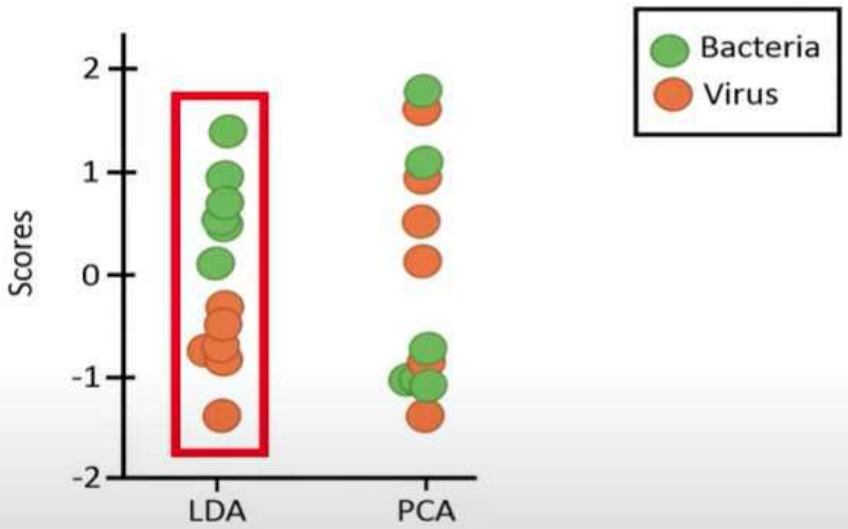
Note that, when we run PCA, we will not use this grouping variable at all, whereas in LDA, we must provide this grouping variable since the method aims to find the maximal separation between these two groups.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**



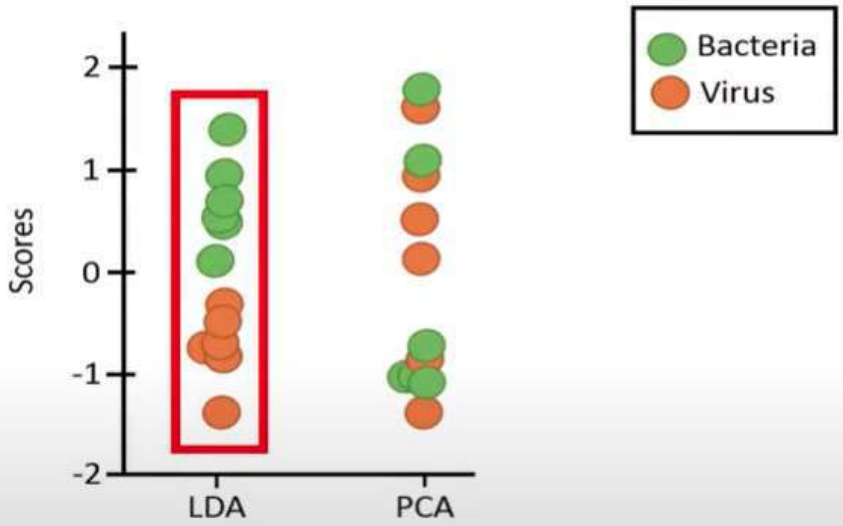
Since PCA will only be based on these two variables, without any information about the groups, it will not be as effective as LDA in separating the groups.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**



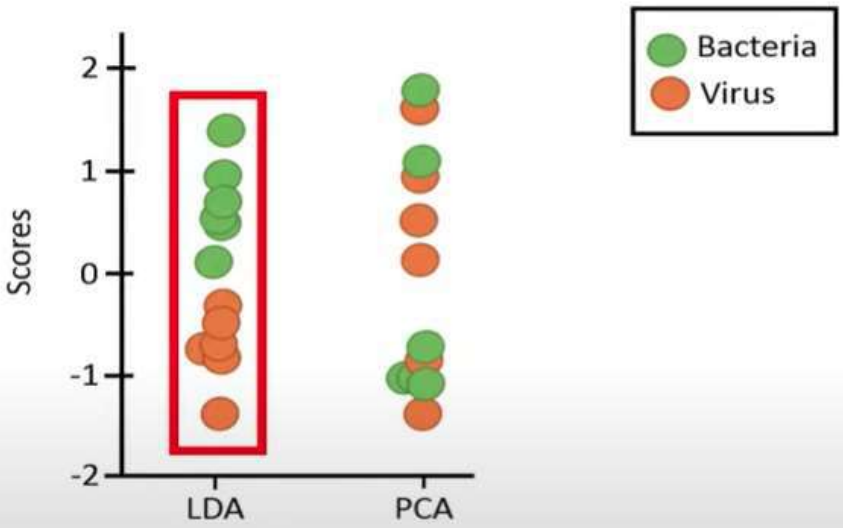
PCA is sometimes referred to as an unsupervised method because it has no information if a data point belongs to a certain group,

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Scores LDA	Scores PCA
Viral	0.5	-1.4	-0.5	-1.4
Viral	-1.1	-0.9	-1.4	0.1
Viral	0.0	-1.2	-0.8	-0.8
Viral	-0.5	0.2	-0.3	0.5
Viral	-1.1	0.3	-0.7	1.0
Viral	-1.5	0.8	-0.7	1.6
Bacterial	0.7	-0.7	0.1	-0.9
Bacterial	0.0	1.5	1.0	1.0
Bacterial	1.1	-0.3	0.7	-1.0
Bacterial	1.7	0.2	1.4	-1.1
Bacterial	0.9	-0.2	0.6	-0.8
Bacterial	-0.7	1.7	0.5	1.8
var	**1**	**1**	**0.71**	**1.29**



whereas LDA is referred to as a supervised method because it has the information about which group a data point belongs to in order to find the best separation between the groups.

Linear Discriminant Analysis (LDA) Algorithm

$$PC1 = -0.71\ zCRP + 0.71 \cdot zTemp$$

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

We will now focus on how we can calculate the weights for the LDA. Remember that the weights for the PCA were extracted from the first eigenvector of the covariance matrix. These weights can be rescaled to loadings.

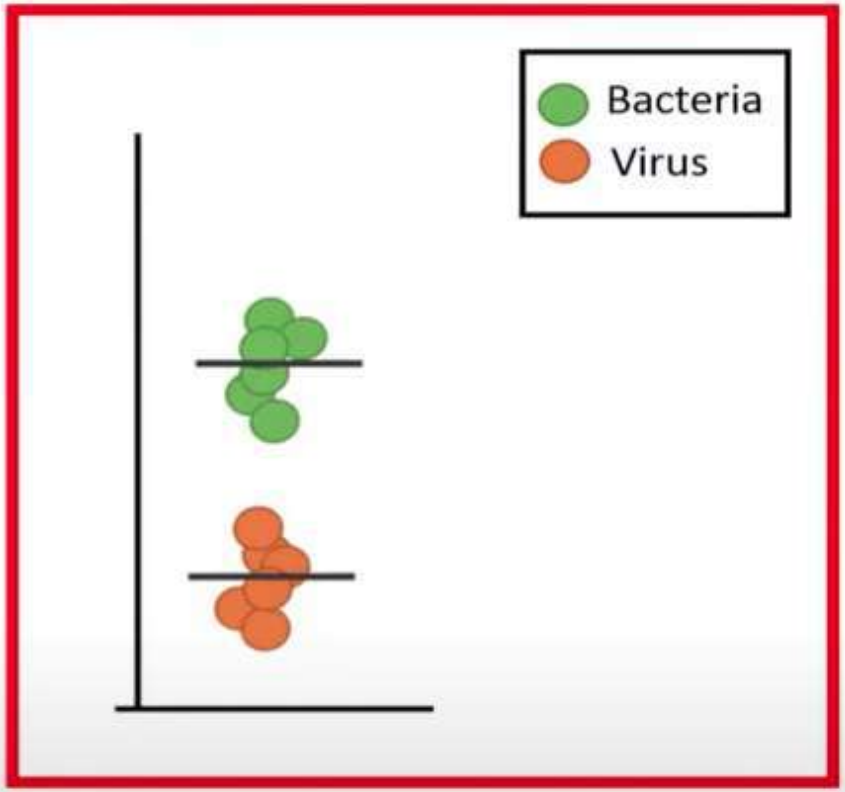
Linear Discriminant Analysis (LDA) Algorithm

$$PC1 = -0.71 \cdot zCRP + 0.71 \cdot zTemp$$

$$LD1 = 0.79 \cdot zCRP + 0.61 \cdot zTemp$$

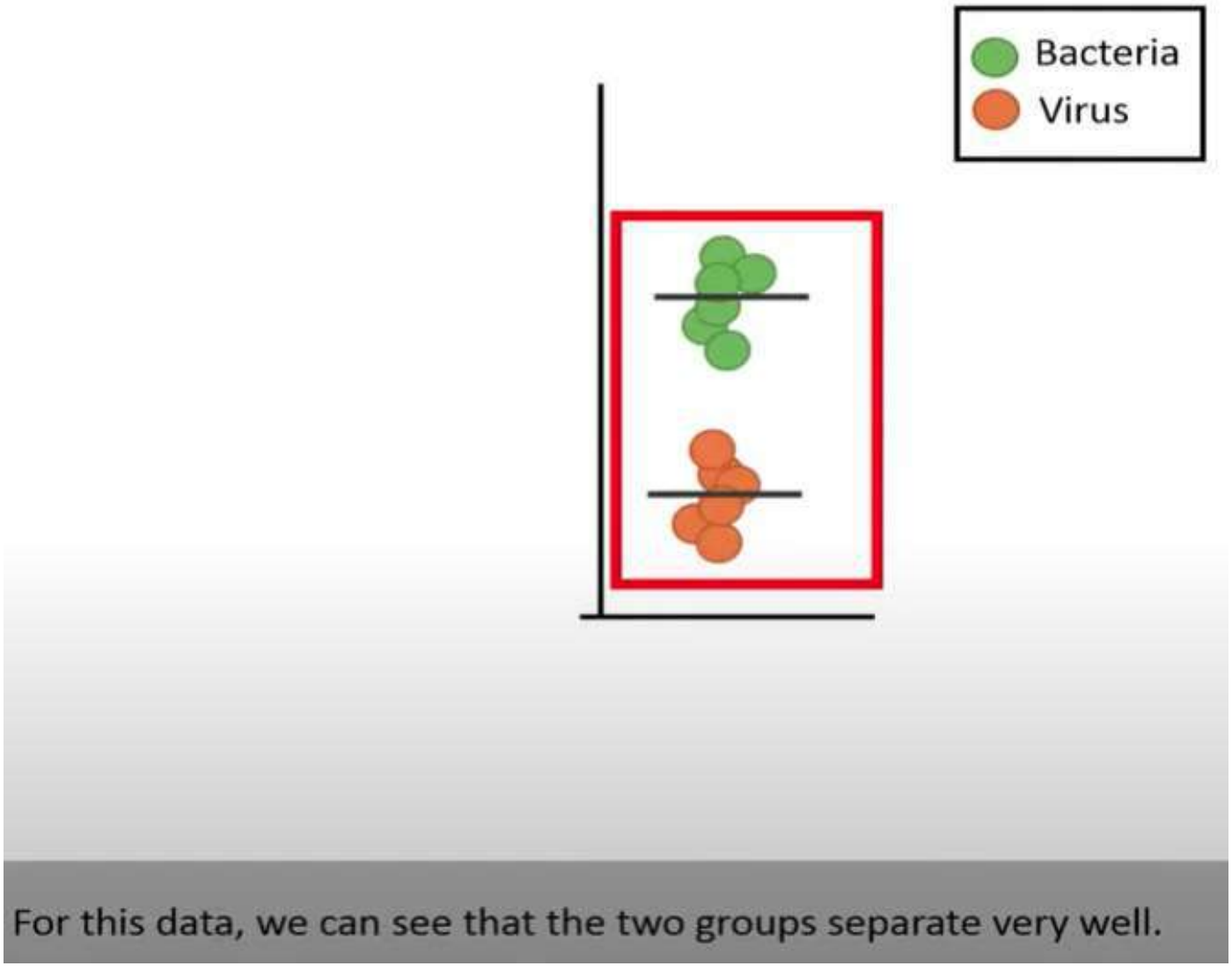
The weights in LDA are also extracted from the first eigenvector, but an eigenvector of a different matrix. These weights are also usually rescaled.

Linear Discriminant Analysis (LDA) Algorithm

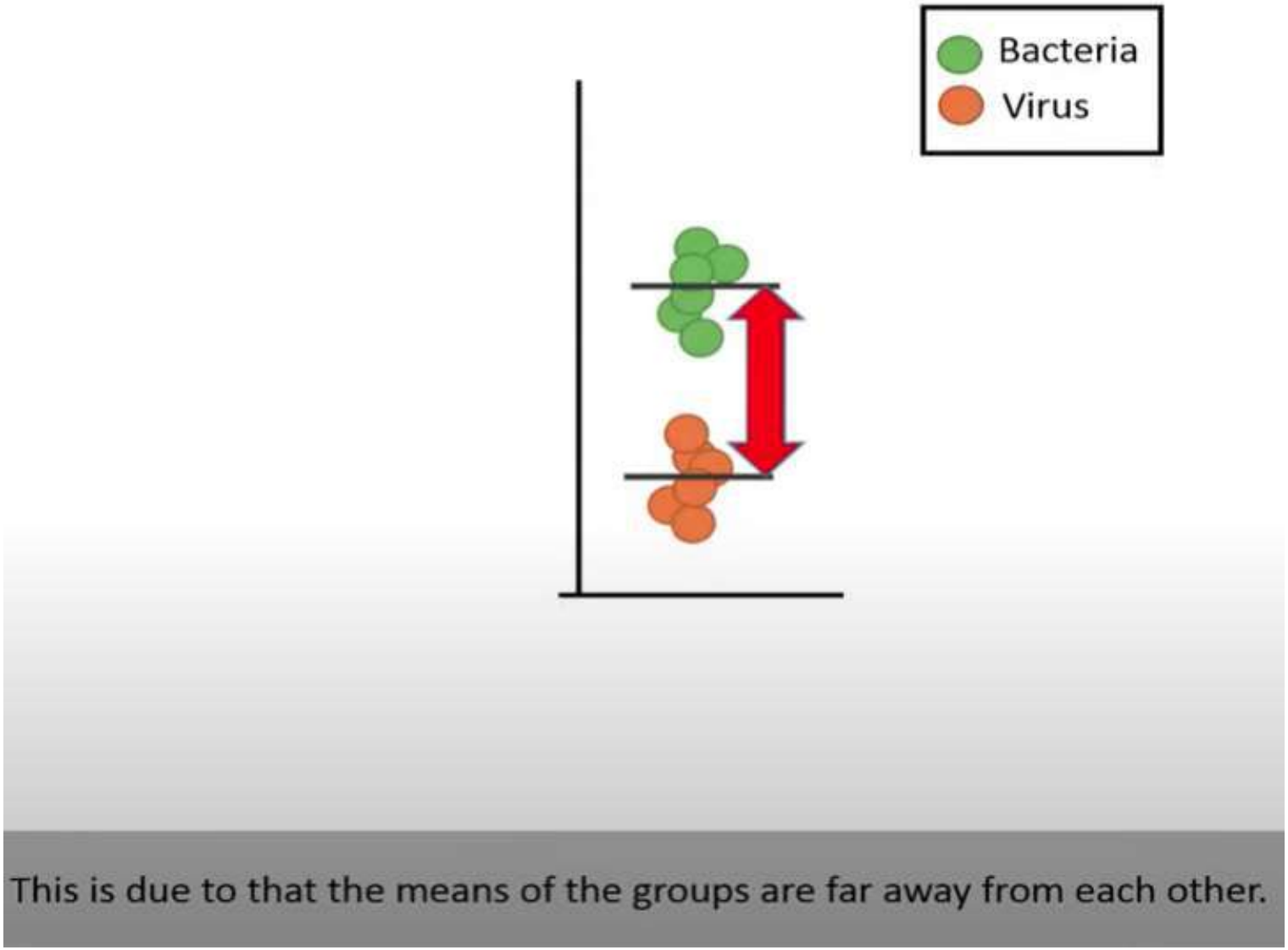


Let's first have a look at what determines how well two groups are separated.

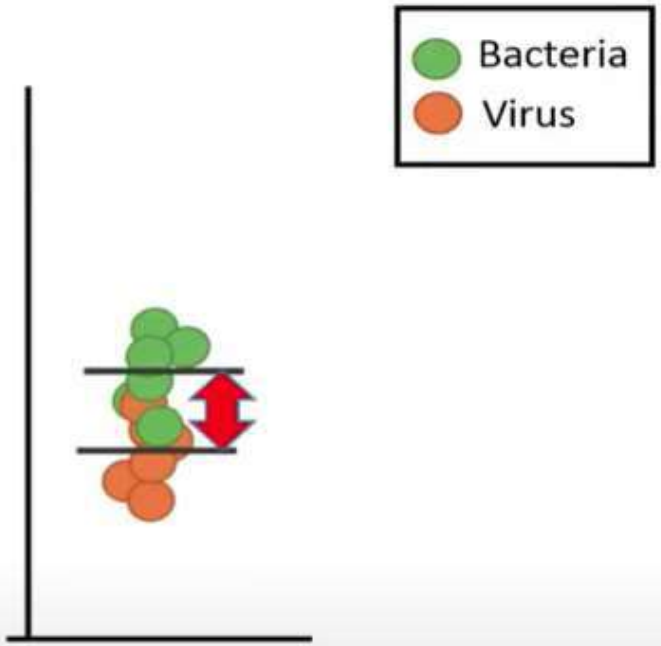
Linear Discriminant Analysis (LDA) Algorithm



Linear Discriminant Analysis (LDA) Algorithm

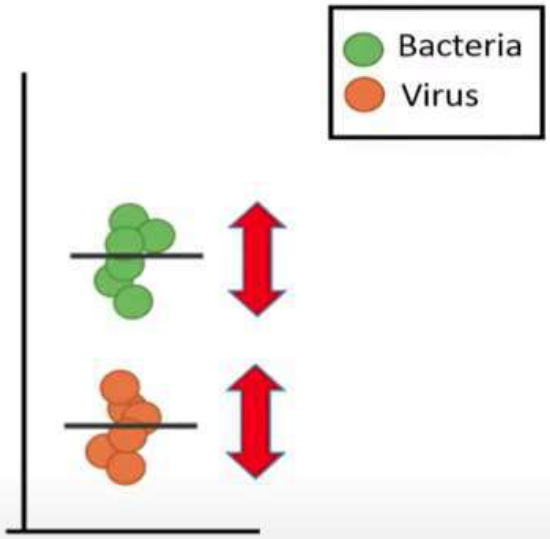


Linear Discriminant Analysis (LDA) Algorithm



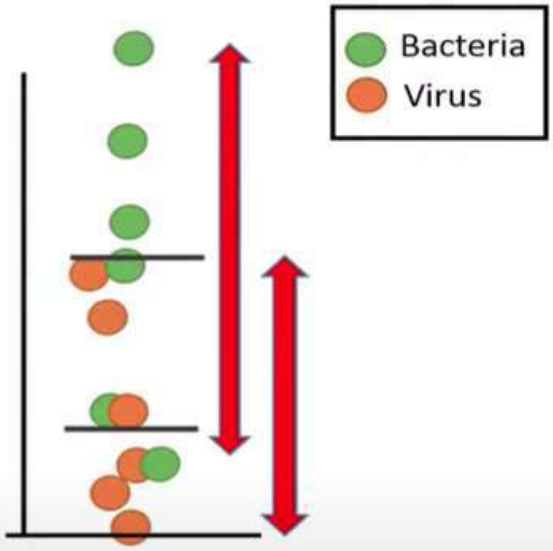
If the means are closer to each other, we can no longer see a clear separation between the two groups.

Linear Discriminant Analysis (LDA) Algorithm



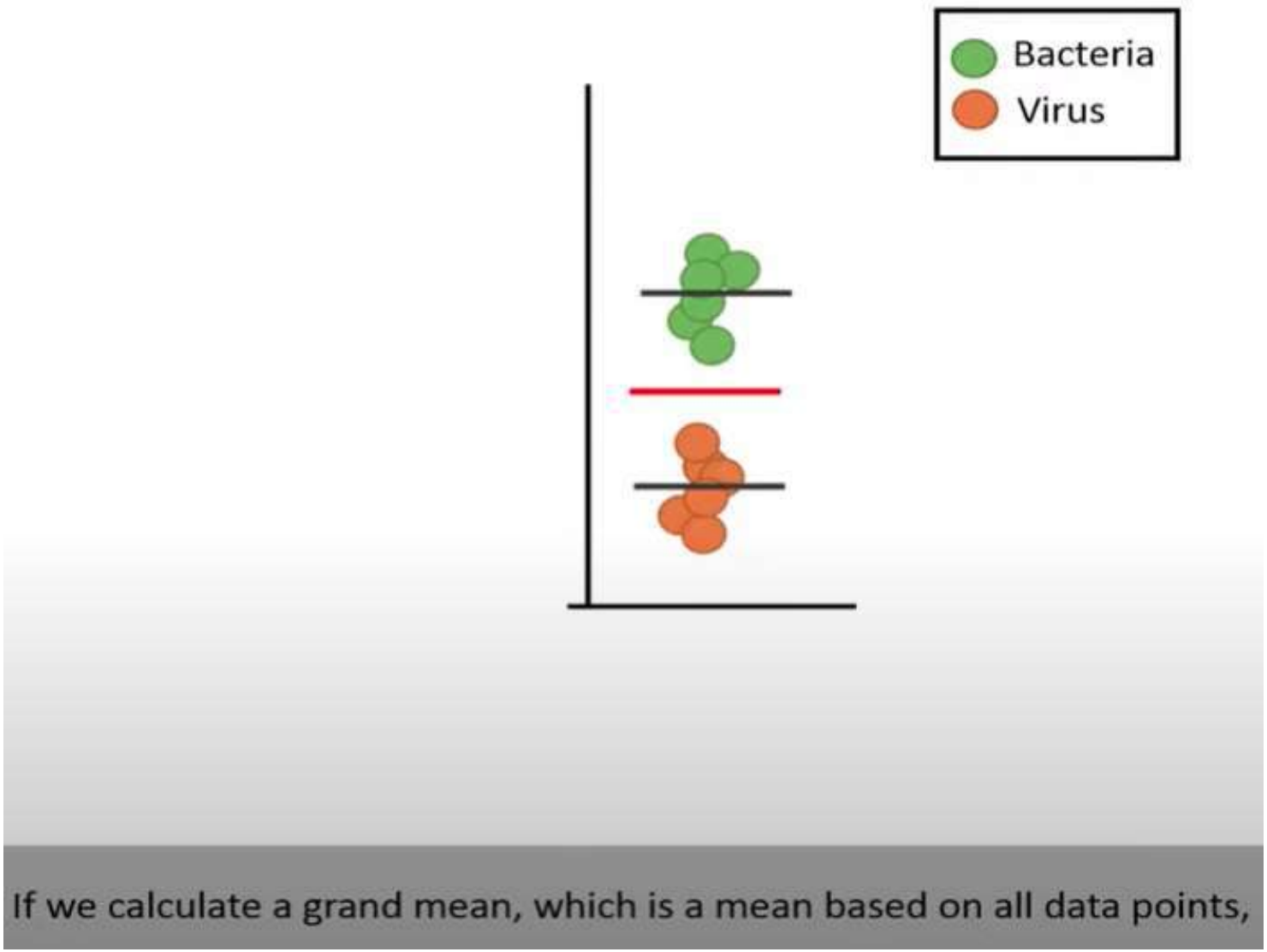
The second reason why these two groups show a good separation is because the spread within the groups is small. The data points are close to the mean values.

Linear Discriminant Analysis (LDA) Algorithm

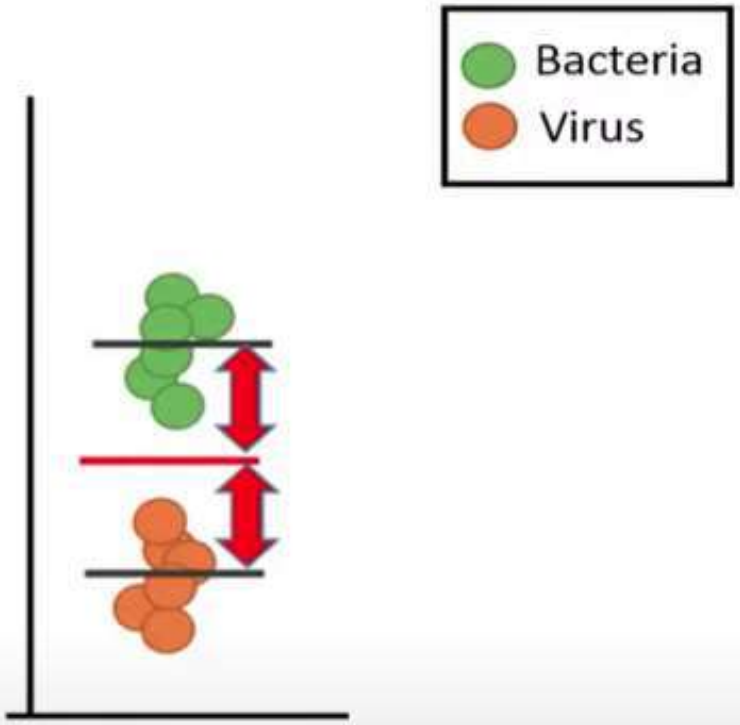


However, if there is a much larger spread of the observations around the means, then there is no longer a clear separation between the groups even though the difference in their means stay the same.

Linear Discriminant Analysis (LDA) Algorithm

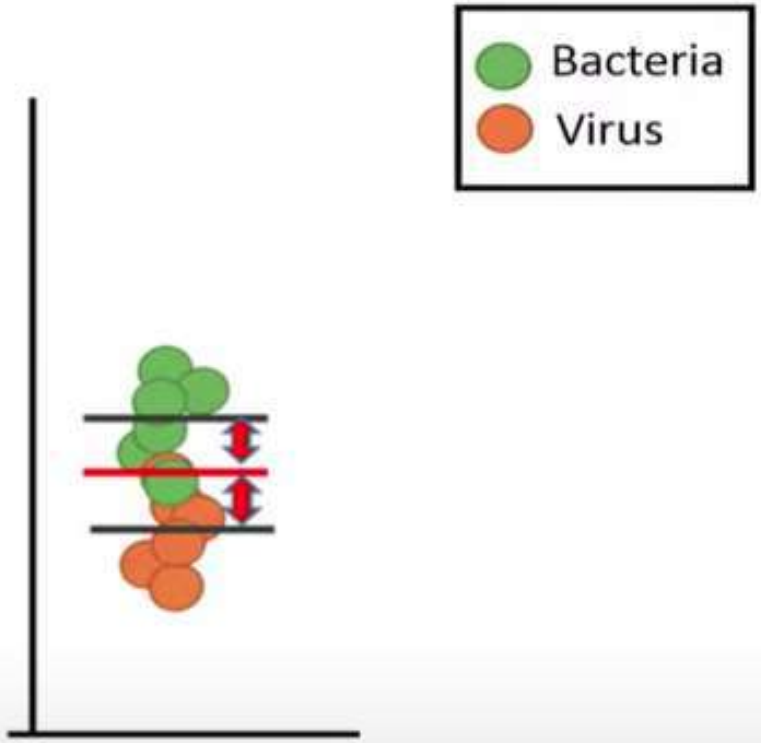


Linear Discriminant Analysis (LDA) Algorithm



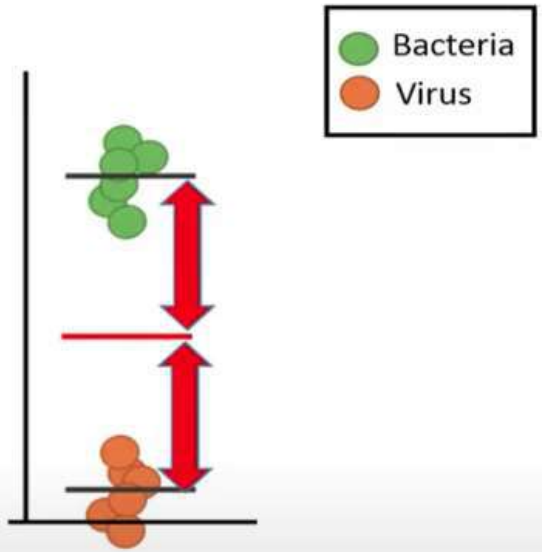
we can think of these distances as how much the group means varies from the grand mean.

Linear Discriminant Analysis (LDA) Algorithm



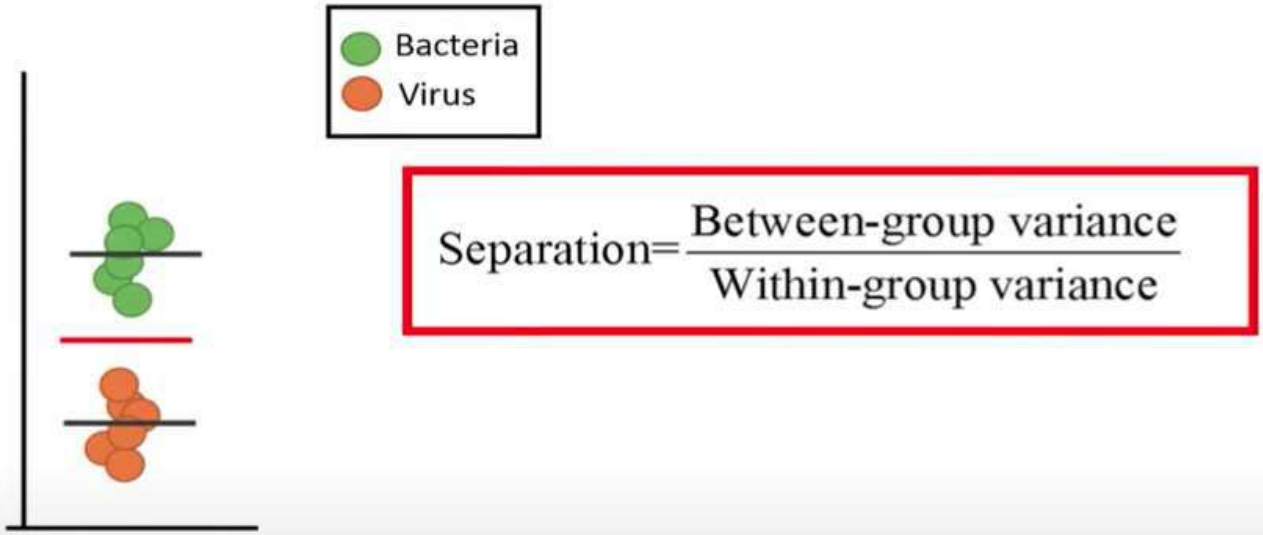
If the means are close, the variation of the group means around the grand mean is small,

Linear Discriminant Analysis (LDA) Algorithm



and if the two means are far away from each other, there will be a large variation of the group means around the grand mean.

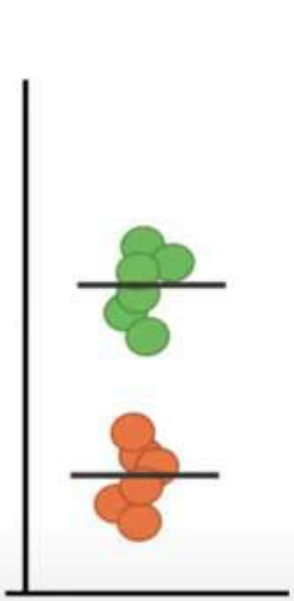
Linear Discriminant Analysis (LDA) Algorithm



$$\text{Separation} = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

We can conclude that the separation between the groups depends on the variance between the groups, which is here called between-group variance, and the variance within the groups, the within-group variance.

Linear Discriminant Analysis (LDA) Algorithm

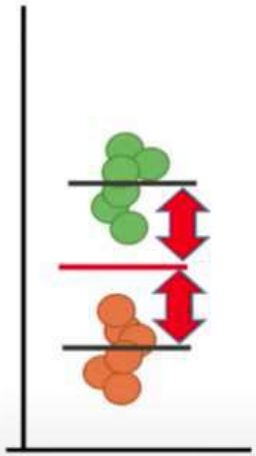


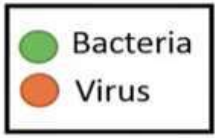
Bacteria
Virus

$$\text{Separation} = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

Thus, the ratio of these measures should be as large as possible to get a clear separation between the groups.

Linear Discriminant Analysis (LDA) Algorithm

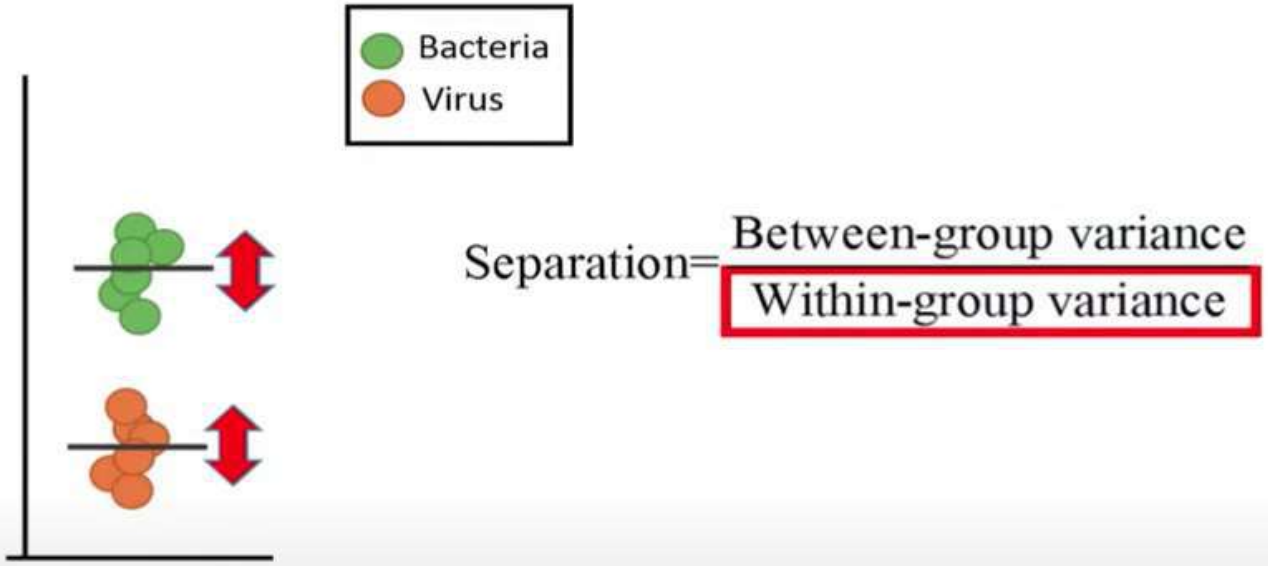




$$\text{Separation} = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

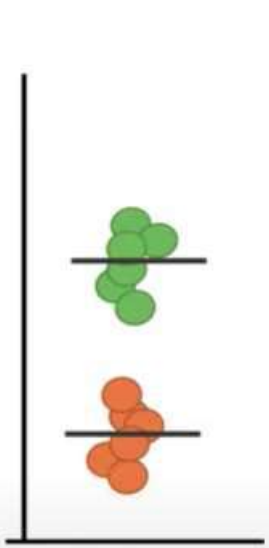
To get a good separation between the groups, we like that the two group means are far away from each other, which means that the between-group variance should be large,

Linear Discriminant Analysis (LDA) Algorithm



and that the variation within the groups should be small, which will result in a low value of the within-group variance.

Linear Discriminant Analysis (LDA) Algorithm

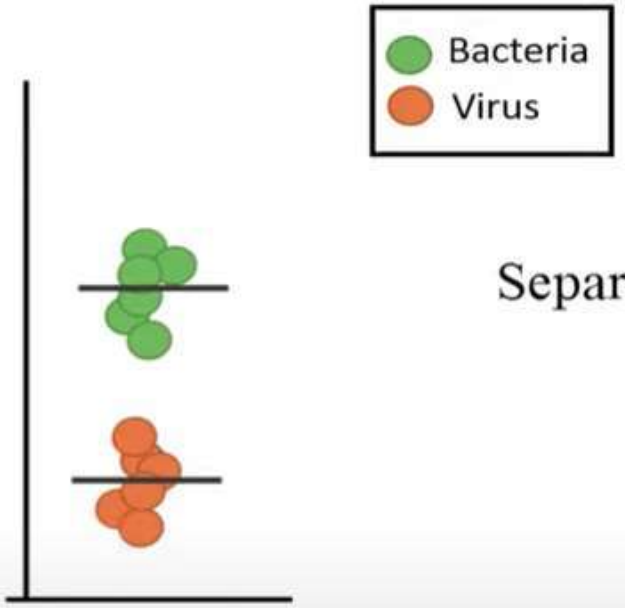


Bacteria
Virus

$$\text{Separation} = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

LDA combines variables based on this ratio, where the aim is to transform the data so that the between-group variance is increased and the within-group variance is reduced.

Linear Discriminant Analysis (LDA) Algorithm



$$\text{Separation} = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

$$\boxed{S} = W^{-1}B$$

Matrix S is our target matrix. We will compute the eigenvectors of this matrix in order to get our weights.

Linear Discriminant Analysis (LDA) Algorithm

$$S = W^{-1}B$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

$$\text{cov}_{\text{virus}} = \begin{bmatrix} 188 & -21 \\ -21 & 4 \end{bmatrix}$$

To calculate the pooled within-group covariance matrix, we first calculate the covariance matrix for the virus group,

Linear Discriminant Analysis (LDA) Algorithm

$$S = W^{-1}B$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

$$\text{cov}_{\text{virus}} = \begin{bmatrix} 188 & -21 \\ -21 & 4 \end{bmatrix}$$

$$\text{cov}_{\text{Bacteria}} = \begin{bmatrix} 228 & -24 \\ -24 & 4 \end{bmatrix}$$

and then for the bacteria group. Note that the values in these two matrices have been rounded.

Linear Discriminant Analysis (LDA) Algorithm

$$S = W^{-1}B$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

$$\text{cov}_{\text{virus}} = \begin{bmatrix} 188 & -21 \\ -21 & 4 \end{bmatrix}$$

$$\text{cov}_{\text{Bacteria}} = \begin{bmatrix} 228 & -24 \\ -24 & 4 \end{bmatrix}$$

$$W = \frac{\begin{bmatrix} 188 & -21 \\ -21 & 4 \end{bmatrix} + \begin{bmatrix} 228 & -24 \\ -24 & 4 \end{bmatrix}}{2} = \begin{bmatrix} 208.1 & -22.5 \\ -22.5 & 4.1 \end{bmatrix}$$

$$W = \frac{(n_1 - 1)\text{cov}(A) + (n_2 - 1)\text{cov}(B)}{n_1 + n_2 - 2}$$

If the two groups have an unequal sample size, we should use the following equation to compute the weighted average of the two covariance matrices.

Linear Discriminant Analysis (LDA) Algorithm

$$S = W^{-1}B$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

$$T = B + W$$

$$B = T - W$$

$$W = \begin{bmatrix} 208.1 & -22.5 \\ -22.5 & 4.1 \end{bmatrix}$$

$$T = \begin{bmatrix} 317.1 & -11.0 \\ -11.0 & 4.4 \end{bmatrix}$$

$$B = \begin{bmatrix} 317.1 & -11.0 \\ -11.0 & 4.4 \end{bmatrix} - \begin{bmatrix} 208.1 & -22.5 \\ -22.5 & 4.1 \end{bmatrix} = \begin{bmatrix} 108.9 & 11.5 \\ 11.5 & 0.32 \end{bmatrix}$$

by subtracting the pooled within-group covariance matrix,

Linear Discriminant Analysis (LDA) Algorithm

$$S = W^{-1}B$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

$$W = \begin{bmatrix} 208.1 & -22.5 \\ -22.5 & 4.1 \end{bmatrix}$$

$$W^{-1} = \begin{bmatrix} 0.012 & 0.066 \\ 0.066 & 0.609 \end{bmatrix}$$

$$B = \begin{bmatrix} 108.9 & 11.5 \\ 11.5 & 0.32 \end{bmatrix}$$

$$S = \begin{bmatrix} 0.012 & 0.066 \\ 0.066 & 0.609 \end{bmatrix} \cdot \begin{bmatrix} 108.9 & 11.5 \\ 11.5 & 0.32 \end{bmatrix} = \begin{bmatrix} 2.05 & 0.16 \\ 14.15 & 0.96 \end{bmatrix}$$

We calculate matrix S by

Linear Discriminant Analysis (LDA) Algorithm

$$S = W^{-1}B$$

Infection	CRP (mg/L)	Temp (C)
Viral	40.0	36.0
Viral	11.1	37.2
Viral	30.0	36.5
Viral	21.4	39.4
Viral	10.7	39.6
Viral	3.4	40.7
Bacterial	42.0	37.6
Bacterial	31.1	42.2
Bacterial	50.0	38.5
Bacterial	60.4	39.4
Bacterial	45.7	38.6
Bacterial	17.3	42.7

$$S = \begin{bmatrix} 2.05 & 0.16 \\ 14.15 & 0.96 \end{bmatrix}$$

$$\text{Eigenvectors} = \begin{bmatrix} 0.150 & -0.074 \\ 0.989 & 0.997 \end{bmatrix}$$

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

We then combine the two variables, the CRP and the body temperature, by using the weights that are provided by the first eigenvector.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	41.6
Viral	11.1	37.2	38.4
Viral	30.0	36.5	40.6
Viral	21.4	39.4	42.2
Viral	10.7	39.6	40.8
Viral	3.4	40.7	40.8
Bacterial	42.0	37.6	43.5
Bacterial	31.1	42.2	46.4
Bacterial	50.0	38.5	45.5
Bacterial	60.4	39.4	48.0
Bacterial	45.7	38.6	45.0
Bacterial	17.3	42.7	44.8

In LDA, the weights are usually rescaled so that the pooled group variance of the scores is equal to one.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	41.6
Viral	11.1	37.2	38.4
Viral	30.0	36.5	40.6
Viral	21.4	39.4	42.2
Viral	10.7	39.6	40.8
Viral	3.4	40.7	40.8
Bacterial	42.0	37.6	43.5
Bacterial	31.1	42.2	46.4
Bacterial	50.0	38.5	45.5
Bacterial	60.4	39.4	48.0
Bacterial	45.7	38.6	45.0
Bacterial	17.3	42.7	44.8

$$\text{var(scores)}_{\text{Viral}} = 1.60$$

$$\boxed{\text{var(scores)}_{\text{pooled}} = 1.989}$$

$$\text{var(scores)}_{\text{Bacterial}} = 2.37$$

The average of these variances is 1.989, which is the pooled variance since the two groups have an equal sample size. If the two groups have an unequal sample size, we need to calculate a weighted mean instead.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	41.6
Viral	11.1	37.2	38.4
Viral	30.0	36.5	40.6
Viral	21.4	39.4	42.2
Viral	10.7	39.6	40.8
Viral	3.4	40.7	40.8
Bacterial	42.0	37.6	43.5
Bacterial	31.1	42.2	46.4
Bacterial	50.0	38.5	45.5
Bacterial	60.4	39.4	48.0
Bacterial	45.7	38.6	45.0
Bacterial	17.3	42.7	44.8

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

$$\sqrt{1.989}$$

$$\text{var(scores)}_{\text{pooled}} = 1.989$$

If we now divide the weights by the square root of the pooled variance, which corresponds to the pooled standard deviation,

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

Infection	CRP (mg/L)	Temp (C)	Scores
Viral	40.0	36.0	41.6
Viral	11.1	37.2	38.4
Viral	30.0	36.5	40.6
Viral	21.4	39.4	42.2
Viral	10.7	39.6	40.8
Viral	3.4	40.7	40.8
Bacterial	42.0	37.6	43.5
Bacterial	31.1	42.2	46.4
Bacterial	50.0	38.5	45.5
Bacterial	60.4	39.4	48.0
Bacterial	45.7	38.6	45.0
Bacterial	17.3	42.7	44.8

$$LD1 = 0.11 \cdot CRP + 0.70 \cdot Temp$$

we would get the following types of weights, or loadings, that are usually presented by most statistical software tools.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

Infection	CRP (mg/L)	Temp (C)	Scores	Scores
Viral	40.0	36.0	41.6	29.5
Viral	11.1	37.2	38.4	27.3
Viral	30.0	36.5	40.6	28.8
Viral	21.4	39.4	42.2	29.9
Viral	10.7	39.6	40.8	28.9
Viral	3.4	40.7	40.8	28.9
Bacterial	42.0	37.6	43.5	30.8
Bacterial	31.1	42.2	46.4	32.9
Bacterial	50.0	38.5	45.5	32.3
Bacterial	60.4	39.4	48.0	34.0
Bacterial	45.7	38.6	45.0	31.9
Bacterial	17.3	42.7	44.8	31.8

$$LD1 = 0.11 \cdot CRP + 0.70 \cdot Temp$$

By using this discriminant function with these rescaled weights, we can compute the following scores. These scores are the same as the ones in the first example in this video.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.150 \cdot CRP + 0.989 \cdot Temp$$

Infection	CRP (mg/L)	Temp (C)	Scores	Scores
Viral	40.0	36.0	41.6	29.5
Viral	11.1	37.2	38.4	27.3
Viral	30.0	36.5	40.6	28.8
Viral	21.4	39.4	42.2	29.9
Viral	10.7	39.6	40.8	28.9
Viral	3.4	40.7	40.8	28.9
Bacterial	42.0	37.6	43.5	30.8
Bacterial	31.1	42.2	46.4	32.9
Bacterial	50.0	38.5	45.5	32.3
Bacterial	60.4	39.4	48.0	34.0
Bacterial	45.7	38.6	45.0	31.9
Bacterial	17.3	42.7	44.8	31.8

$$\text{var(scores)}_{\text{Viral}} = 0.81$$

$$\text{var(scores)}_{\text{pooled}} = 1$$

$$\text{var(scores)}_{\text{Bacterial}} = 1.19$$

$$LD1 = 0.11 \cdot CRP + 0.70 \cdot Temp$$

and pool those, we see that this pooled variance is now equal to one.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Scores	Scores	Cent. scores
Viral	40.0	36.0	41.6	29.5	-1.1
Viral	11.1	37.2	38.4	27.3	-3.3
Viral	30.0	36.5	40.6	28.8	-1.8
Viral	21.4	39.4	42.2	29.9	-0.7
Viral	10.7	39.6	40.8	28.9	-1.7
Viral	3.4	40.7	40.8	28.9	-1.7
Bacterial	42.0	37.6	43.5	30.8	0.2
Bacterial	31.1	42.2	46.4	32.9	2.3
Bacterial	50.0	38.5	45.5	32.3	1.7
Bacterial	60.4	39.4	48.0	34.0	3.5
Bacterial	45.7	38.6	45.0	31.9	1.3
Bacterial	17.3	42.7	44.8	31.8	1.2

$$LD = 0.11 \cdot \left(\mathrm{CRP} - \overline{\mathrm{CRP}}\right) + 0.70 \cdot \left(\mathrm{Temp} - \overline{\mathrm{Temp}}\right)$$

and the means of the two variables. By doing this, the score is calculated to about minus 1.1.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Scores	Scores	Cent. scores
Viral	40.0	36.0	41.6	29.5	-1.1
Viral	11.1	37.2	38.4	27.3	-3.3
Viral	30.0	36.5	40.6	28.8	-1.8
Viral	21.4	39.4	42.2	29.9	-0.7
Viral	10.7	39.6	40.8	28.9	-1.7
Viral	3.4	40.7	40.8	28.9	-1.7
Bacterial	42.0	37.6	43.5	30.8	0.2
Bacterial	31.1	42.2	46.4	32.9	2.3
Bacterial	50.0	38.5	45.5	32.3	1.7
Bacterial	60.4	39.4	48.0	34.0	3.5
Bacterial	45.7	38.6	45.0	31.9	1.3
Bacterial	17.3	42.7	44.8	31.8	1.2

$$LD = 0.11 \cdot \left(\text{CRP} - \overline{\text{CRP}}\right) + 0.70 \cdot \left(\text{Temp} - \overline{\text{Temp}}\right)$$

we plug in the two measurements for this person,

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.9
Viral	30.0	36.5	0.0	-1.3
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

$$Z = \frac{X - \overline{X}}{\sqrt{\text{var}(X)}} \qquad Z = \frac{X - \overline{X}}{\sqrt{\text{var}(X)_{\text{pooled}}}}$$

In contrast, for LDA we divide by the square root of the pooled variance of the groups for that variable.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.9
Viral	30.0	36.5	0.0	-1.3
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

$$Z = \frac{X - \overline{X}}{\sqrt{\text{var}(X)_{\text{pooled}}}}$$

If we use this equation, we will get the following standardized variables.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.9
Viral	30.0	36.5	0.0	-1.2
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

$$Z = \frac{X - \overline{X}}{\sqrt{\text{var}(X)_{\text{pooled}}}}$$

$$Z = \frac{40 - 30.3}{\sqrt{208.1}} = 0.7$$

$$\overline{X} = 30.3$$

$$\text{var}(X)_{\text{Viral}} = 188.3$$

$$\text{var}(X)_{\text{Bacterial}} = 228.0$$

$$\text{var}(X)_{\text{pooled}} = 208.1$$

As an example, we will here calculate the standardized CRP value for person number one.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.5
Viral	30.0	36.5	0.0	-1.3
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

$$Z = \frac{X - \overline{X}}{\sqrt{\text{var}(X)_{\text{pooled}}}}$$

$$Z = \frac{40 - 30.3}{\sqrt{208.1}} = 0.7$$

$$\overline{X} = 30.3$$

$$\text{var}(X)_{\text{Viral}} = 188.3$$

$$\text{var}(X)_{\text{Bacterial}} = 228.0$$

$$\text{var}(X)_{\text{pooled}} = 208.1$$

We see that the calculation results in a standardized value of about 0.7.

Linear Discriminant Analysis (LDA) Algorithm

$$LD1 = 0.73 \cdot zCRP + 0.68 \cdot zTemp$$

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.9
Viral	30.0	36.5	0.0	-1.3
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

We now simply use these values to compute the scores of the first discriminant function.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.9
Viral	30.0	36.5	0.0	-1.3
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

$$LD1 = 0.73 \cdot zCRP + 0.68 \cdot zTemp$$

$$LD1 = 1.53 \cdot zCRP + 1.41 \cdot zTemp$$

and if we rescale the weights in the same way as we have seen previously, we will get the following standardized discriminant function coefficients, which are commonly reported by most statistical software tools.

Linear Discriminant Analysis (LDA) Algorithm

Infection	CRP (mg/L)	Temp (C)	Z CRP	Z Temp
Viral	40.0	36.0	0.7	-1.5
Viral	11.1	37.2	-1.3	-0.9
Viral	30.0	36.5	0.0	-1.3
Viral	21.4	39.4	-0.6	0.2
Viral	10.7	39.6	-1.4	0.3
Viral	3.4	40.7	-1.9	0.8
Bacterial	42.0	37.6	0.8	-0.7
Bacterial	31.1	42.2	0.1	1.6
Bacterial	50.0	38.5	1.4	-0.3
Bacterial	60.4	39.4	2.1	0.2
Bacterial	45.7	38.6	1.1	-0.2
Bacterial	17.3	42.7	-0.9	1.8

$$LD1 = 0.73 \cdot zCRP + 0.68 \cdot zTemp$$

$$LD1 = 1.53 \cdot zCRP + 1.41 \cdot zTemp$$

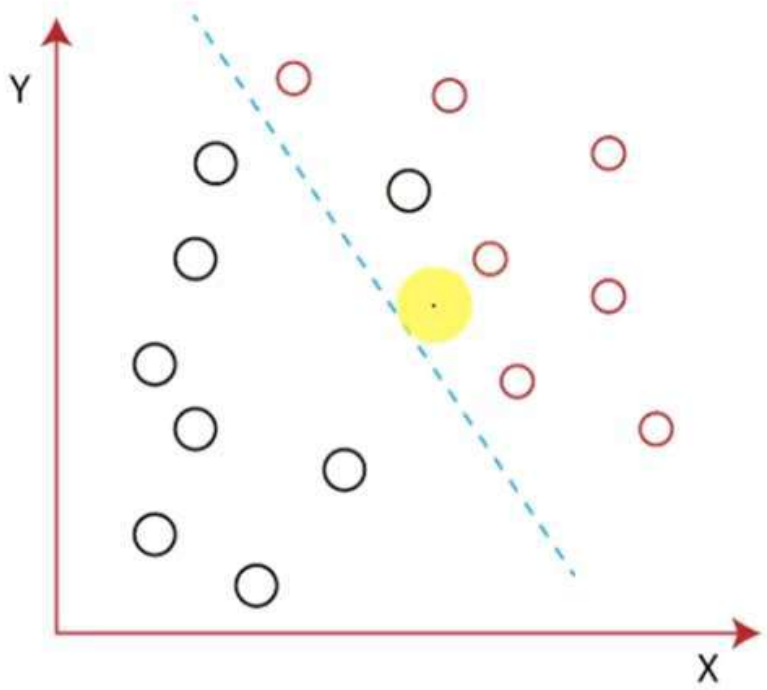
In our example, the values of the two coefficients are quite similar, which means that the variables CRP and body temperature contribute to about the same extent to separate the virus group from the bacteria group.

Linear Discriminant Analysis (LDA) Algorithm

- Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems.
- It is used to project the features in higher dimension space into a lower dimension space.

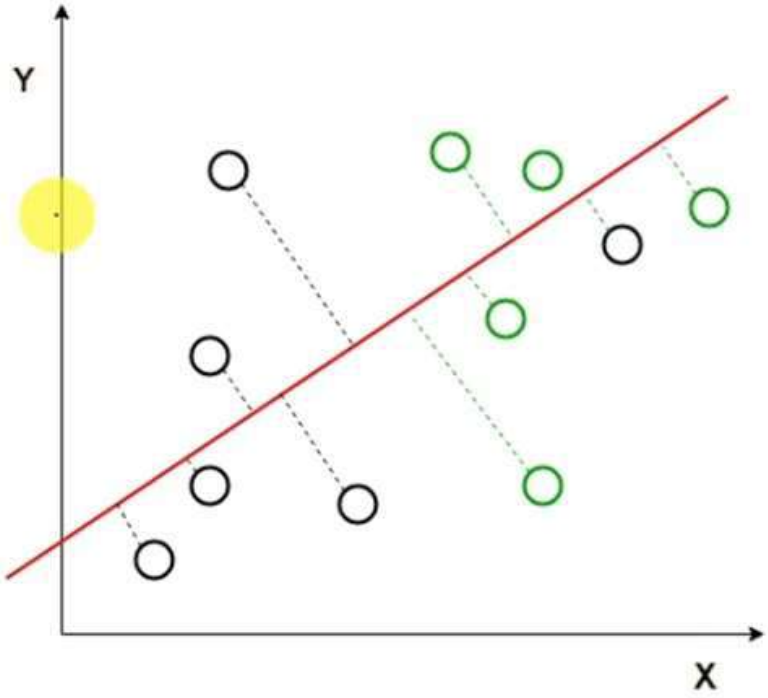
Linear Discriminant Analysis (LDA) Algorithm

- Suppose we have two sets of data points belonging to two different classes that we want to classify.
- When the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of the data points completely.



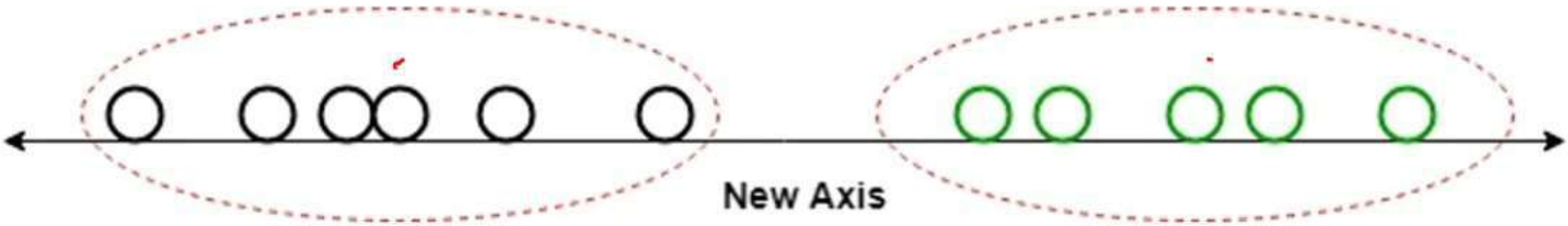
Linear Discriminant Analysis (LDA) Algorithm

- Here, Linear Discriminant Analysis uses both the axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximize the separation of the two categories and hence, reducing the 2D graph into a 1D graph.



Linear Discriminant Analysis (LDA) Algorithm

- Two criteria are used by LDA to create a new axis:
  - Maximize the distance between means of the two classes.
  - Minimize the variation within each class.



Linear Discriminant Analysis (LDA) Algorithm

1. Compute the class means of dependent variable
2. Derive the covariance matrix of the class variable
3. Compute the within class — scatter matrix (S1+S2)
4. Compute the between class scatter matrix
5. Compute the Eigen values and eigen vectors from the within class and between class scatter matrix

$$\mu_1 = \frac{1}{N_1} \sum_{x \in \omega_1} x$$

$$S_1 = \sum_{x \in \omega_1} (x - \mu_1)(x - \mu_1)^T$$

$$S_w = S_1 + S_2$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$S_W^{-1} S_B w = \lambda w$$

Linear Discriminant Analysis (LDA) Algorithm

6. Sort the values of eigen values and select the top k values
7. Find the eigen vectors corresponds to the top k eigen vectors
8. Obtain the LDA by taking the dot product of eigen vectors and original data

$$\left(S_W^{-1} S_B - \lambda I\right)\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = 0$$

Linear Discriminant Analysis (LDA) Algorithm

- Compute the Linear Discriminant projection for the following two dimensional dataset.

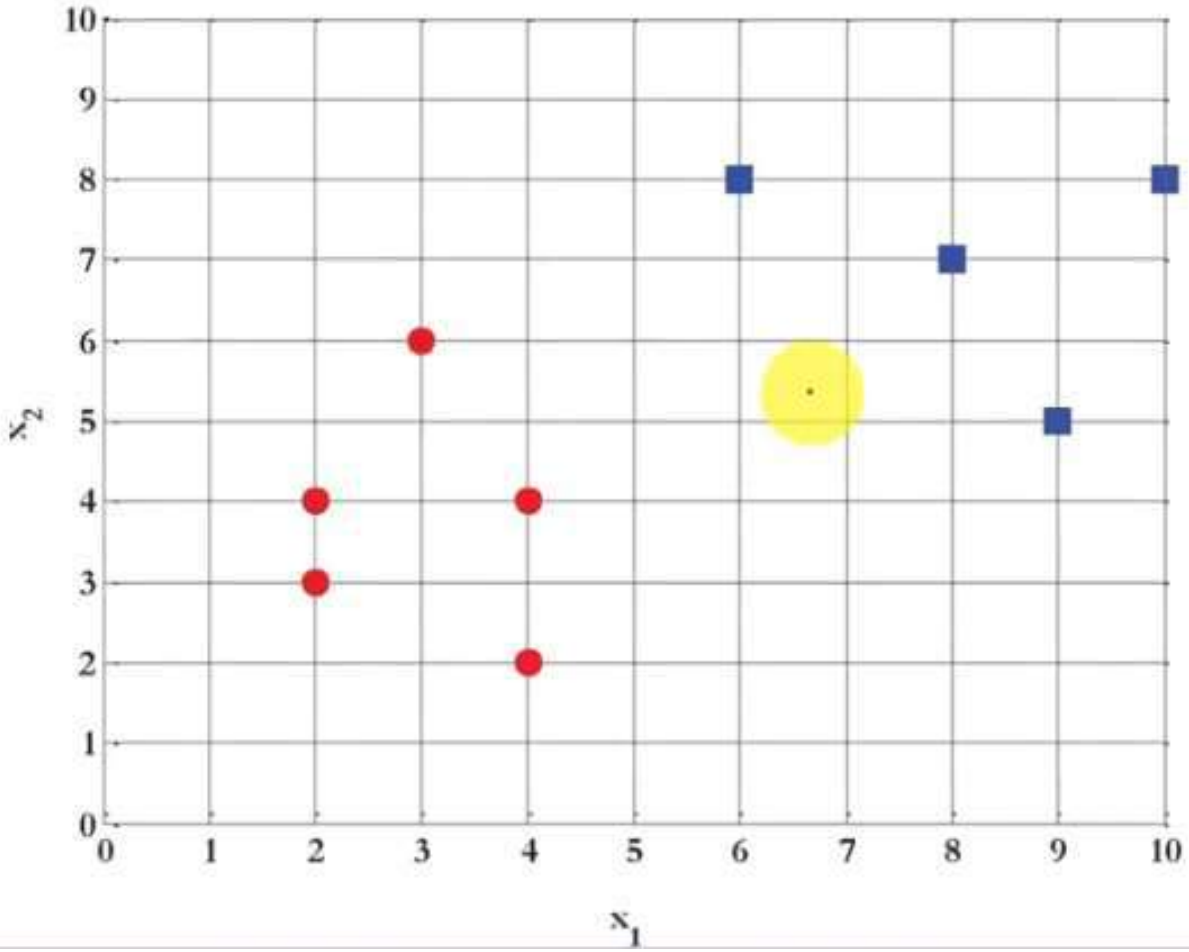
Samples for class ω_1 : $\mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class ω_2 : $\mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

Linear Discriminant Analysis

Samples for class $\omega_1 : \mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class $\omega_2 : \mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$



Linear Discriminant Analysis (LDA) Algorithm

Samples for class $\omega_1 : \mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class $\omega_2 : \mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

The classes mean are :

$$\mu_1 = \frac{1}{N_1}\sum_{x\in\omega_1} x = \frac{1}{5}\left[\begin{pmatrix}4\\2\end{pmatrix}+\begin{pmatrix}2\\4\end{pmatrix}+\begin{pmatrix}2\\3\end{pmatrix}+\begin{pmatrix}3\\6\end{pmatrix}+\begin{pmatrix}4\\4\end{pmatrix}\right] = \begin{pmatrix}3\\3.8\end{pmatrix}$$

$$\mu_2 = \frac{1}{N_2}\sum_{x\in\omega_2} x = \frac{1}{5}\left[\begin{pmatrix}9\\10\end{pmatrix}+\begin{pmatrix}6\\8\end{pmatrix}+\begin{pmatrix}9\\5\end{pmatrix}+\begin{pmatrix}8\\7\end{pmatrix}+\begin{pmatrix}10\\8\end{pmatrix}\right] = \begin{pmatrix}8.4\\7.6\end{pmatrix}$$

Linear Discriminant Analysis (LDA) Algorithm

Samples for class ω_1 : $\mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class ω_2 : $\mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

Covariance matrix of the first class:

$$S_1 = \sum_{x\in\omega_1} \frac{(x-\mu_1)(x-\mu_1)^T}{\text{N-1}} = \left[\begin{pmatrix}4\\2\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}4\\2\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]^T+\left[\begin{pmatrix}2\\4\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}2\\4\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]^T$$

$$+\left[\begin{pmatrix}2\\3\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}2\\3\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]^T+\left[\begin{pmatrix}3\\6\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}3\\6\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]^T+\left[\begin{pmatrix}4\\4\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]\left[\begin{pmatrix}4\\4\end{pmatrix}-\begin{pmatrix}3\\3.8\end{pmatrix}\right]^T \Big/ \text{N-1}$$

$$=\begin{pmatrix}1 & -0.25\\ -0.25 & 2.2\end{pmatrix}$$

Linear Discriminant Analysis (LDA) Algorithm

Samples for class ω_1 : $\mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class ω_2 : $\mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

Covariance matrix of the second class:

$$S_2=\sum_{x\in\omega_2}\frac{(x-\mu_2)(x-\mu_2)^T}{\text{N-1}}=\left[\begin{pmatrix}9\\10\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}9\\10\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T+\left[\begin{pmatrix}6\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}6\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T$$

$$+\left[\begin{pmatrix}9\\5\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}9\\5\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T+\left[\begin{pmatrix}8\\7\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}8\\7\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T+\left[\begin{pmatrix}10\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]\left[\begin{pmatrix}10\\8\end{pmatrix}-\begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^T\Big/\text{N-1}$$

$$=\begin{pmatrix}2.3 & -0.05\\ -0.05 & 3.3\end{pmatrix}$$

Linear Discriminant Analysis (LDA) Algorithm

Samples for class $\omega_1 : \mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class $\omega_2 : \mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

Within-class scatter matrix:

$$S_w = S_1 + S_2 = \begin{pmatrix} 1 & -0.25 \\ -0.25 & 2.2 \end{pmatrix} + \begin{pmatrix} 2.3 & -0.05 \\ -0.05 & 3.3 \end{pmatrix}$$

$$= \begin{pmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{pmatrix}$$

Linear Discriminant Analysis

Samples for class $\omega_1 : \mathbf{X}_1 = (x_1, x_2) = \{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class $\omega_2 : \mathbf{X}_2 = (x_1, x_2) = \{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$= \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}\right]\left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}\right]^T$$

$$= \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix}\begin{pmatrix} -5.4 & -3.8 \end{pmatrix}$$

$$= \begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix}$$

Linear Discriminant Analysis

Samples for class $\omega_1 : \mathbf{X}_1 = (x_1, x_2) = \{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class $\omega_2 : \mathbf{X}_2 = (x_1, x_2) = \{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$= \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}\right]\left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}\right]^T$$

$$= \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix}\begin{pmatrix} -5.4 & -3.8 \end{pmatrix}$$

$$= \begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix}$$

Linear Discriminant Analysis (LDA) Algorithm

- Find Eigen Vector

$$\left(S_W^{-1} S_B - \lambda I \right) \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = 0$$

$$w_1 = \begin{pmatrix} -0.5755 \\ 0.8178 \end{pmatrix}$$

$$w_2 = \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix} = w^*$$

Linear Discriminant Analysis (LDA) Algorithm

Or directly;

$$w^* = S_W^{-1}(\mu_1 - \mu_2) = \begin{pmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{pmatrix}^{-1} \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]$$

$$= \begin{pmatrix} 0.3045 & 0.0166 \\ 0.0166 & 0.1827 \end{pmatrix} \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix}$$

$$= \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix}$$

Linear Discriminant Analysis (LDA) Algorithm

- Obtain the LDA by taking the dot product of eigen vectors and original data

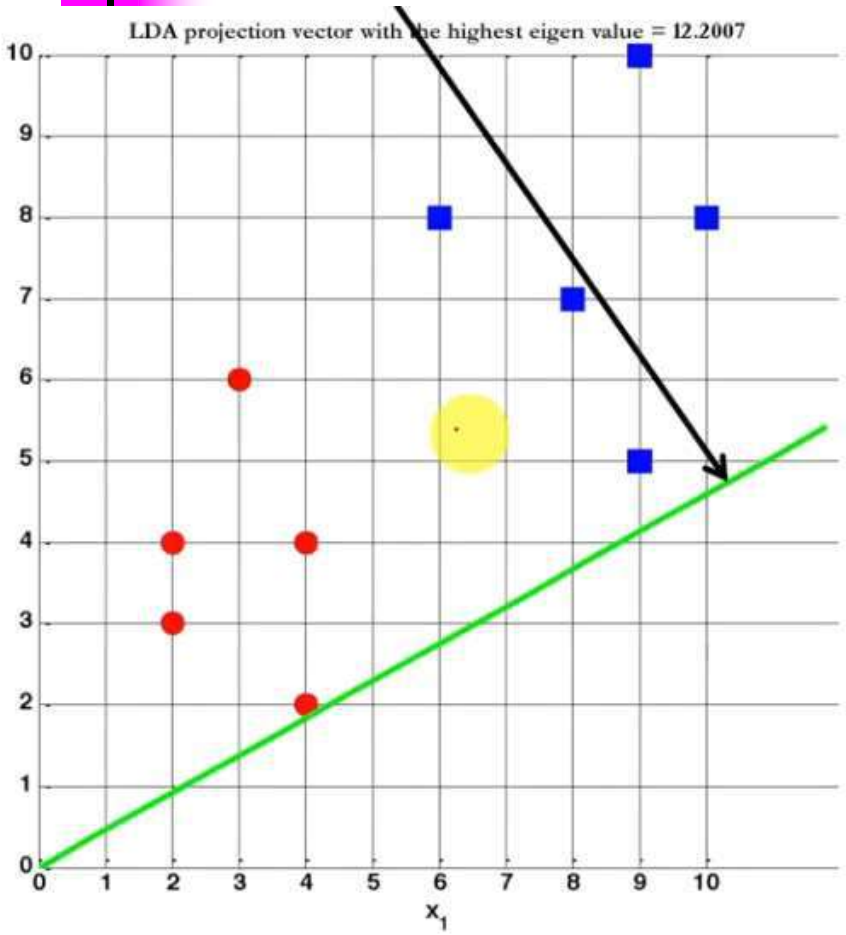
$$w_2 = \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix} = w^*$$

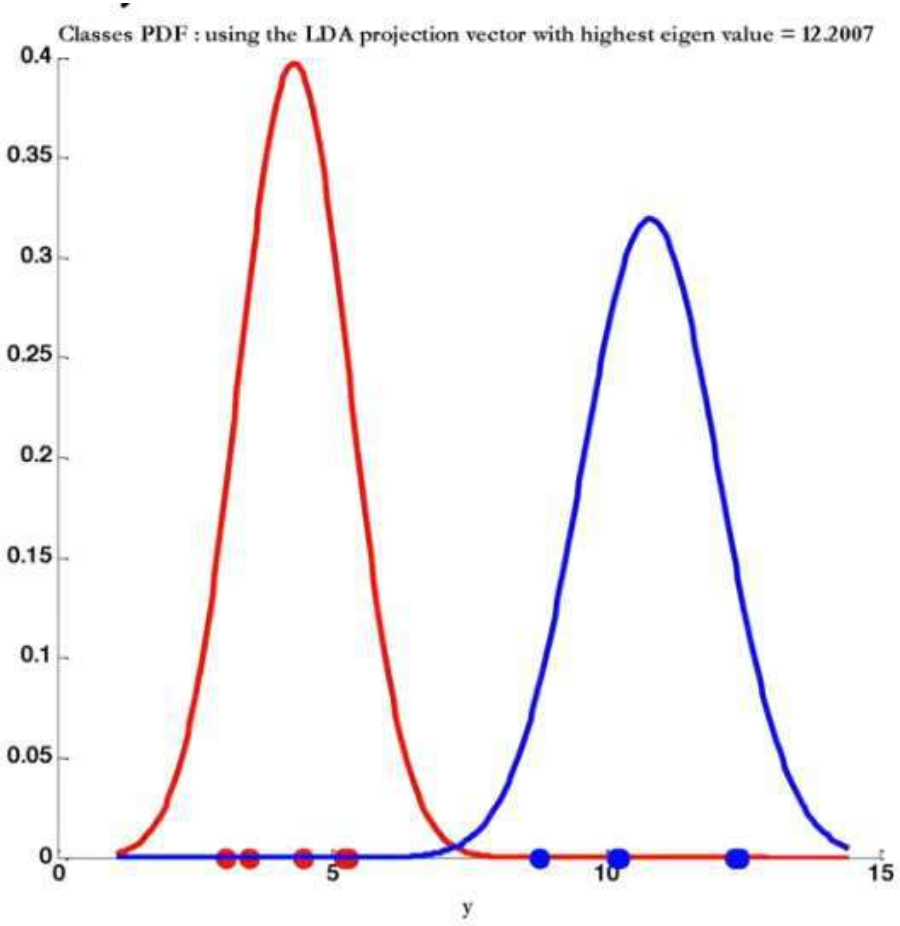
Samples for class ω_1 : $\mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class ω_2 : $\mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

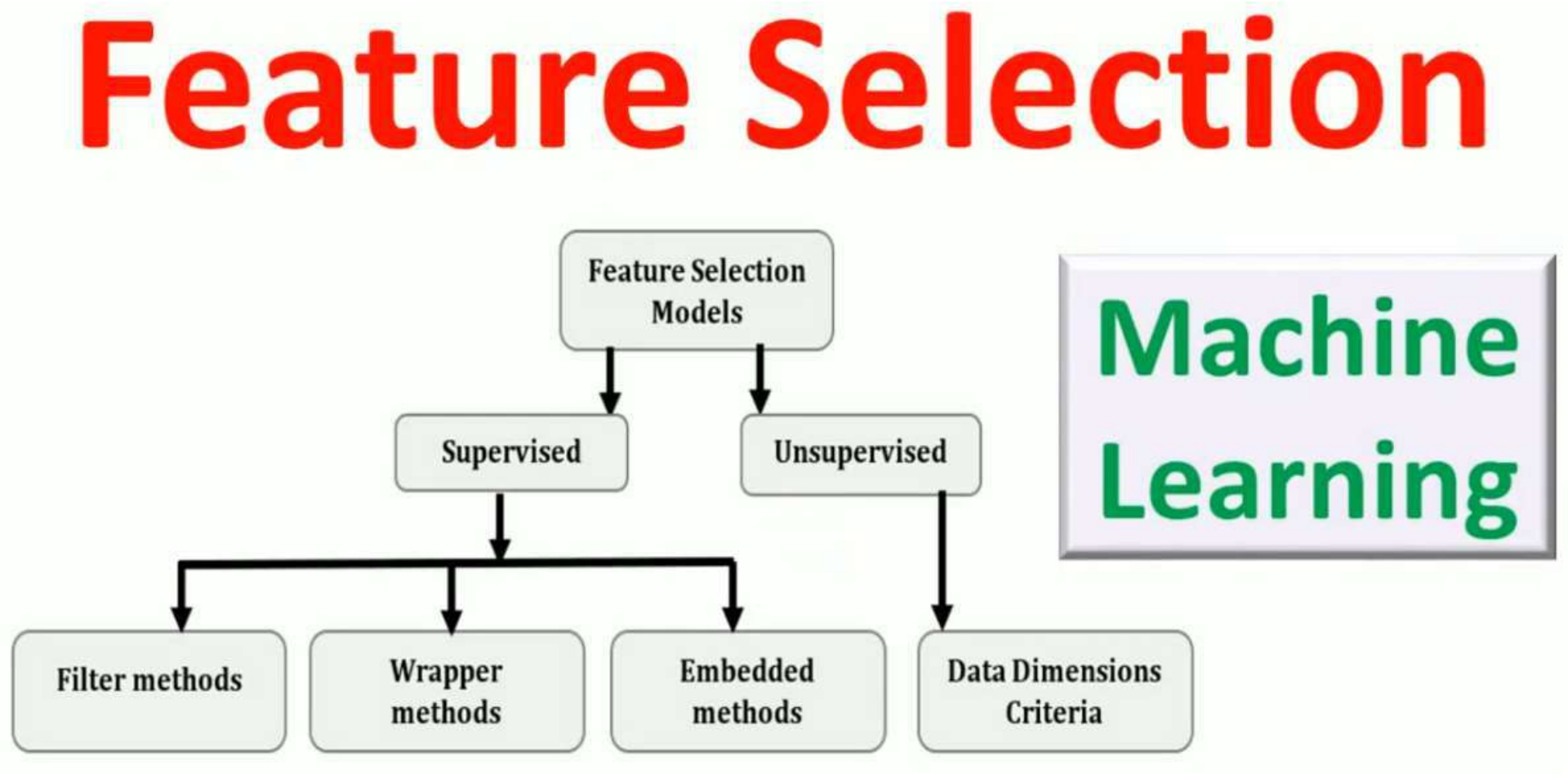
X1	4	2	2	3	4	9	6	9	8	10
X2	2	4	3	6	4	10	8	5	7	8
1st LD	4.46	3.48	3.06	5.2	5.3	12.35	8.8	10.2	10.19	12.42

Linear Discriminant Analysis (LDA) Algorithm





Feature Selection



Feature Selection

- Machine learning (ML) is a subfield of artificial intelligence that allows computers to learn without being explicitly programmed.
- Usually, machine learning model is built with help of dataset.
- A dataset is usually represented in a tabular form: rows, and columns are features.

Address	Number of Rooms	House Age	owner	price

Feature Selection

- In machine learning, feature selection selects the most relevant subset of features from the original feature set by dropping redundant, noisy, and irrelevant features.
- Feature selection is used today in many applications like Object detection, NLP, remote sensing, image retrieval, etc.

Why do We Need Feature Selection?

- If there are too many features, our model can may become weak or generate some misleading patterns.
- That happens because, usually, some features aren't correlated with the target variable and represent noise.
- If our model outputs predictions based on such features, its accuracy is likely to be subpar.
- Also, a dataset with many columns slows down the training process.

Feature Selection

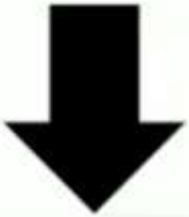
- For example, the number of rooms and address are relevant for predicting the sale price of a house, but the current owner's name isn't.

Address	Number of Rooms	House Age	owner	price

- It can confuse the learning algorithm to let the name affect the sale price prediction, which is likely to lead to wrong results and make the obtained model imprecise.

Feature Selection

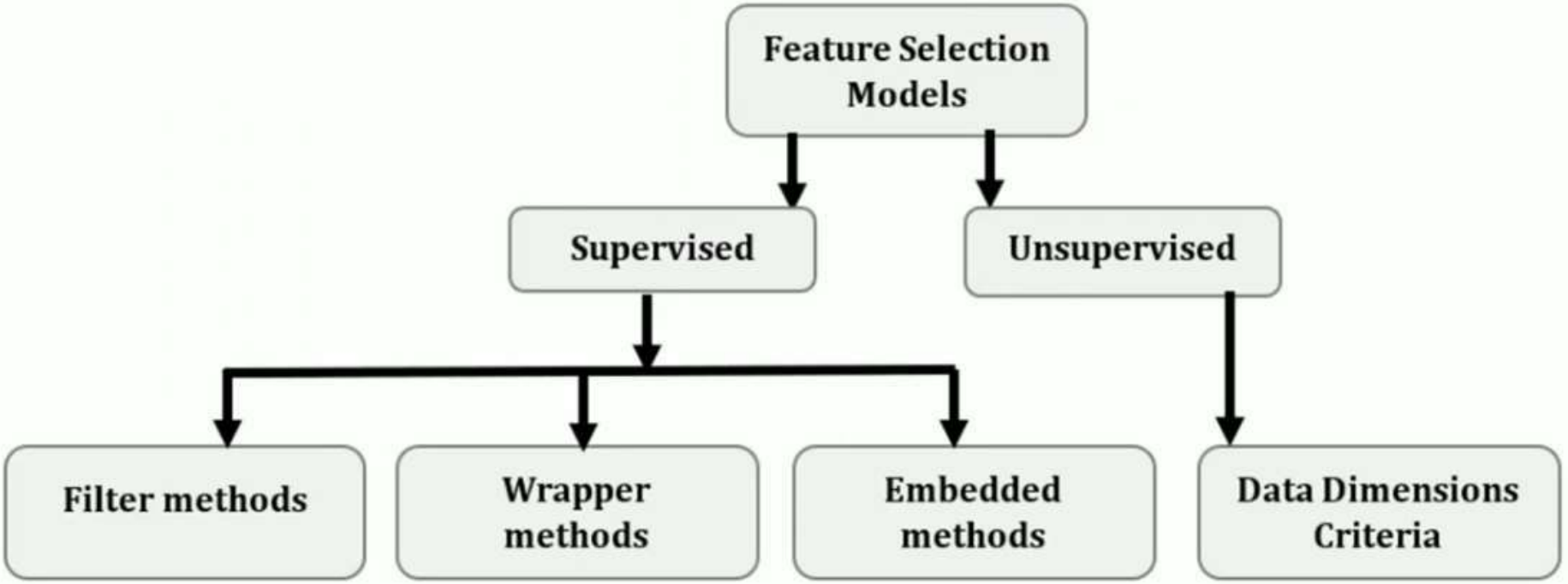
Address	Number of Rooms	House Age	owner	price



Address	Number of Rooms	House Age	Price

Feature Selection Methods

- There are several methods of doing feature selection:



Feature Selection Methods

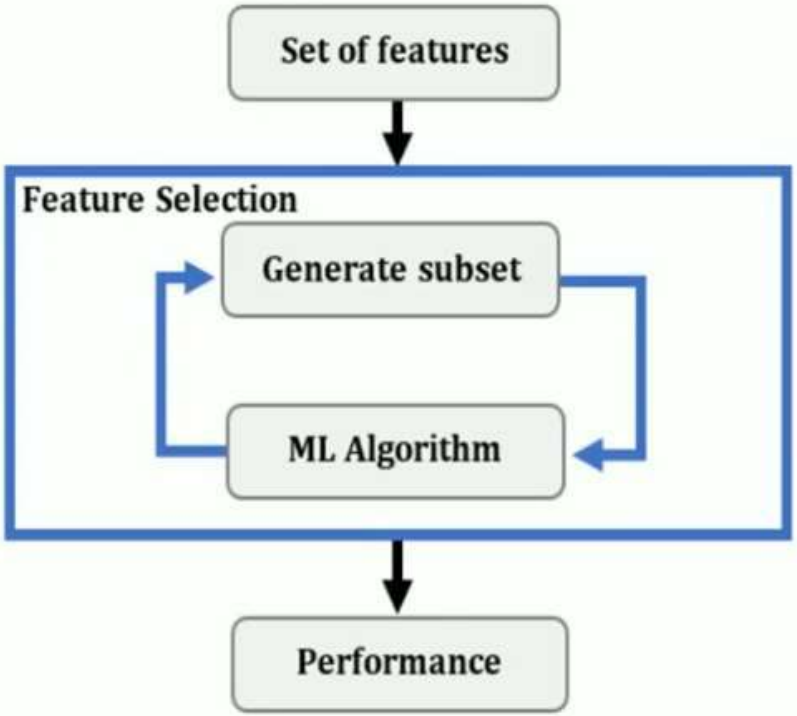
- **Unsupervised Methods**
- Unsupervised feature selection methods are applied to unlabeled data.
- An unsupervised selection method rates each feature dimension according to a number of factors, including entropy, variance, and the capacity to maintain local similarity.

Feature Selection Methods

- **Supervised Methods**
- On the other hand, we use supervised feature selection methods on labeled data.
- They determine the features that are expected to maximize the supervised model's performance.
- Supervised feature selection methods can be split into three primary categories based on the feature selection strategy.

Feature Selection Methods

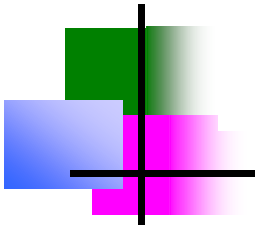
- **Wrapper Methods**
- We use a wrapper method after choosing the ML algorithm to use.
- For each feature subset, we estimate the algorithm's performance by training and evaluating it using only the features in a subset.
- Then, we add or remove features based on the estimate.
- This is an iterative process.



Feature Selection Methods

We use a greedy strategy to form feature subsets.

- In **forward wrapper methods**, we start from an empty feature set and add the feature maximizing the performance in each step until no substantial improvement is observed.
- So, if there are **n** features, we build **n** ML models in the first iteration.
- Then, we select the feature corresponding to the model with the best performance.
- In the second iteration, we repeat the process with the remaining **n-1** features.
- We continue like this as long as there's a significant performance improvement between models with which we end successive iterations.



Feature Selection Methods

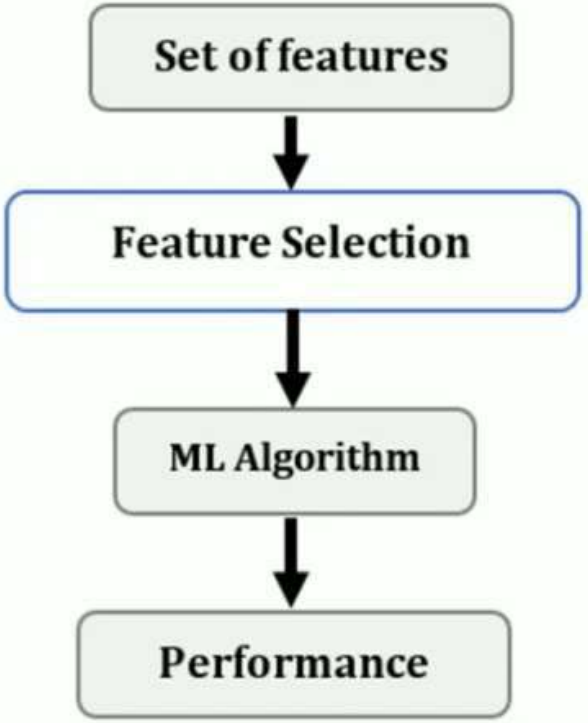
We use a greedy strategy to form feature subsets.

- **Backward methods** work the opposite way.
- They start from the full feature set and remove them one by one.
- Finally, stepwise methods reconsider features.
- So, in each iteration, they can remove a feature previously added as well as add a feature discarded in a previous step.

Feature Selection Methods

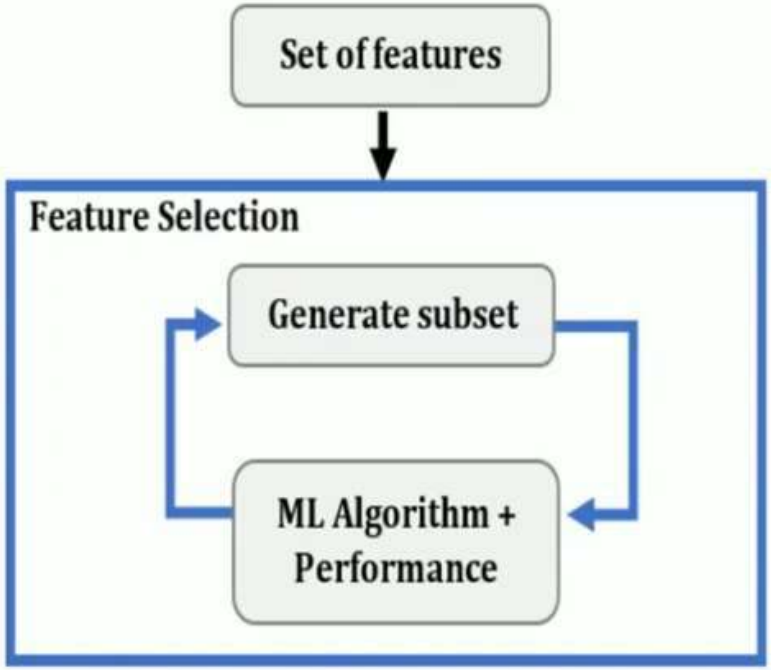
Filter Methods

- Filter methods use statistical tools to select feature subsets based on their relationship with the target.
- These methods remove features with low correlation with the target variable before training the final ML model.
- In doing so, they compute correlation and estimate the strength of the relationship using the, and other statistical tools.
- Chi-Square Test, Information Gain, Fisher's Score, Pearson correlation, ANOVA, variance thresholding



Feature Selection Methods

- **Intrinsic (or Embedded) Methods**
- Here selection of feature happens simultaneously with and is performed implicitly by the ML algorithm of our choice.
- During training, some steps of the ML algorithm do feature selection:
- For instance, this is the case with decision trees.
- At each node split, they choose the best feature to split the data by.
- Those choices represent feature selection.



Benefits and Shortcomings Feature Selection

Feature selection methods allow us to:

- Reduce overfitting as less redundant data means less chance to make decisions based on noise;
- Improve accuracy by removing misleading and unimportant data;
- Reduce training time since data with fewer columns mean faster training.
- However, feature selection methods are hard to apply to high-dimensional data.
- The more features we have, the longer it takes for selection to complete.
- Also, there's the risk of overfitting when there aren't enough observations.

Forward Feature Selection

Subset Selection

- In machine learning subset selection or feature selection is the process of selecting a subset of relevant features for use in model construction.
- The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information.
- There are mainly two approaches to subset selection.
  - forward selection and
  - backward selection.

Forward Feature Selection

x_1 x_2 x_3 t

We use the following notations:

n	:	number of input variables
$x_1, \ldots, x_n$	:	input variables
F_i	:	a subset of the set of input variables
$E(F_i)$	:	error incurred on the validation sample when only the inputs in F_i are used

Forward Feature Selection

1. Set $F_0 = \varnothing$ and $E(F_0) = \infty$.
2. For $i = 0, 1, \ldots,$ repeat the following until $E(F_{i+1}) \geq E(F_i)$:
   (a) For all possible input variables x_j, train the model with the input variables $F_i \cup \{x_j\}$ and calculate $E(F_i \cup \{x_j\})$ on the validation set.
   (b) Choose that input variable x_m that causes the least error $E(F_i \cup \{x_j\})$:

$$m = \arg\min_j E(F_i \cup \{x_j\})$$

   (c) Set $F_{i+1} = F_i \cup \{x_m\}$.
3. The set F_i is outputted as the best subset.

Forward Feature Selection

1. Set $F_0 = \varnothing$ and $E(F_0) = \infty$.
2. For $i = 0, 1, \ldots,$ repeat the following until $E(F_{i+1}) \geq E(F_i)$:
   (a) For all possible input variables x_j, train the model with the input variables $F_i \cup \{x_j\}$ and calculate $E(F_i \cup \{x_j\})$ on the validation set.
   (b) Choose that input variable x_m that causes the least error $E(F_i \cup \{x_j\})$:

$$m = \arg\min_j E(F_i \cup \{x_j\})$$

   (c) Set $F_{i+1} = F_i \cup \{x_m\}$.
3. The set F_i is outputted as the best subset.

Forward Feature Selection

Remarks

1. In this procedure, we stop if adding any feature does not decrease the error E. We may even decide to stop earlier if the decrease in error is too small, where there is a user-defined threshold that depends on the application constraints.
2. This process may be costly because to decrease the dimensions from n to k, we need to train and test the system $n + (n - 1) + (n - 2) + \ldots + (n - k)$ times, which is $O(n^2)$.

Feature Selection
Backward Elimination Technique

We use the following notations:

n	:	number of input variables
$x_1, \ldots, x_n$	:	input variables
F_i	:	a subset of the set of input variables
$E(F_i)$	:	error incurred on the validation sample when only the inputs in F_i are used

Feature Selection
Backward Elimination Technique

1. Set $F_0 = \{x_1, \ldots, x_n\}$ and $E(F_0) = \infty$.
2. For $i = 0, 1, \ldots,$ repeat the following until $E(F_{i+1}) \geq E(F_i)$:
   (a) For all possible input variables x_j, train the model with the input variables $F_i - \{x_j\}$) and calculate $E(F_i - \{x_j\})$ on the validation set.
   (b) Choose that input variable x_m that causes the least error $E(F_i - \{x_j\})$:

$$m = \arg\min_j E(F_i - \{x_j\})$$

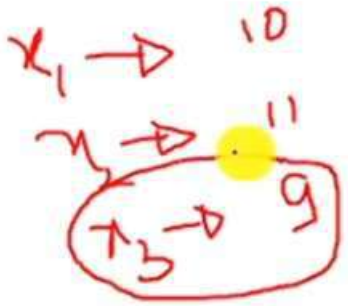
   (c) Set $F_{i+1} = F_i - \{x_m\}$.
3. The set F_i is outputted as the best subset.

Feature Selection
Backward Elimination Technique

1. Set $F_0 = \{x_1, \ldots, x_n\}$ and $E(F_0) = \infty$.
2. For $i = 0, 1, \ldots,$ repeat the following until $E(F_{i+1}) \geq E(F_i)$:
   (a) For all possible input variables x_j, train the model with the input variables $F_i - \{x_j\}$ and calculate $E(F_i - \{x_j\})$ on the validation set.
   (b) Choose that input variable x_m that causes the least error $E(F_i - \{x_j\})$:

$$m = \arg\min_j E(F_i - \{x_j\})$$

   (c) Set $F_{i+1} = F_i - \{x_m\}$.
3. The set F_i is outputted as the best subset.



Chi-Square Test for Feature Selection

- Feature selection is process of selecting relevant features or variables that have strong correlation with the target variable.
- By identifying and keeping only the relevant features, we can improve the performance of the machine learning model.

Gender	Occupation	Income	Target
Male	Student	Low	Subscribed
Male	Working	High	Subscribed
Female	Student	Medium	Not Subscribed
Female	Self Employed	Low	Subscribed

Chi-Square Test for Feature Selection

- The chi-square test is a statistical test used to determine if there is a significant correlation between two categorical variables.
- It is a non-parametric test, meaning it makes no assumptions about the distribution of the data.

Chi-Square Test for Feature Selection

- **Step 1:** Define the Null and Alternate hypothesis
  - **Null Hypothesis (H0):** There is no significant association between the two categorical variables.
  - **Alternative Hypothesis (H1):** There is a significant association between the two categorical variables.

Chi-Square Test for Feature Selection

- **Step 2:** Calculate the contingency table form the given data

Gender	Occupation	Income	Target
Male	Student	Low	Subscribed
Male	Working	High	Subscribed
Female	Student	Medium	Not Subscribed
Female	Self Employed	Low	Subscribed

	Subscribed (O)	Not Subscribed(O)
Low Income	20	30
Medium Income	40	25
High Income	10	15

Chi-Square Test for Feature Selection

- **Step 3:** Calculate the expected frequencies

- $E_{LI,NS} = \frac{Row\ Total * Col\ Total}{Total}$
- $E_{LI,NS} = \frac{50*70}{140} = 25$
- $E_{MI,NS} = \frac{65*70}{140} = 32.5$
- $E_{HI,NS} = \frac{25*70}{140} = 12.5$

	Subscribed (O)	Not Subscribed (O)	Total
Low Income	20	30	50
Medium Income	40	25	65
High Income	10	15	25
Total	70	70	140

Chi-Square Test for Feature Selection

- **Step 4:** Calculate the chi-square value

	Subscribed (O)	Not Subscribed (O)	Subscribed (E)	Not Subscribed (E)
Low Income	20	30	35	35
Medium Income	40	25	32.5	32.5
High Income	10	15	12.5	12.5

- $\chi^2 = \sum \frac{(O-E)^2}{E}$
- $\chi^2 = \frac{(20-35)^2}{35} + \frac{(40-32.5)^2}{32.5} + \frac{(10-12.5)^2}{12.5} + \frac{(30-35)^2}{35} + \frac{(25-32.5)^2}{32.5} + \frac{(15-12.5)^2}{12.5}$

Chi-Square Test for Feature Selection

- **Step 4:** Calculate the chi-square value

	Subscribed (O)	Not Subscribed (O)	Subscribed (E)	Not Subscribed (E)
Low Income	20	30	35	35
Medium Income	40	25	32.5	32.5
High Income	10	15	12.5	12.5

- $\chi^2 = \sum \frac{(O-E)^2}{E}$
- $\chi^2 = \frac{(20-35)^2}{35} + \frac{(40-32.5)^2}{32.5} + \frac{(10-12.5)^2}{12.5} + \frac{(30-35)^2}{35} + \frac{(25-32.5)^2}{32.5} + \frac{(15-12.5)^2}{12.5}$
- $\chi^2 = 11.6$

Chi-Square Test for Feature Selection

Chi-Square Test for Feature Selection

- **Step 5:** Compare the chi-square value with critical value to Accept or Reject the Null hypothesis
- $Degree\ of\ Freedom(df) = (No\ of\ Rows\ - 1)(No\ of\ Collums - 1)$
- $Degree\ of\ Freedom(df) = 2 * 1 = 2$
- $Significance\ Level = 0.05$

Degrees of freedom (*df*)	Significance level (α)							
	.99	.975	.95	.9	.1	.05	.025	.01
1	-------	0.001	0.004	0.016	2.706	3.841	5.024	6.635
2	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210
3	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345
4	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277
5	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086
6	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812
7	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475
8	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090
9	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666
10	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209
11	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725
12	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217
13	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688
14	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141
15	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578
16	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000
17	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409
18	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805
19	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191
20	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566
21	8.897	10.283	11.591	13.240	29.615	32.671	35.479	38.932
22	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289
23	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638
24	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.980
25	11.524	13.120	14.611	16.473	34.382	37.652	40.646	44.314
26	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642
27	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963
28	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278
29	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588
30	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892
40	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691
50	29.707	32.357	34.764	37.689	63.167	67.505	71.420	76.154
60	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379
70	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425

Chi-Square Test for Feature Selection

Chi-Square Test for Feature Selection

- **Step 5:** Compare the chi-square value with critical value to Accept or Reject the Null hypothesis
- $Degree\ of\ Freedom(df) = (No\ of\ Rows - 1)(No\ of\ Collums - 1)$
- $Degree\ of\ Freedom(df) = 2 * 1 = 2$
- $Significance\ Level = 0.05$
- $\chi^2_{2,0.05} = 5.99$

Degrees of freedom (*df*)	Significance level (α)							
	.99	.975	.95	.9	.1	.05	.025	.01
1	-------	0.001	0.004	0.016	2.706	3.841	5.024	6.635
2	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210
3	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345
4	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277
5	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086
6	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812
7	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475
8	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090
9	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666
10	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209
11	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725
12	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217
13	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688
14	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141
15	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578
16	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000
17	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409
18	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805
19	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191
20	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566
21	8.897	10.283	11.591	13.240	29.615	32.671	35.479	38.932
22	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289
23	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638
24	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.980
25	11.524	13.120	14.611	16.473	34.382	37.652	40.646	44.314
26	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642
27	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963
28	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278
29	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588
30	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892
40	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691
50	29.707	32.357	34.764	37.689	63.167	67.505	71.420	76.154
60	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379
70	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425
80	53.540	57.153	60.391	64.278	96.578	101.879	106.629	112.329

Chi-Square Test for Feature Selection

- **Step 5:** Compare the chi-square value with critical value to Accept or Reject the Null hypothesis
- $\chi^2 = 11.6$
- $\chi^2_{2,0.05} = 5.99$
- The calculated value is greater than the critical value, you would **reject the null hypothesis** and **accept the alternate hypothesis**
- This means "income level" is a relevant feature for predicting subscription status.

Issues in Machine Learning

1. **Which algorithms perform best for which types of problems and representations?**
   - What algorithms exist for learning general target functions from specific training examples?
   - In what settings will particular algorithms converge to the desired function, given sufficient training data?
2. **How much training data is sufficient?**
   - What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?
3. **Can prior knowledge be helpful even when it is only approximately correct?**
   - When and how can **prior knowledge** held by the learner guide the process of generalizing from examples?

Issues in Machine Learning

4. What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
5. How can the learner automatically alter **its** representation to improve its ability to represent and learn the target function?
6. What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?

Thank you for your attention