LINEAR DISCRIMINANT ANALYSIS

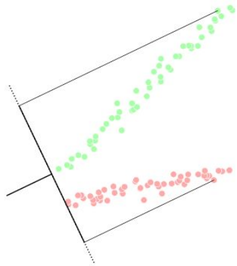
Group 1 and 2

What is LDA?

Linear Discriminant Analysis (LDA) is a supervised machine learning method used for both classification and dimensionality reduction.

It finds linear boundaries between classes by projecting data into a lower-dimensional space where class separation is maximized.

Medical diagnosis, Fraud detection, Face recognition and Marketing.



Why LDA?

Classification is everywhere: spam filters, diagnostics, risk assessment.

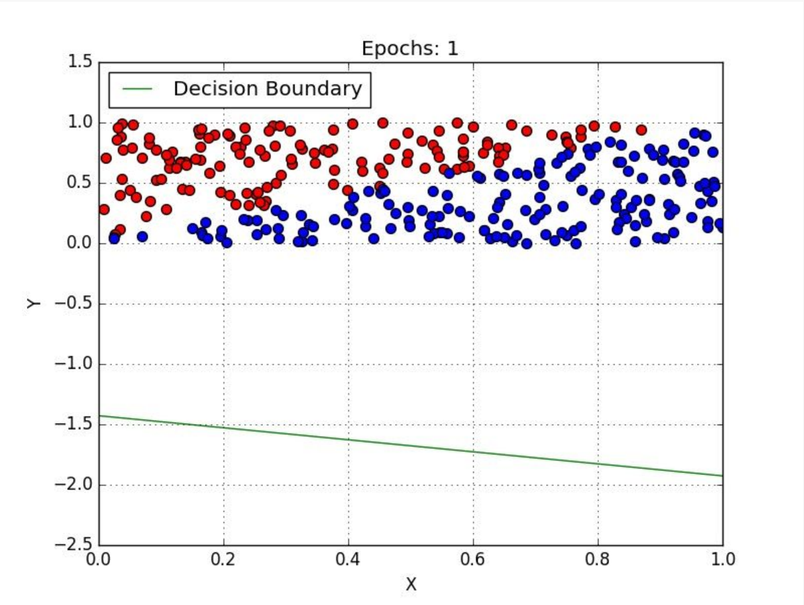
We want models that are: **Accurate, Stable, Interpretable, Scalable to multiple classes**.

Are existing classifiers enough?

Why LDA?

Logistic Regression:

- Models probabilities with a sigmoid curve.
- Good for binary outcomes, interpretable.
- Fails with overlapping or non-linear decision boundaries.



Issues:

- Coefficients are unstable when classes are well-separated.
- Hard to extend beyond binary classes.

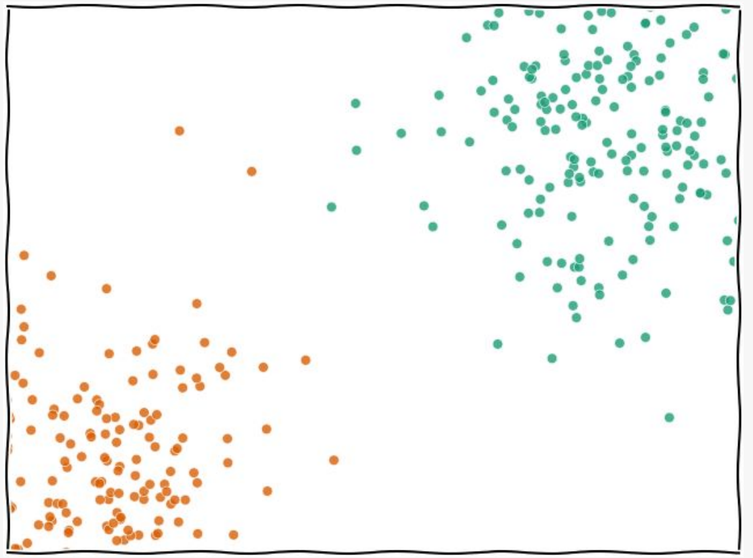
Why LDA?

K Nearest Neighbors:

- Assigns class based on closest neighbors.

- Flexible, non-parametric.

- Fails with high dimensions, noisy data, and requires lots of data.



Issues:

- Suffers in high dimensions (curse of dimensionality).

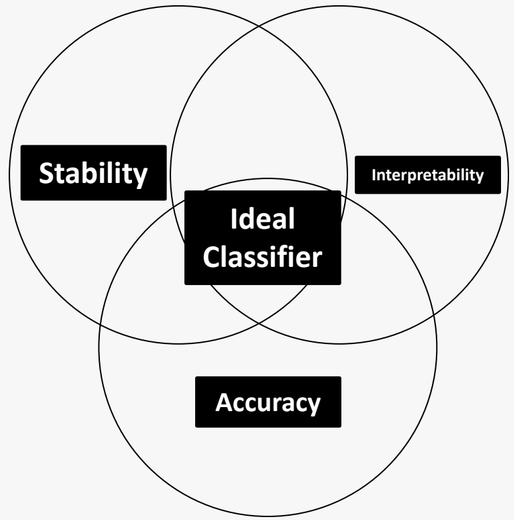
- Sensitive to irrelevant features.

What an Ideal Classifier Should Do?

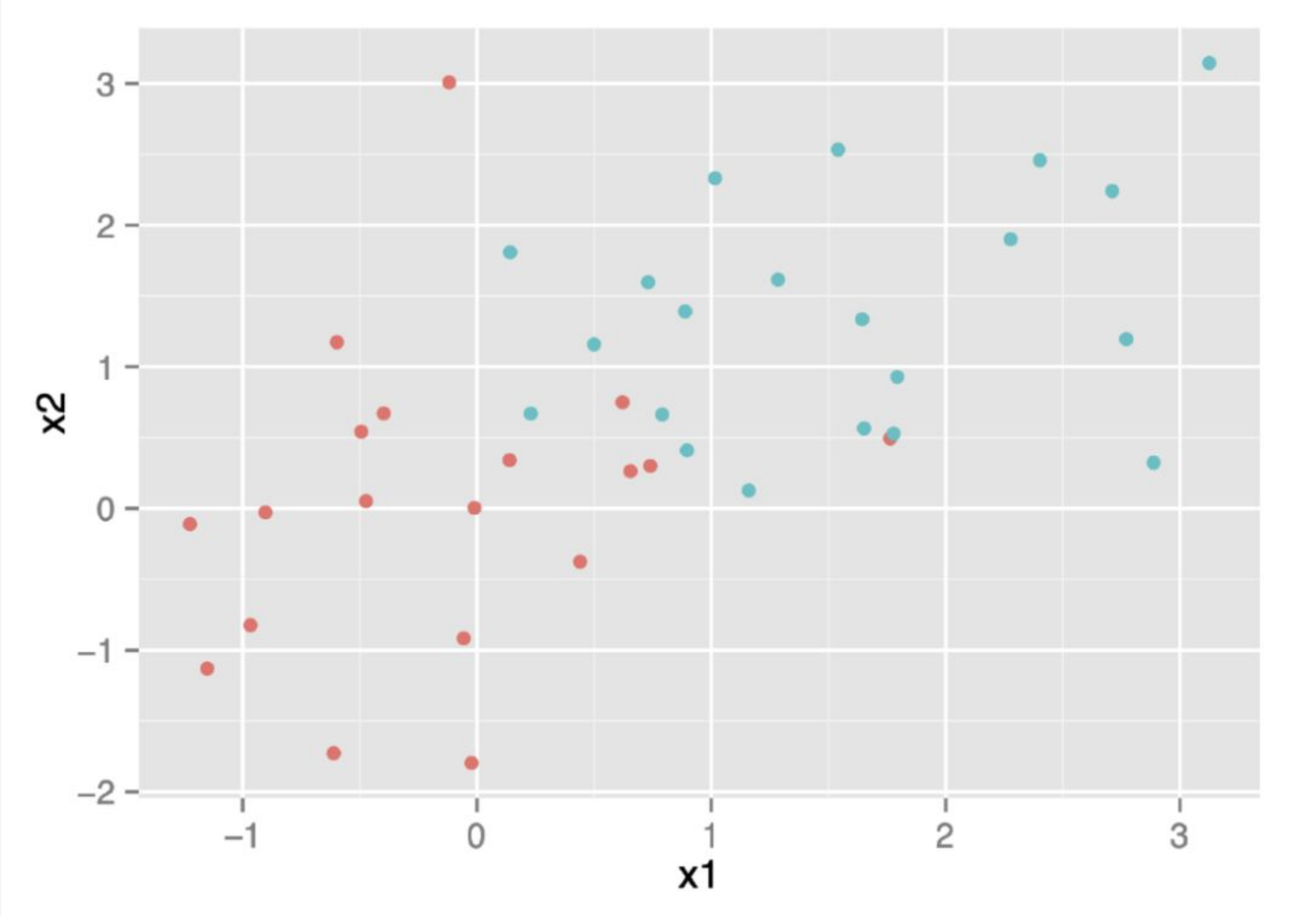
Balance between **simplicity and power**:

- Handle multiple classes naturally.
- Be robust to collinearity and high dimensions.
- Provide clear decision boundaries.

Ideally, it should also reduce dimension without losing discriminative power.

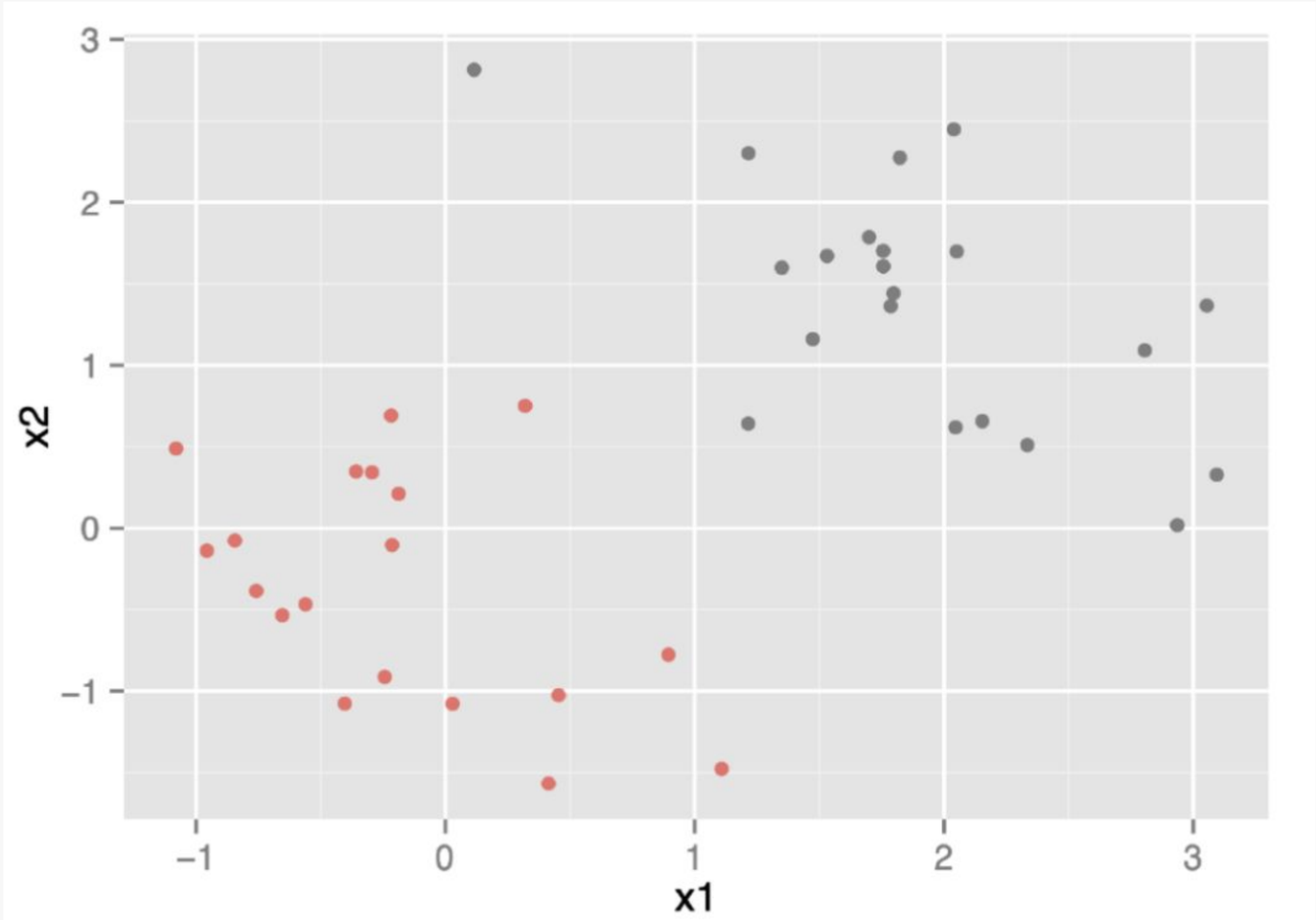


Scenario 1



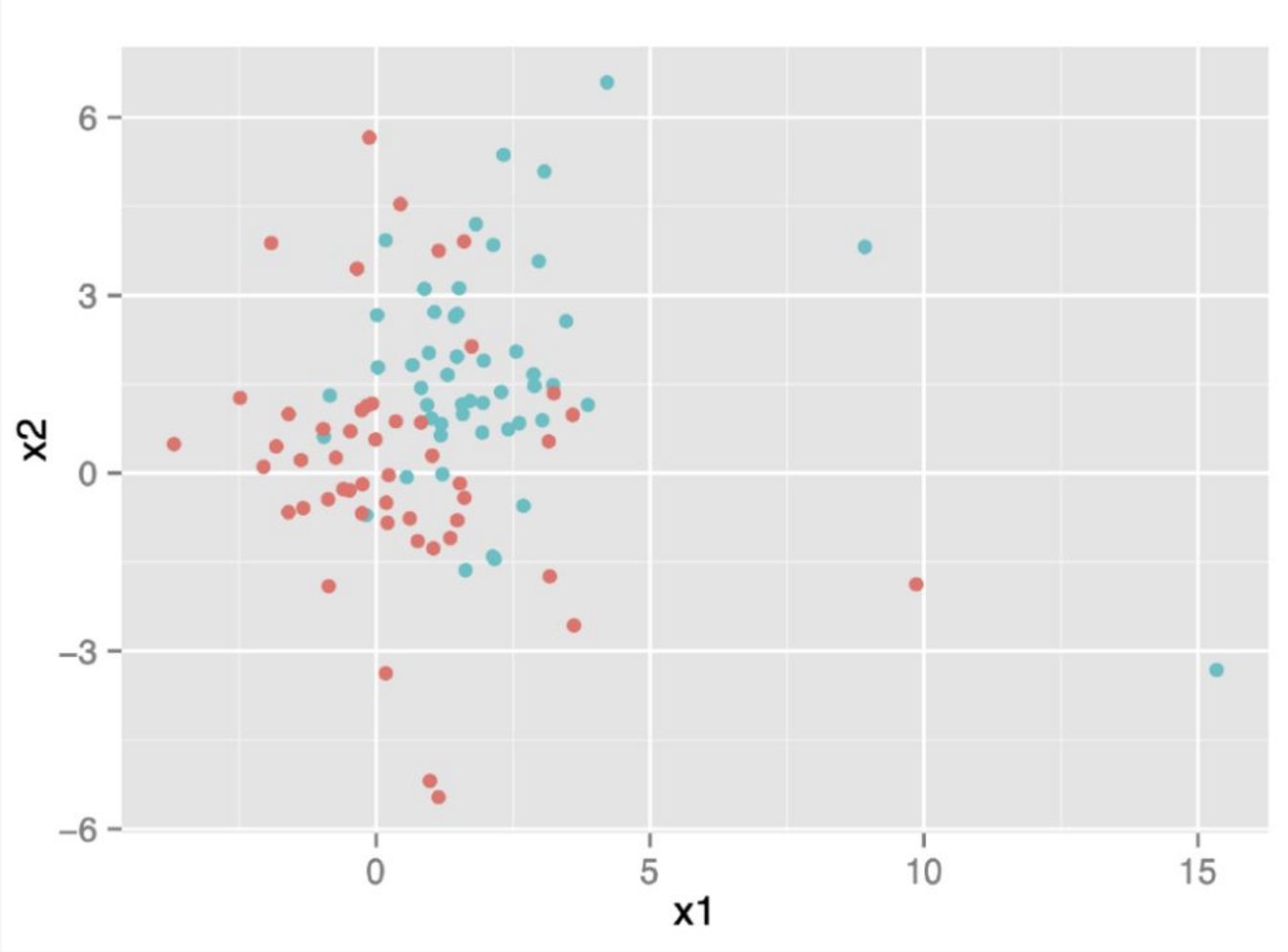
- x1 x2 normal with identical variance
- No correlation in either class

Scenario 2



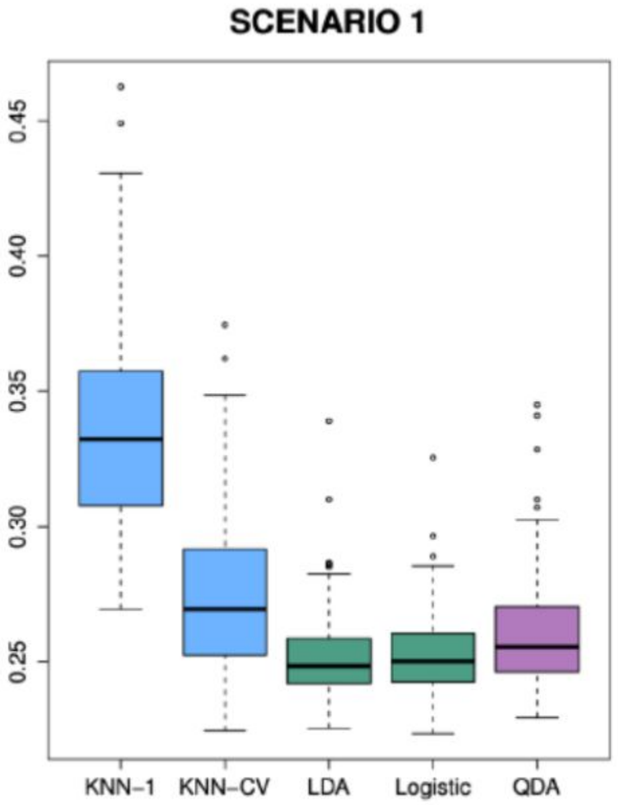
- x1 x2 normal with identical variance
- Correlation is -0.5 I both classes

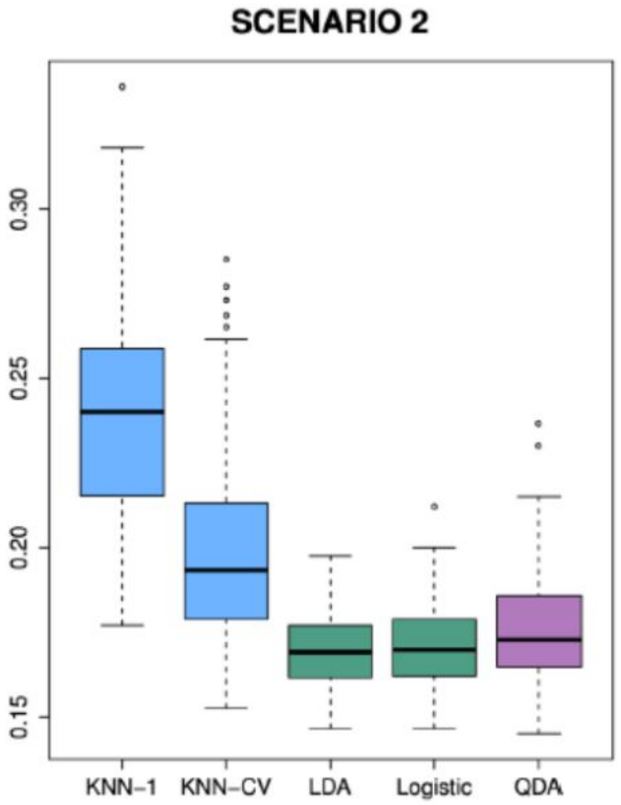
Scenario 3

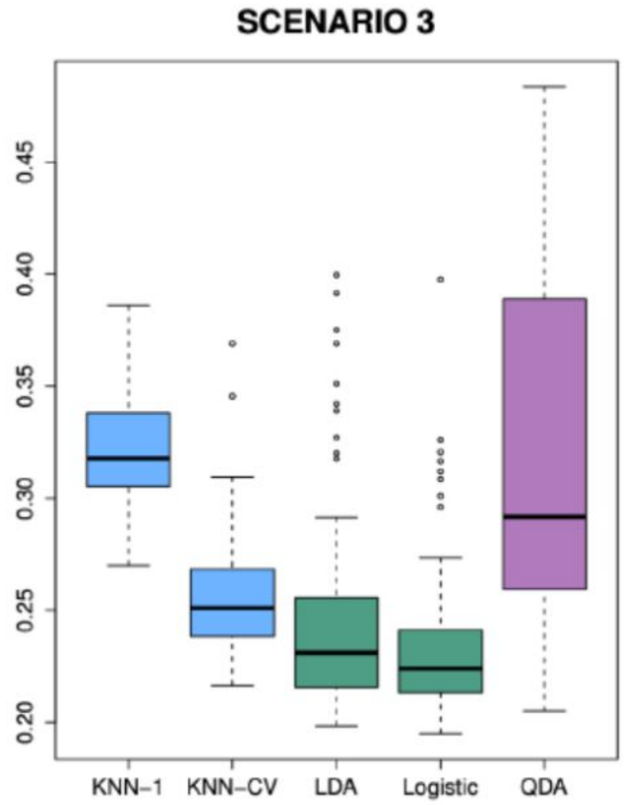


- x1 x2 student T
- No correlation in either class

Results

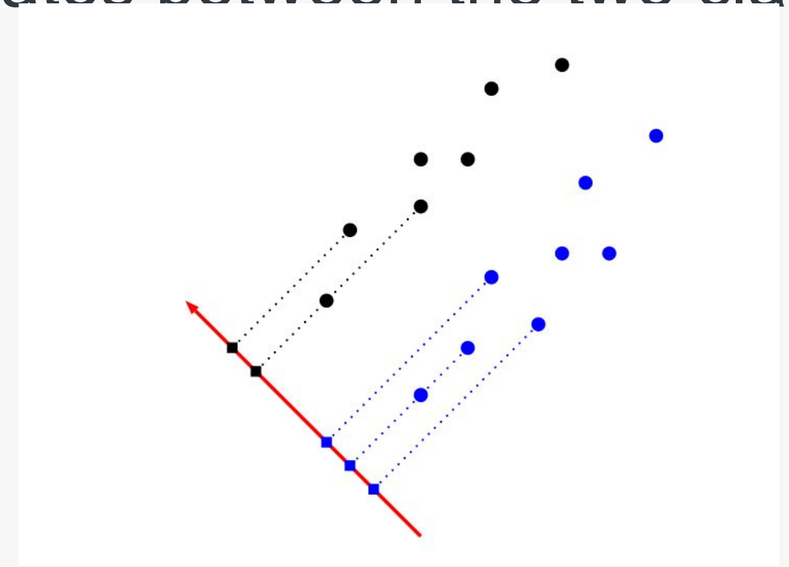

SCENARIO 1
0.25
0.30
0.35
0.40
0.45
KNN−1
KNN−CV
LDA
Logistic
QDA


SCENARIO 2
0.15
0.20
0.25
0.30
KNN−1
KNN−CV
LDA
Logistic
QDA

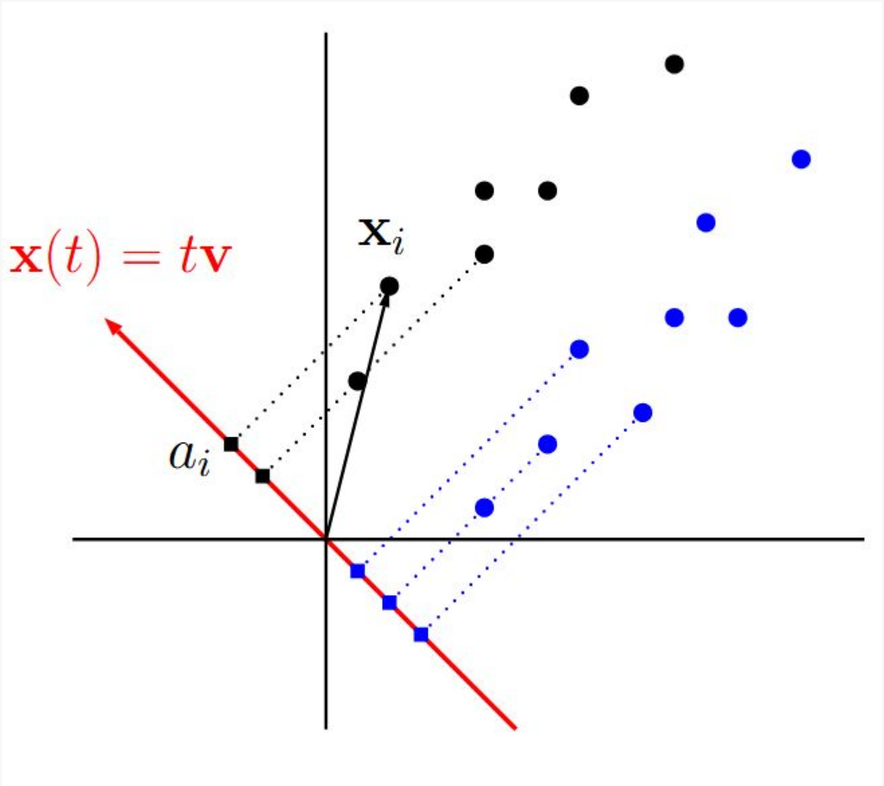

SCENARIO 3
0.20
0.25
0.30
0.35
0.40
0.45
KNN−1
KNN−CV
LDA
Logistic
QDA

The two-class LDA problem

Given a training data set x1, . . . , xn ∈ Rd consisting of two classes C1, C2, find a (unit-vector) direction that “best” discriminates between the two classes.



Mathematical setup



First, observe that projections of the two classes onto parallel lines always have “the same amount of separation”.

This time we are going to focus on lines that pass through the origin.

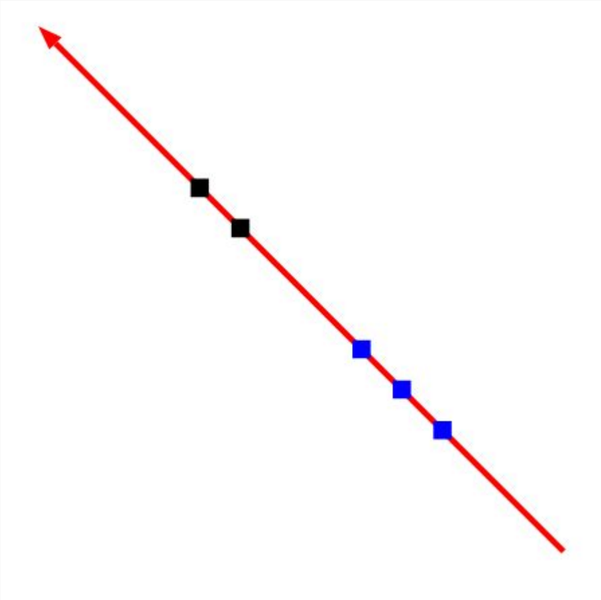
The 1D proje

$$a_i = \mathbf{v}^T \mathbf{x}_i, \quad i = 1, \ldots, n$$

Note that they also carry the labels of the original data.

Mathematical setup

Now the data looks like:



One (naive) idea is to measure the distance between the two class means in the 1D projection space: |μ1 − μ2|

$$\mu_1 = \frac{1}{n_1} \sum_{\mathbf{x}_i \in C_1} a_i = \frac{1}{n_1} \sum_{\mathbf{x}_i \in C_1} \mathbf{v}^T \mathbf{x}_i$$
$$= \mathbf{v}^T \cdot \frac{1}{n_1} \sum_{\mathbf{x}_i \in C_1} \mathbf{x}_i = \mathbf{v}^T \mathbf{m}_1$$

How do we quantify the separation between the two classes (in order to compare different directions v and select the best one)?

$$\mu_2 = \mathbf{v}^T \mathbf{m}_2, \quad \mathbf{m}_2 = \frac{1}{n_2} \sum_{\mathbf{x}_i \in C_2} \mathbf{x}_i.$$

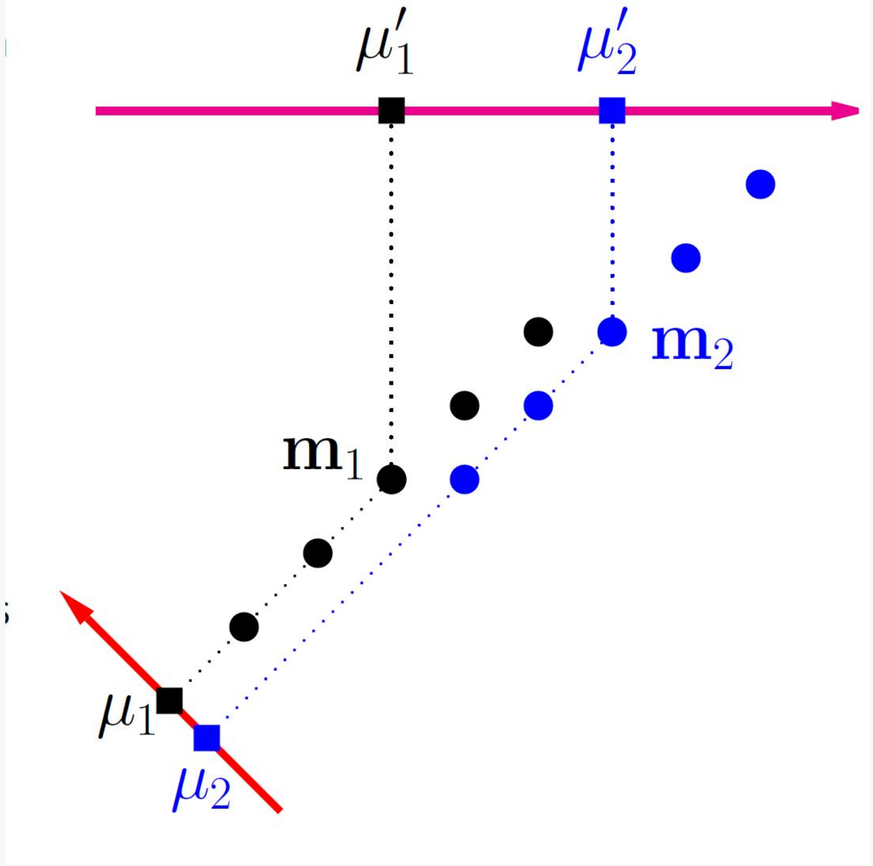
Mathematical setup

That is, we solve the following problem

$$\max_{\mathbf{v}:\,\|\mathbf{v}\|=1} |\mu_1 - \mu_2|$$

$$\mu_j = \mathbf{v}^T \mathbf{m}_j, \;\; j = 1, 2.$$

However, this criterion does not always work (as shown in the right plot).



* What else do we need to control?

Mathematical setup

It turns out that we should also pay attention to the variances of the projected classes:

$$s_1^2 = \sum_{\mathbf{x}_i \in C_1} (a_i - \mu_1)^2, \quad s_2^2 = \sum_{\mathbf{x}_i \in C_2} (a_i - \mu_2)^2$$

Ideally, the projected classes have both faraway means and small variances.

This can be achieved through the following modified formulation:

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1} \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2}.$$

The optimal v should be such that

- $(\mu_1 - \mu_2)^2$: large
- s_1^2, s_2^2: both small

Mathematical Derivation

First, we derive a formula for the distance between the two projected centroids:

$$\begin{aligned}(\mu_1 - \mu_2)^2 &= (\mathbf{v}^T\mathbf{m}_1 - \mathbf{v}^T\mathbf{m}_2)^2 = (\mathbf{v}^T(\mathbf{m}_1 - \mathbf{m}_2))^2 \\ &= \mathbf{v}^T(\mathbf{m}_1 - \mathbf{m}_2) \cdot (\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{v} \\ &= \mathbf{v}^T\mathbf{S}_b\mathbf{v},\end{aligned}$$

Where the between-class scatter matrix:

$$\mathbf{S}_b = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \in \mathbb{R}^{d\times d}$$

Remark. Clearly, Sb is square, symmetric and positive semidefinite. Moreover, rank(Sb) = 1, which implies that it only has 1 positive eigenvalue!

Mathematical Derivation

The total within-class scatter of the two classes in the projection space is

$$s_1^2 + s_2^2 = \mathbf{v}^T \mathbf{S}_1 \mathbf{v} + \mathbf{v}^T \mathbf{S}_2 \mathbf{v} = \mathbf{v}^T (\mathbf{S}_1 + \mathbf{S}_2) \mathbf{v} = \mathbf{v}^T \mathbf{S}_w \mathbf{v}$$

Where the total within-class scatter matrix:

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2 = \sum_{\mathbf{x}_i \in C_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^T + \sum_{\mathbf{x}_i \in C_2} (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2)^T$$

Remark. Sw from Rd×d is also square, symmetric, and positive semidefinite

Mathematical Derivation

Putting everything together, we have derived the following optimization problem:

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1} \frac{\mathbf{v}^T \mathbf{S}_b \mathbf{v}}{\mathbf{v}^T \mathbf{S}_w \mathbf{v}}$$

Suppose Sw is nonsingular. The maximizer of the problem is given by the largest eigenvector v1 of S−1 w Sb, i.e.

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{v}_1 = \lambda_1 \mathbf{v}_1.$$

Remark. rank(S−1w Sb) = rank(Sb) = 1, so 1 is the only nonzero (positive) eigenvalue that can be found. It represents the the largest amount of separation between the two classes along any single direction.

Computing

The following are different ways of finding the optimal direction v1:

- Slowest way (via three expensive steps):
  - 1. work really hard to invert the d × d matrix Sw,
  - 2. do the matrix multiplication Sw−1Sb
  - 3. solve the eigenvalue problem Sw−1Sbv1 = lambda1v1
- A slight better way: Rewrite as a generalized eigenvalue problem
  - Sbv1 = lambda1Swv1,
  - and then solve it through functions like eigs(A,B) in MATLAB

Computing

- The smartest way is to rewrite as

$$\lambda_1 \mathbf{v}_1 = \mathbf{S}_w^{-1} \underbrace{(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T}_{\mathbf{S}_b} \mathbf{v}_1$$

$$= \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \cdot \underbrace{(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{v}_1}_{\text{scalar}}$$

This implies that

$$\mathbf{v}_1 \propto \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

and it can be computed from $\mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$ through rescaling!

Remark: Here, inverting Sw should still be avoided; instead, one should implement this by solving a linear system Swx = m1 −m2. This can be done through Sw \ (m1 −m2) in MATLAB

Examples

Data

- Class 1 has three points (1,2), (2,3), (3, 4.9), with mean m1 = (2, 3.3)^T
- Class 2 has three points (2,1), (3,2), (4, 3.9), with mean m2 = (3, 2.3)^T

Within-class scatter matrix

$$\mathbf{S}_w = \begin{pmatrix} 4 & 5.8 \\ 5.8 & 8.68 \end{pmatrix}$$

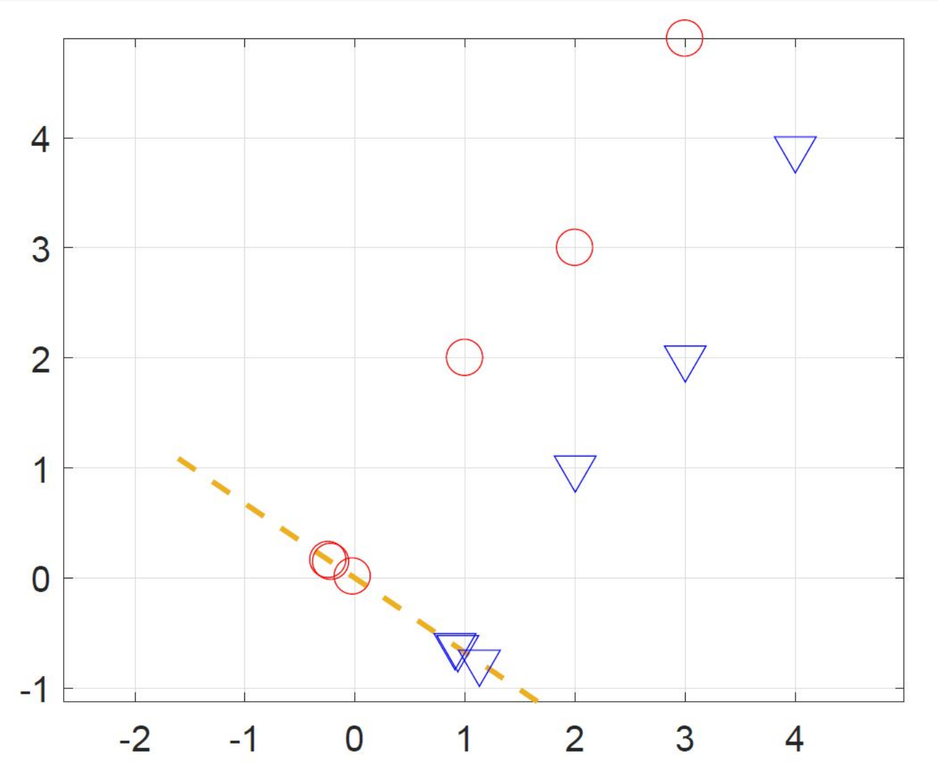
Thus, the optimal direction is

$$\mathbf{v} = \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

= (−13.4074, 9.0741)^T
= (−0.8282, 0.5605)^T [On normalization]

Examples

and the projection coordinates are
Y = [0.2928, 0.0252, 0.2619,−1.0958,−1.3635,−1.1267]



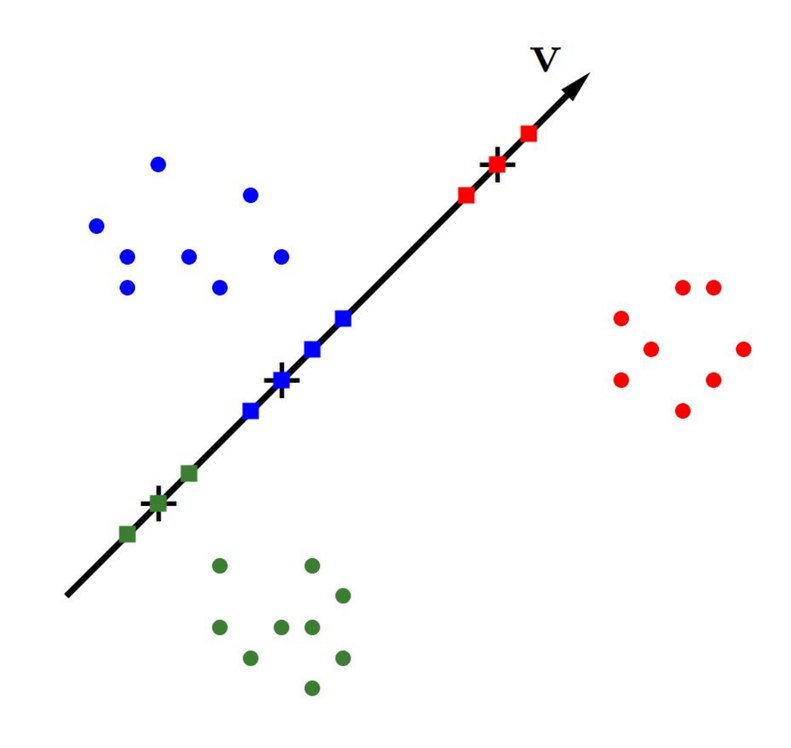
Multiclass Extension

The previous procedure only applies to 2 classes. When there are c >= 3 classes, what is the “most discriminatory” direction?

It will be based on the same intuition that the optimal direction v should project the different classes such that

- each class is as tight as possible;
- their centroids are as far from each other as possible.

Both are actually about **variances.**



Mathematical Derivation

For any unit vector v, the tightness of the projected classes (of the training data) is still described by the total within-class scatter:

$$\sum_{j=1}^{c} s_j^2 = \sum \mathbf{v}^T \mathbf{S}_j \mathbf{v} = \mathbf{v}^T \left(\sum \mathbf{S}_j \right) \mathbf{v} = \mathbf{v}^T \mathbf{S}_w \mathbf{v}$$

where the Sj , 1 <= j <= c are defined in the same way as before:

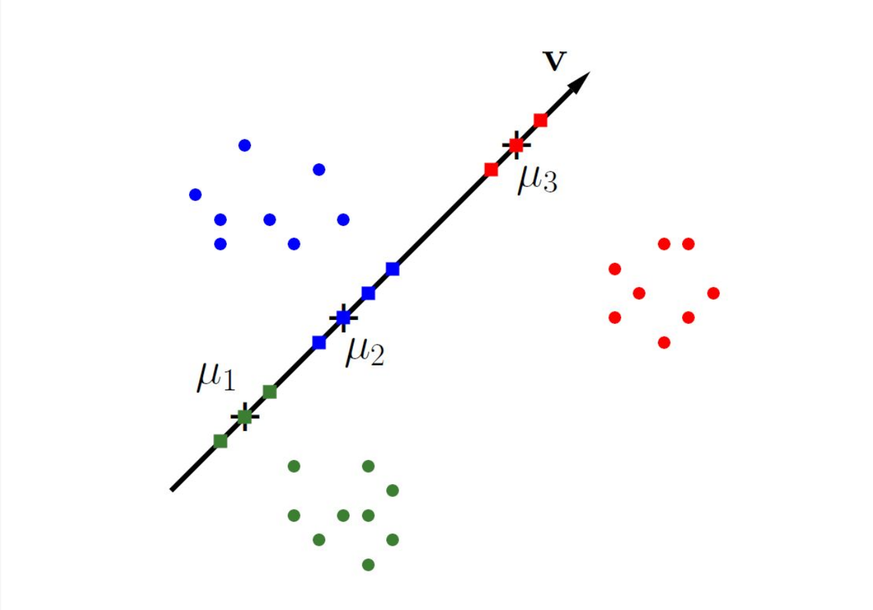
$$\mathbf{S}_j = \sum_{\mathbf{x} \in C_j} (\mathbf{x} - \mathbf{m}_j)(\mathbf{x} - \mathbf{m}_j)^T$$

and Sw = ΣSj is the total within-class scatter matrix.

Mathematical Derivation

To make the class centroids μj (in the projection space) as far from each other as possible, we can just maximize the variance of the centroids set {μ1, . . . , μk}:

$$\sum_{j=1}^{c}(\mu_j-\bar{\mu})^2=\frac{1}{c}\sum_{j<\ell}(\mu_j-\mu_\ell)^2, \qquad \text{where} \quad \bar{\mu}=\frac{1}{c}\sum_{j=1}^{c}\mu_j$$



Mathematical Derivation

We actually use a weighted mean of the projected centroids to define the between-class scatter:

$$\sum_{j=1}^{c} n_j(\mu_j - \mu)^2, \qquad \text{where} \quad \mu = \frac{1}{n}\sum_{j=1}^{c} n_j\mu_j$$

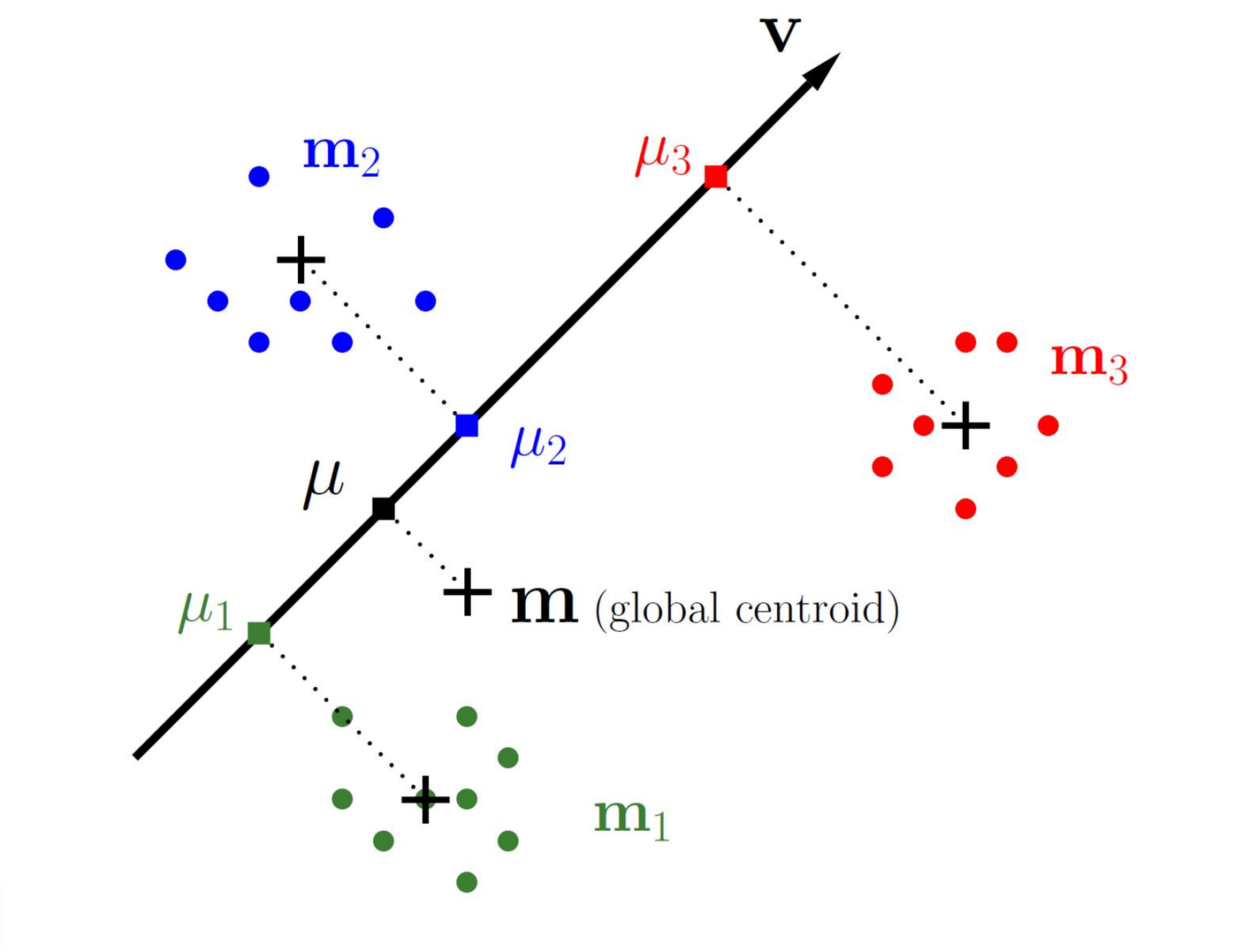
because the weighted mean (μ) is the projection of the global centroid (m) of the training data onto v:

$$\mathbf{v}^T\mathbf{m} = \mathbf{v}^T\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\right) = \mathbf{v}^T\left(\frac{1}{n}\sum_{j=1}^{c} n_j\mathbf{m}_j\right) = \frac{1}{n}\sum_{j=1}^{c} n_j\mu_j = \mu.$$

In contrast, the simple mean does not have such a geometric interpretation:

$$\bar{\mu} = \frac{1}{c}\sum_{j=1}^{c}\mu_j = \frac{1}{c}\sum_{j=1}^{c}\mathbf{v}^T\mathbf{m}_j = \mathbf{v}^T\left(\frac{1}{c}\sum_{j=1}^{c}\mathbf{m}_j\right)$$

Mathematical Derivation



Mathematical Derivation

We simplify the between-class scatter (in the v space) as follows:

$$\begin{aligned}\sum_{j=1}^{c} n_j(\mu_j - \mu)^2 &= \sum n_j(\mathbf{v}^T(\mathbf{m}_j - \mathbf{m}))^2 \\ &= \sum n_j\, \mathbf{v}^T(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T\mathbf{v} \\ &= \mathbf{v}^T\left(\sum n_j(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T\right)\mathbf{v} \\ &= \mathbf{v}^T\mathbf{S}_b\mathbf{v}.\end{aligned}$$

We have thus arrived at the same kind of problem

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1} \frac{\mathbf{v}^T\mathbf{S}_b\mathbf{v}}{\mathbf{v}^T\mathbf{S}_w\mathbf{v}} \quad \longleftarrow \quad \frac{\sum n_j(\mu_j - \mu)^2}{\sum s_j^2}$$

Mathematical Derivation

Remark. When c = 2, it can be verified that

$$\sum_{j=1}^{2} n_j(\mu_j - \mu)^2 = \frac{n_1 n_2}{n}(\mu_1 - \mu_2)^2, \qquad \text{where} \quad \mu = \frac{1}{n}(n_1\mu_1 + n_2\mu_2)$$

$$\sum_{j=1}^{2} n_j(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T = \frac{n_1 n_2}{n}(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T, \ \mathbf{m} = \frac{1}{n}(n_1\mathbf{m}_1 + n_2\mathbf{m}_2)$$

This shows that when there are only two classes, the weighted definitions are just a scalar multiple of the unweighted definitions.

Therefore, the multiclass LDA is a natural generalization of the two-class LDA

Computing

The solution is given by the largest eigenvector of Sw^-1 Sb (when Sw is nonsingular):

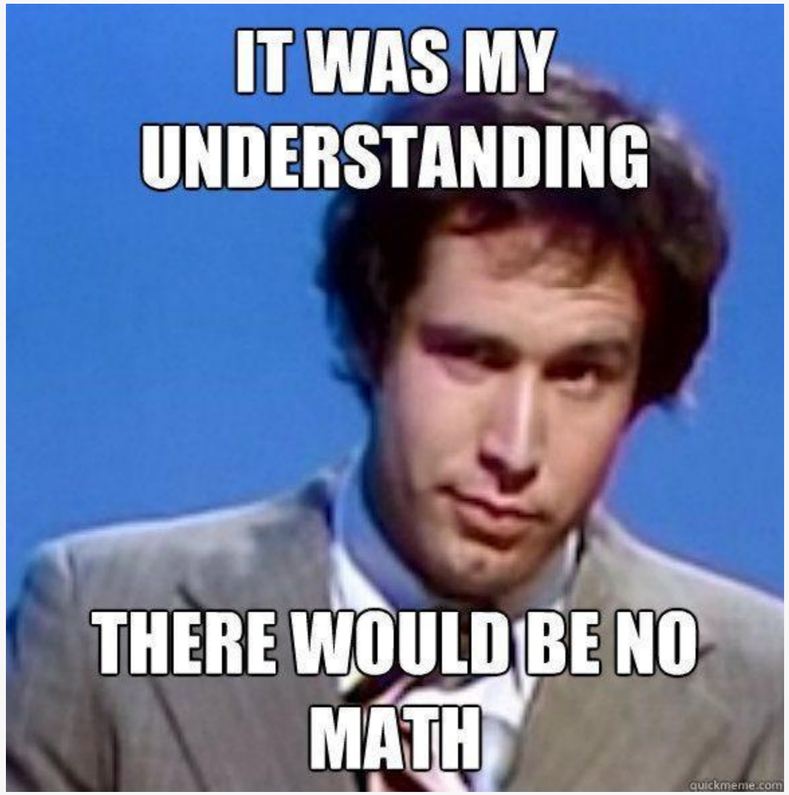
$$\mathbf{S}_w^{-1}\mathbf{S}_b\mathbf{v}_1 = \lambda_1\mathbf{v}_1.$$

However, the formula $\mathbf{v}_1 \propto \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$ is no longer valid:

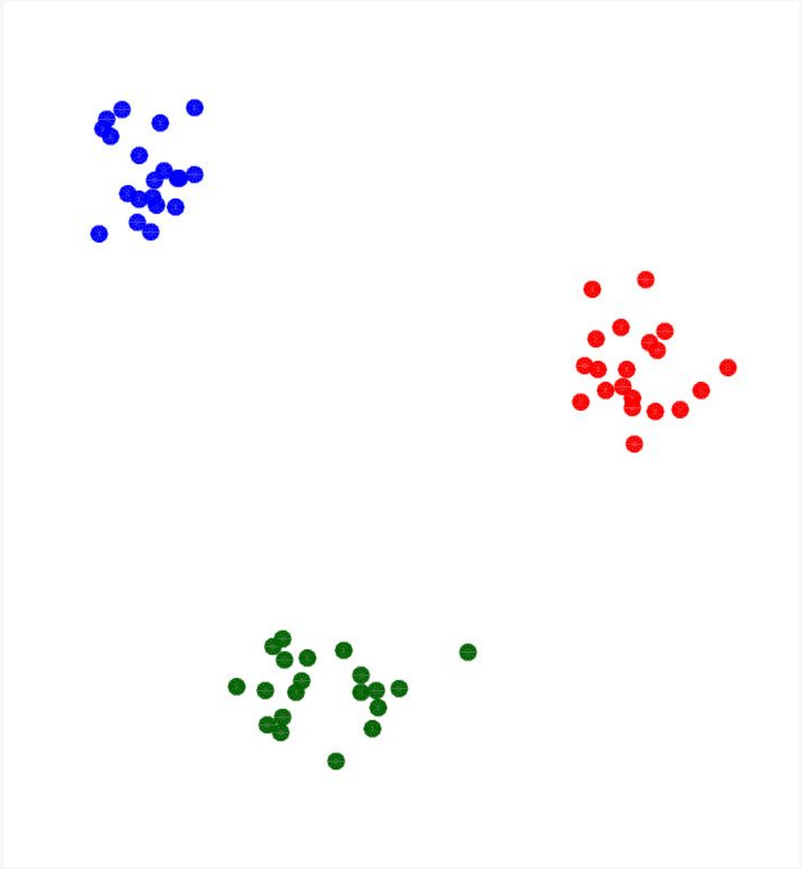
$$\lambda_1\mathbf{v}_1 = \mathbf{S}_w^{-1}\mathbf{S}_b\mathbf{v}_1 = \mathbf{S}_w^{-1}\sum_j n_j(\mathbf{m}_j - \mathbf{m})\underbrace{(\mathbf{m}_j - \mathbf{m})^T\mathbf{v}_1}_{\text{scalar}}$$

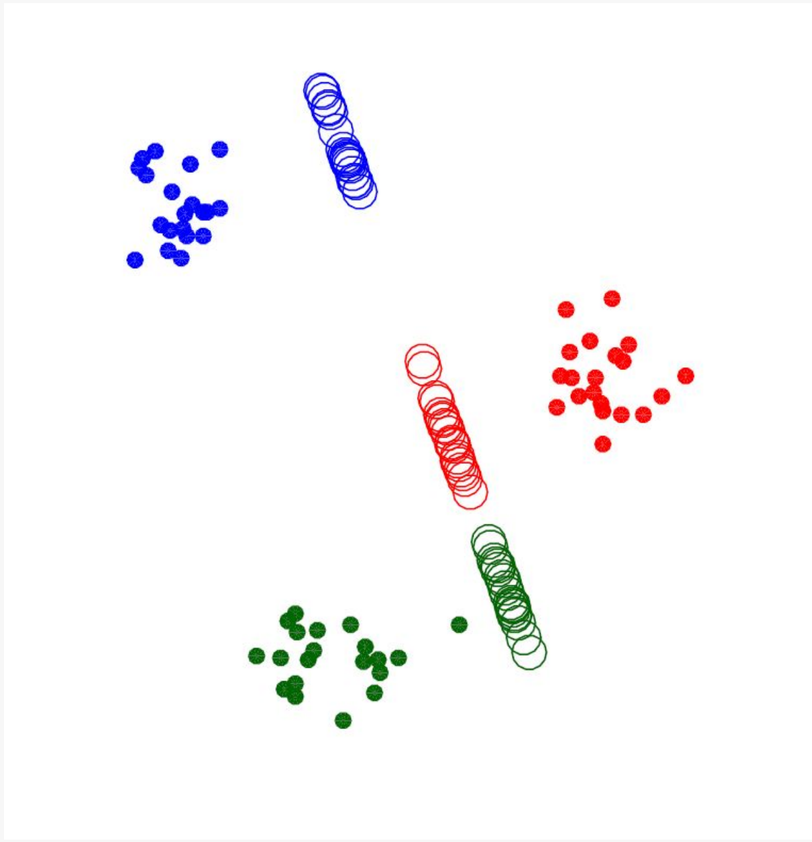
So we have to find v1 by solving a generalized eigenvalue problem:

$$\mathbf{S}_b\mathbf{v}_1 = \lambda_1\mathbf{S}_w\mathbf{v}_1.$$

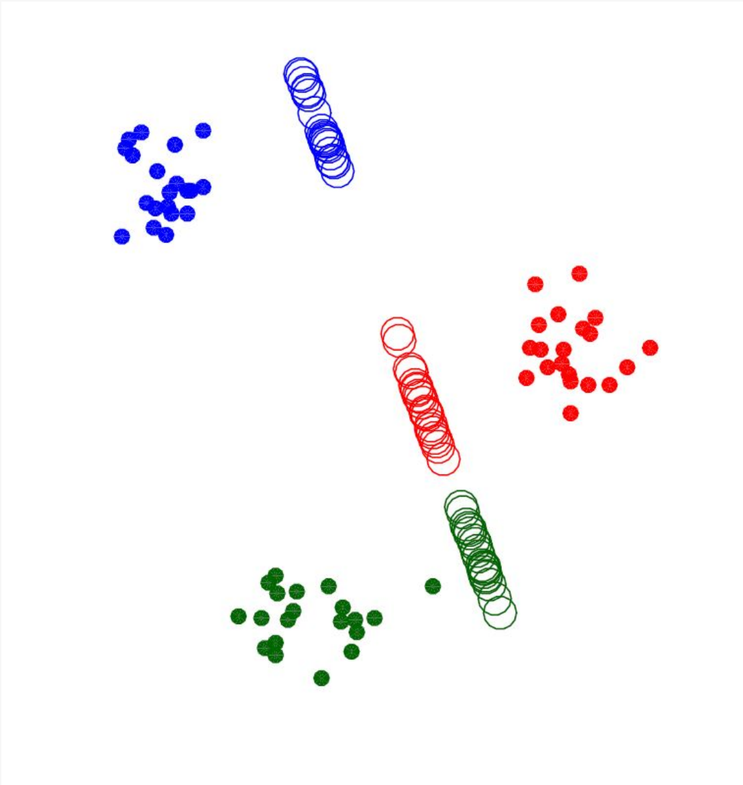

IT WAS MY UNDERSTANDING
THERE WOULD BE NO MATH
quickmeme.com

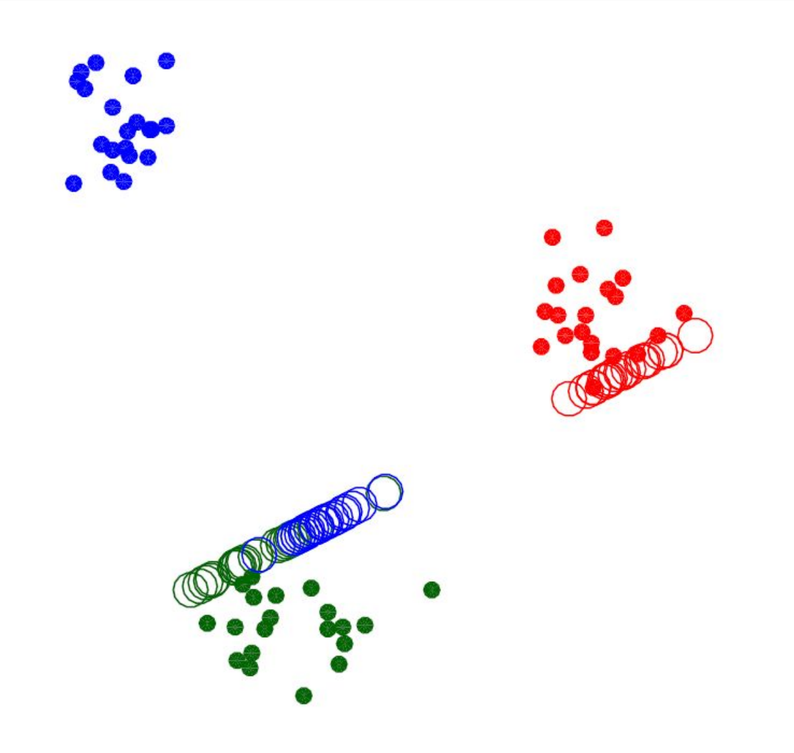
Simulation





What about the second eigenvector v2?





Multiclass LDA Algorithm

Input: Training data **X** (with c classes)
Output: At most c−1 discriminatory directions and projections of X onto them.

1. Compute $$\mathbf{S}_w = \sum_{j=1}^{c} \sum_{\mathbf{x} \in C_j} (\mathbf{x} - \mathbf{m}_j)(\mathbf{x} - \mathbf{m}_j)^T, \quad \mathbf{S}_b = \sum_{j=1}^{c} n_j (\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T.$$

2. Solve the generalized eigenvalue problem Sbv = lambdaSwv to find all nonzero eigenvectors Vk = [v1, . . . , vk] (for some k <= c − 1)

3. Project the data X onto them Y = X · Vk

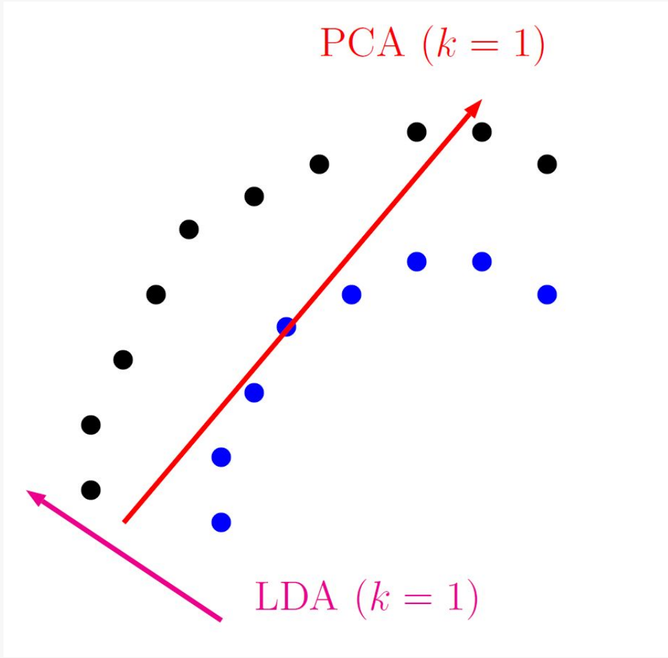
Comparison between PCA and LDA

Criteria	PCA	LDA
Use Labels?	No (Unsupervised)	Yes (Supervised)
Criterion	Variance	Discrimination
Dimensions (k)	any	k<=(c-1)
Computing	SVD	Generalized eigenvectors
Non Linear Boundary	Can handle	Cannot Handle

Comparison between PCA and LDA

In the case of nonlinear separation between the classes, PCA often works better than LDA as the latter can only find at most c−1 directions (which are insufficient to preserve all the discriminatory information in the training data).

- LDA with k = 1: does not work well
- PCA with k = 1: does not work well
- PCA with k = 2: preserves all the nonlinear separation which can be handled by nonlinear classifiers.



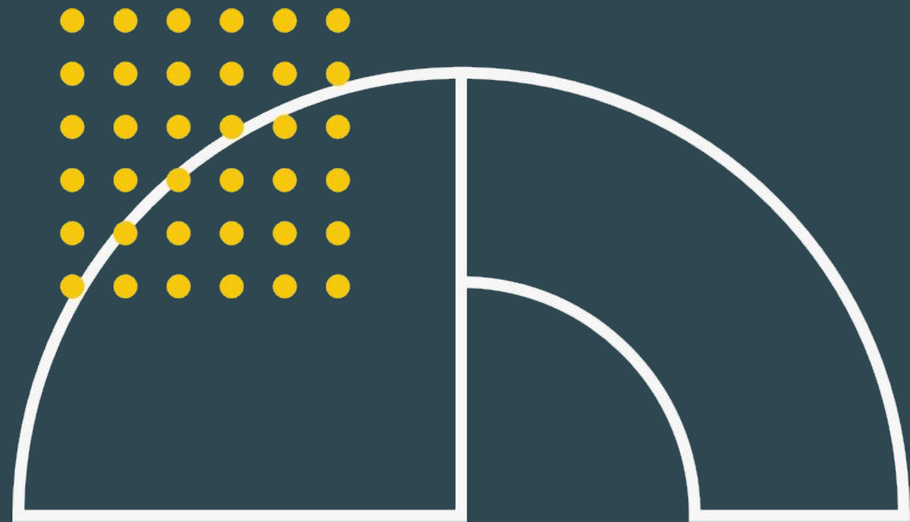
Assumptions in LDA

1. The data for each class is **Gaussian (normally distributed)**
2. All classes share a **common covariance matrix** (covariances are equal across classes)
3. The within-class scatter matrix must be **invertible** (non-singular), which requires enough samples compared to the dimensionality.

Drawbacks of LDA

1. If the actual class covariances differ or data is **not Gaussian**, LDA may perform poorly.
2. In **high-dimensional settings**, the within-class scatter matrix can become singular, causing LDA to fail or require regularization/PCA preprocessing.
3. LDA can only produce **at most c–1 discriminatory directions** for c classes, potentially missing complex patterns in the data.
4. LDA only works for **linear decision boundaries** and is not effective for nonlinear class separation

THANK YOU



Contribution

G1

(2201AI02, Akash Sinha)	4.182%
(2201AI04, Ammar Ahmad)	4.166%
(2201AI51, Mridul Kumar)	4.166%
(2201AI54, Aman Vaibhav Jha)	4.166%
(2201CS07, Aditya Chauhan)	4.166%
(2201CS08, Aditya Yadav)	4.166%
(2201CS11, Akhand Singh)	4.166%
(2201CS15, Anchal Dubey)	4.166%
(2201CS16, Animesh Tripathy)	4.166%
(2201CS45, Mayur Borse)	4.166%
(2201CS54, Prakhar Shukla)	4.166%
(2201CS94, Anirudh D Bhat)	4.166%

G2

(2201AI19, Koppolu Buddha Bhavan)	4.166%
(2201CS38, Komarabatini Vishwa Chaitanya)	4.166%
(2201CS12, Alam sai Bharath)	4.166%
(2201CS71, Sripat Surya Teja)	4.166%
(2201CS29, Gurram Nikhil)	4.166%
(2201AI44, Sadupati Vinay)	4.166%
(2201AI08, Akshita reddy)	4.166%
(2201AI43, Haswanthi)	4.166%
(2201CS69, Sri yash)	4.166%
(2201AI49, Pranav sai)	4.166%
(2201AI37, sowmya)	4.166%
(2201AI36, SOMIL AGGARWAL)	4.166%


OUR MARX