NAIVE BAYES

BAYES THEOREM

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

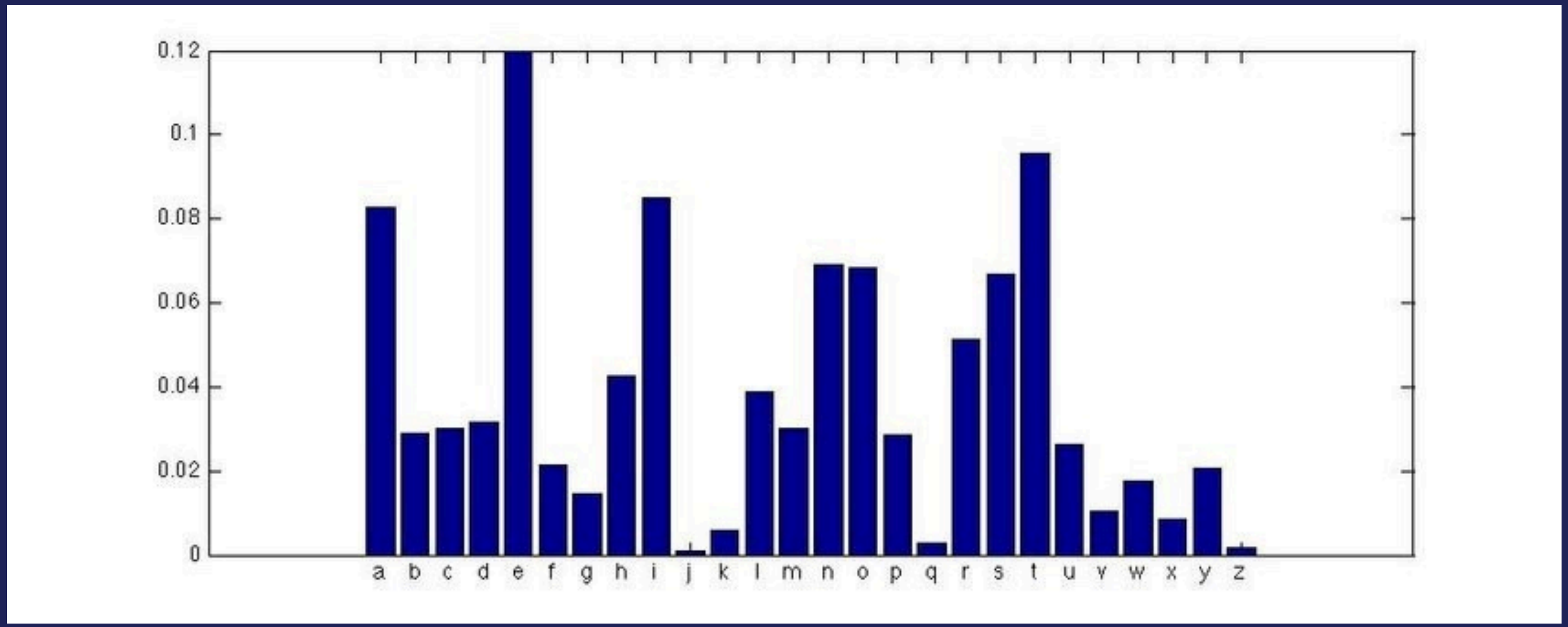
- P(A) -> Probability for A
- P(B) -> Probability for B
- P(B|A) -> Probability for B to occur given that A has occured
- P(A|B) -> Probability for A to occur given that B has occured

TYPES OF NAIVE BAYES

- **MULTINOMIAL NAIVE BAYES**
- **GAUSSIAN NAIVE BAYES**

MULTINOMIAL NAIVE BAYES

The Multinomial Naive Bayes classifier is suitable for classification with discrete features e.g., spam/not spam text classification.



SPAM/NOT SPAM

MESSAGE: DEAR FRIEND

Mathematically, you want to determine two probabilities.

- p(Not Spam | Dear Friend)
- p(Spam | Dear Friend)

If p(Not Spam | Dear Friend) > p(Spam | Dear Friend) then the message is Not Spam else Spam.

SPAM/NOT SPAM
TRAINING PHASE

Not Spam Dataset:
1) Hi Friend
2) Dear
3) Friend, Lunch!
4) Give Money!

Spam Dataset:
1) Money Money Friend
2) Dear Take Money

SPAM/NOT SPAM

TRAINING PHASE

$P(\text{Not Spam}) = 4/6 = 0.66$, $P(\text{Spam}) = 2/6 = 0.33$

$P(\text{Dear} \mid \text{Not Spam}) = 1/7$, $P(\text{Dear} \mid \text{Spam}) = 1/6$

$P(\text{Friend} \mid \text{Not Spam}) = 2/7$, $P(\text{Friend} \mid \text{Spam}) = 1/6$

$P(\text{Money} \mid \text{Not Spam}) = 1/7$, $P(\text{Money} \mid \text{Spam}) = 3/6$

$P(\text{Lunch} \mid \text{Not Spam}) = 1/7$, $P(\text{Lunch} \mid \text{Spam}) = 0$

SPAM/NOT SPAM
TESTING PHASE

Message: Dear Friend

P(Not Spam | Dear Friend) = ?
Using Bayes Theorem,

= P(Dear Friend | Not Spam) * P(Not Spam) / P(Dear Friend)

P(Dear Friend) = 1 (As the given message is Dear Friend)

= P(Dear | Not Spam) * P(Friend | Not Spam) * P(Not Spam)
= 1/7 * 2/7 * 4/6 = 0.0272

Similarly, P(Spam | Dear Friend) = 1/6 * 1/6 * 2/6 = 0.0092 (approx)

SPAM/NOT SPAM

Now try for the message
“Lunch Money Money”

Go to slido.com
Use the code: 2909 400

SPAM/NOT SPAM

SOLUTION:

P(NS | Lunch Money Money)
= P(Lunch | NS) * P(Money | NS)^2 * P(NS)
= 1/7 * (1/7)^2 * 4/6 > 0

P(S | Lunch Money Money)
= P(Lunch | S) * P(Money | S)^2 * P(S)
= 0 * (3/6)^2 * 2/6 = 0

So it is classified as Not Spam.
But there is problem.

LAPLACE SMOOTHING

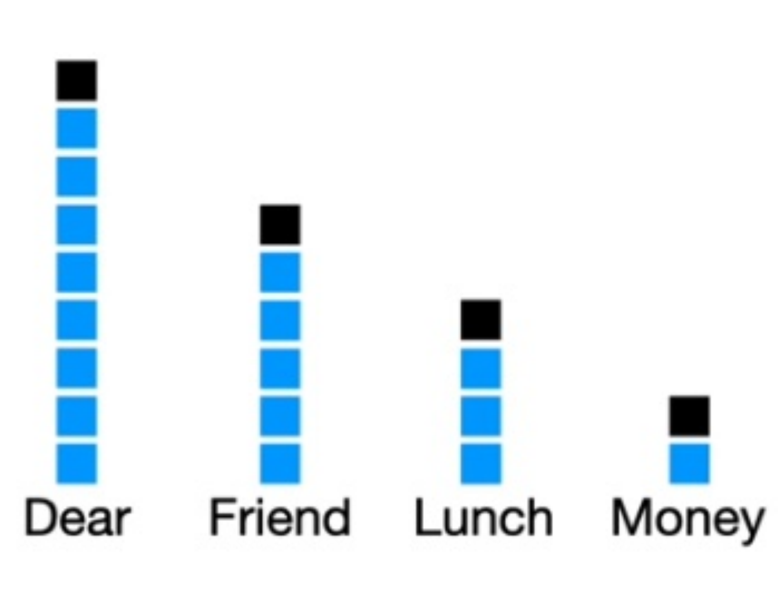
$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class})}{\mathrm{N_{class}}}$$

class ∈ {Positive, Negative}

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class}) + 1}{\mathrm{N_{class}} + \mathrm{V}}$$

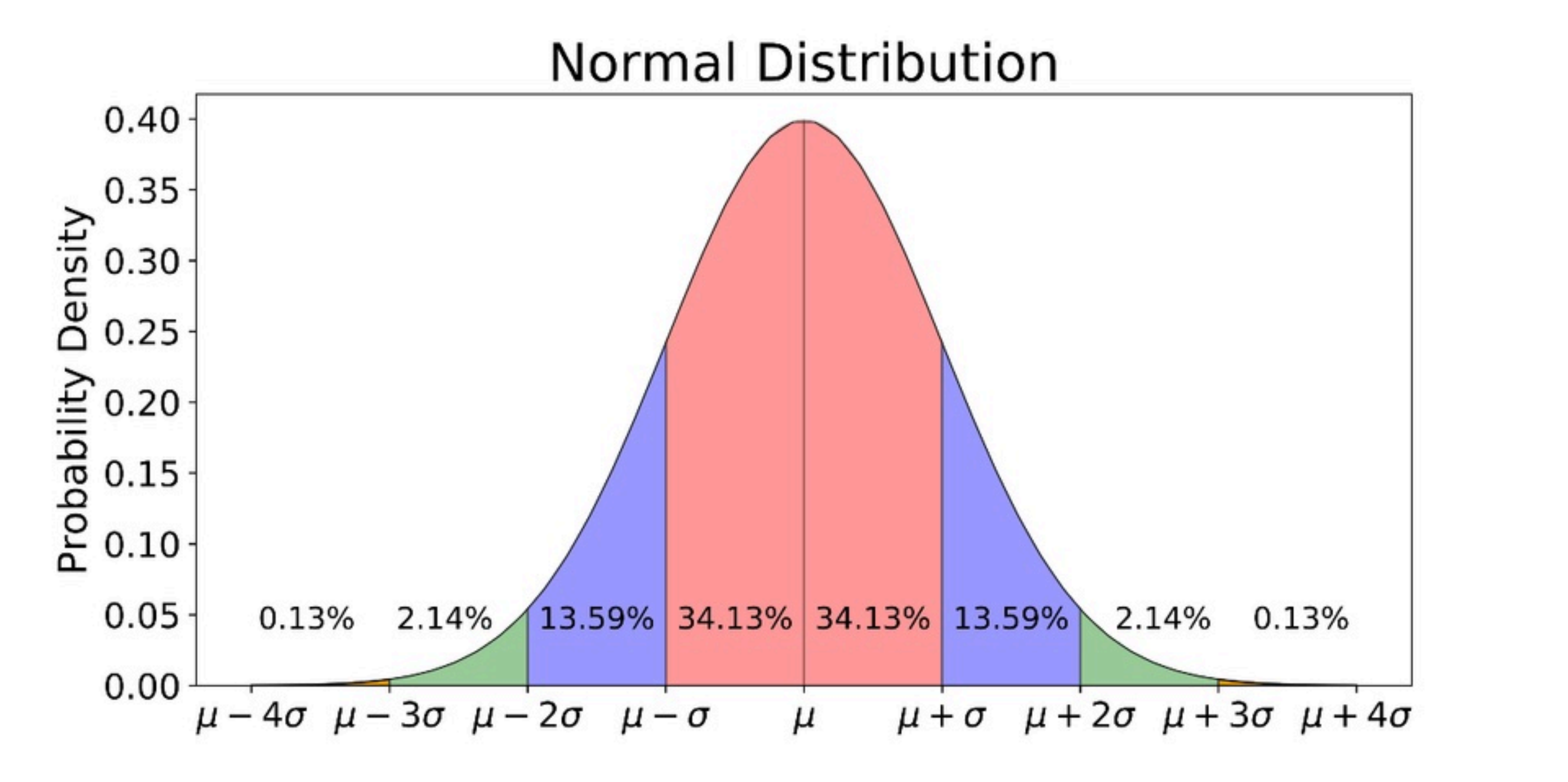
N_{class} = frequency of all words in class

V = number of unique words in vocabulary



GAUSSIAN NAIVE BAYES

Gaussian naive bayes is appropriate for continuous data, which are theoretically dependent of Gaussian distribution methods.



NORMAL DISTRIBUTION

Normal Distribution Formula

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$\mu =$ mean of x

$\sigma =$ standard deviation of x

$\pi \approx 3.14159 \ldots$

$e \approx 2.71828 \ldots$

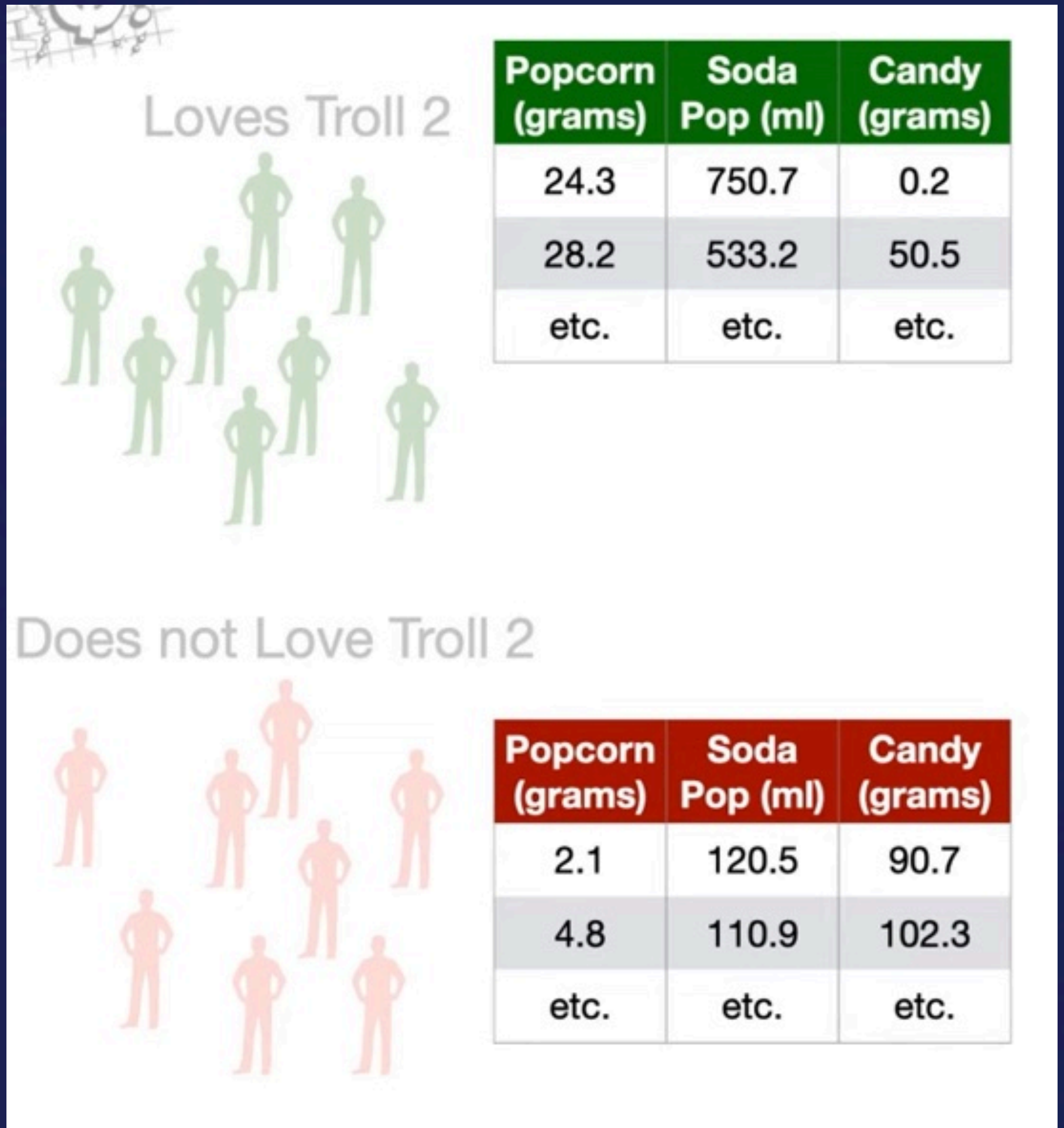
Mean:

Average of the given numbers.

Standard Deviation:

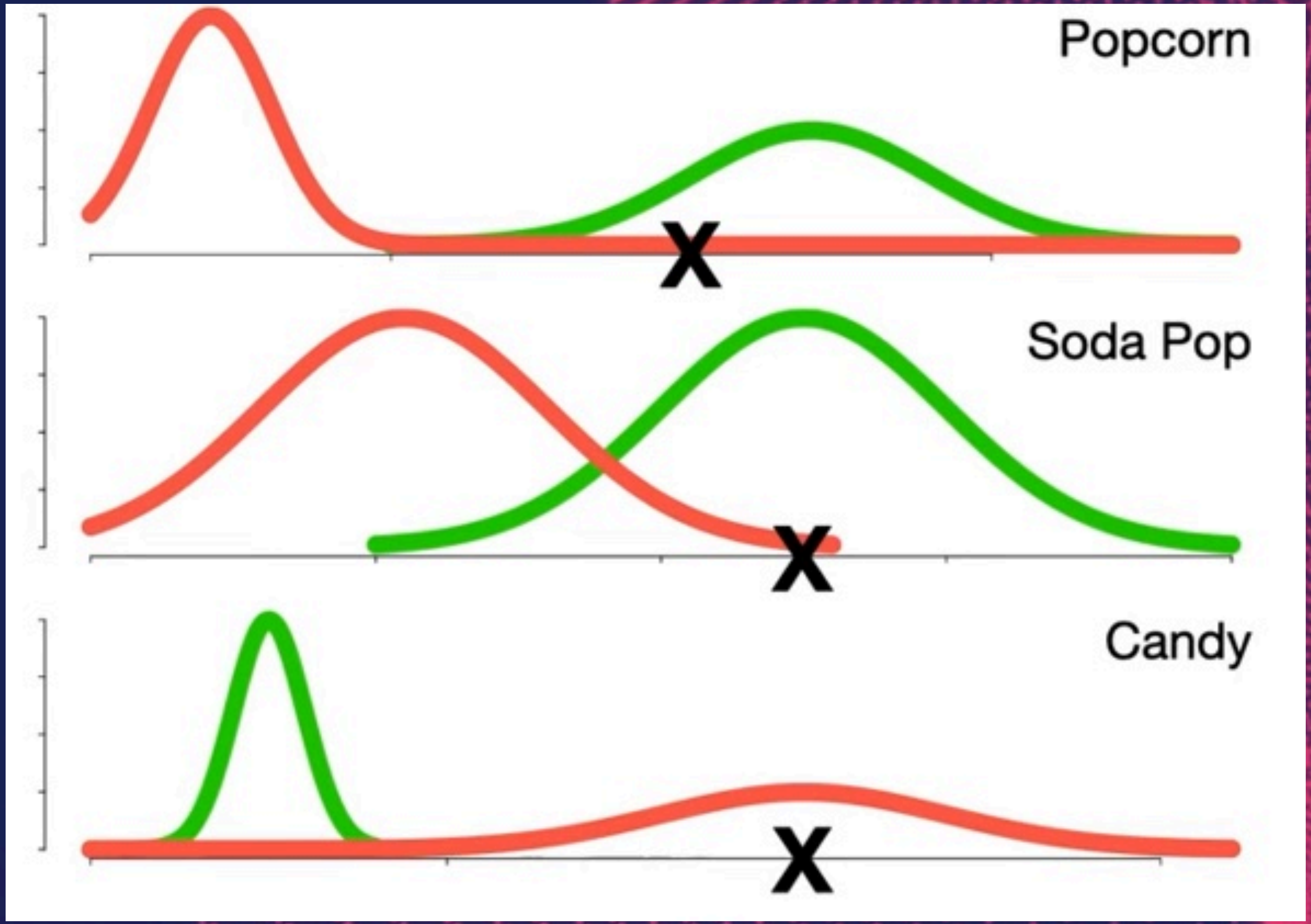
A number which tells how dispersed the data is around the mean.

LIKE / DOES NOT LIKE



Popcorn (grams)	Soda Pop (ml)	Candy (grams)
24.3	750.7	0.2
28.2	533.2	50.5
etc.	etc.	etc.

Popcorn (grams)	Soda Pop (ml)	Candy (grams)
2.1	120.5	90.7
4.8	110.9	102.3
etc.	etc.	etc.



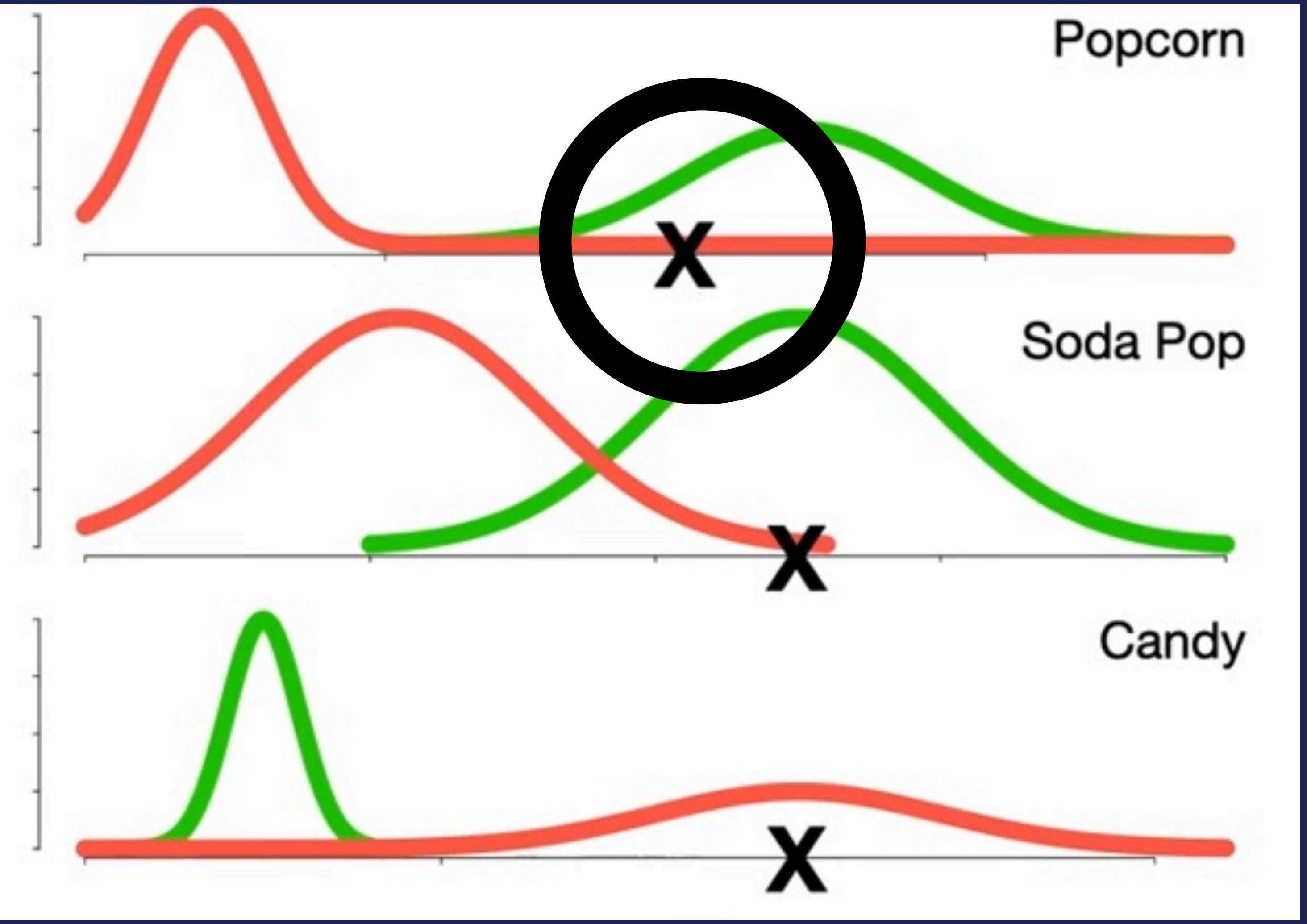
LIKE / DOES NOT LIKE

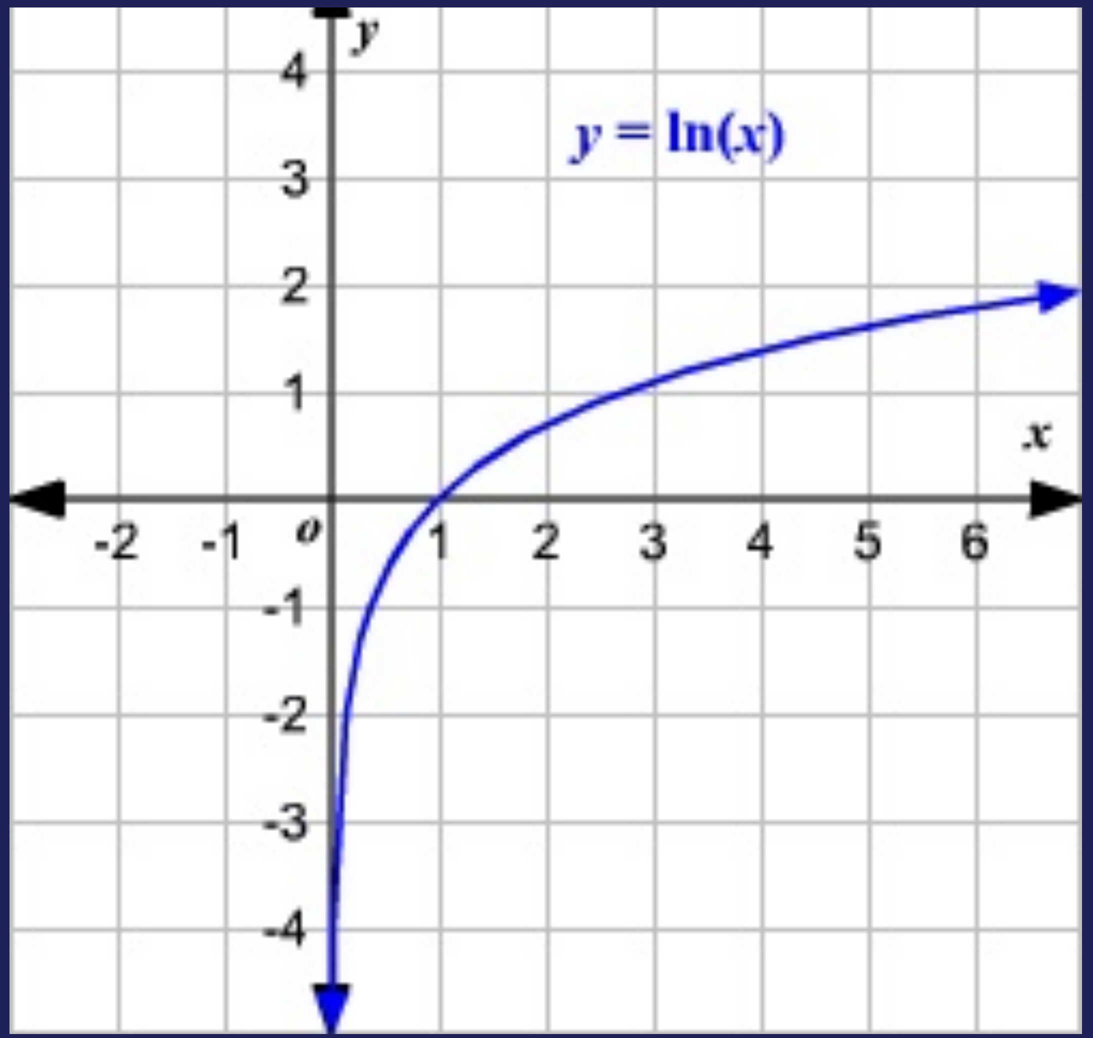
P(likes | sample) =
P(popcorn | sample) * P(soda | sample) * P(candy | sample) * P(likes)

P(not likes | sample) =
P(popcorn | sample) * P(soda | sample) * P(candy | sample) * P(not likes)

The values of probabilities is very small and leads to underflow.

LOG USAGE / UNDERFLOW





Low value leads to underflow

Log tends to magnify low values

WHY IT IS NAIVE?

It is naive because it assumes independence between the features unlike word embeddings.

In multinomial naive bayes the message “Dear Friend” will have the same score as “Friend Dear”

CODE IMPLEMENTATION

Google Colab

google.com

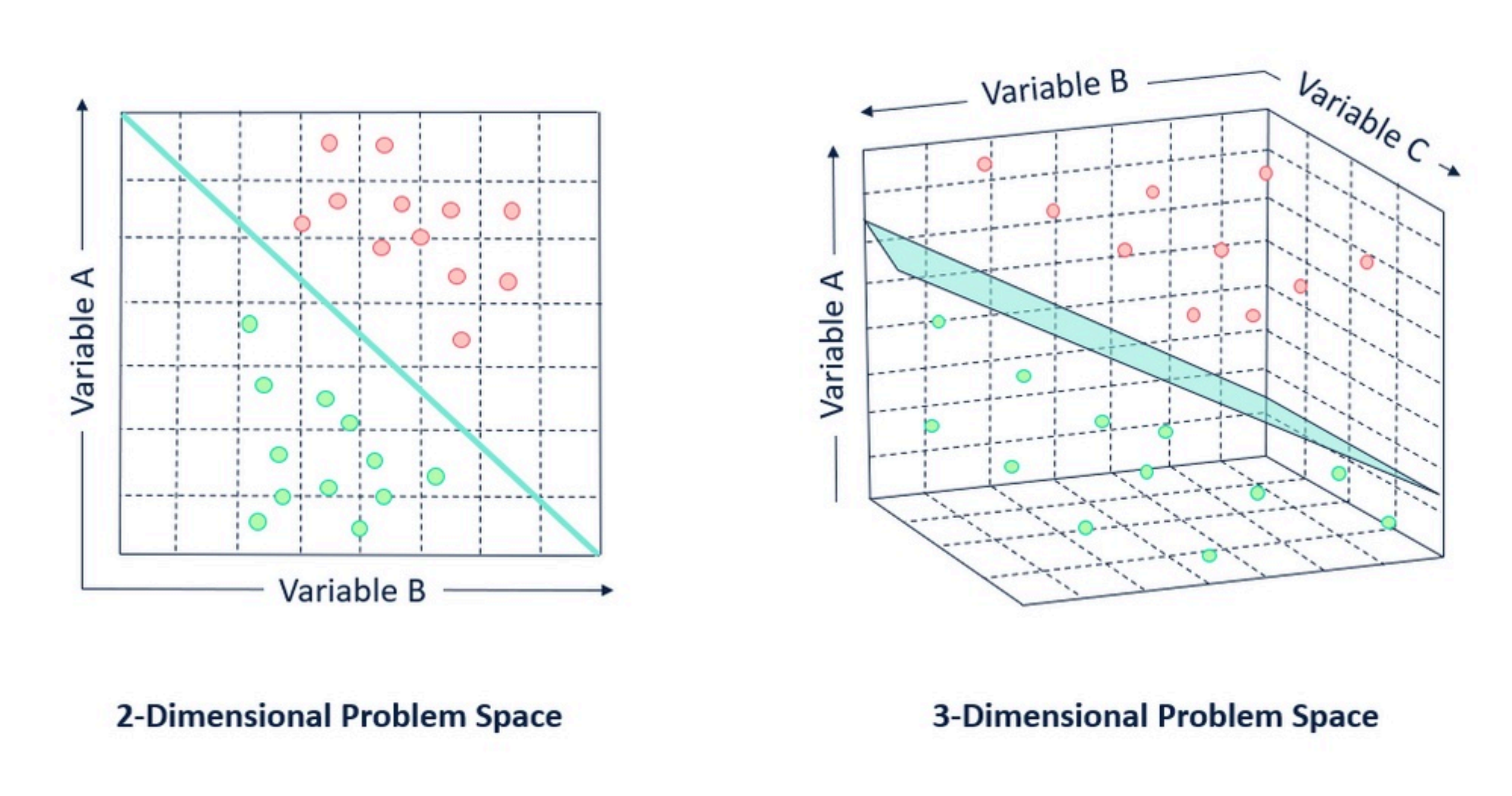
SUPPORT VECTOR MACHINES

SUPPORT VECTOR MACHINES

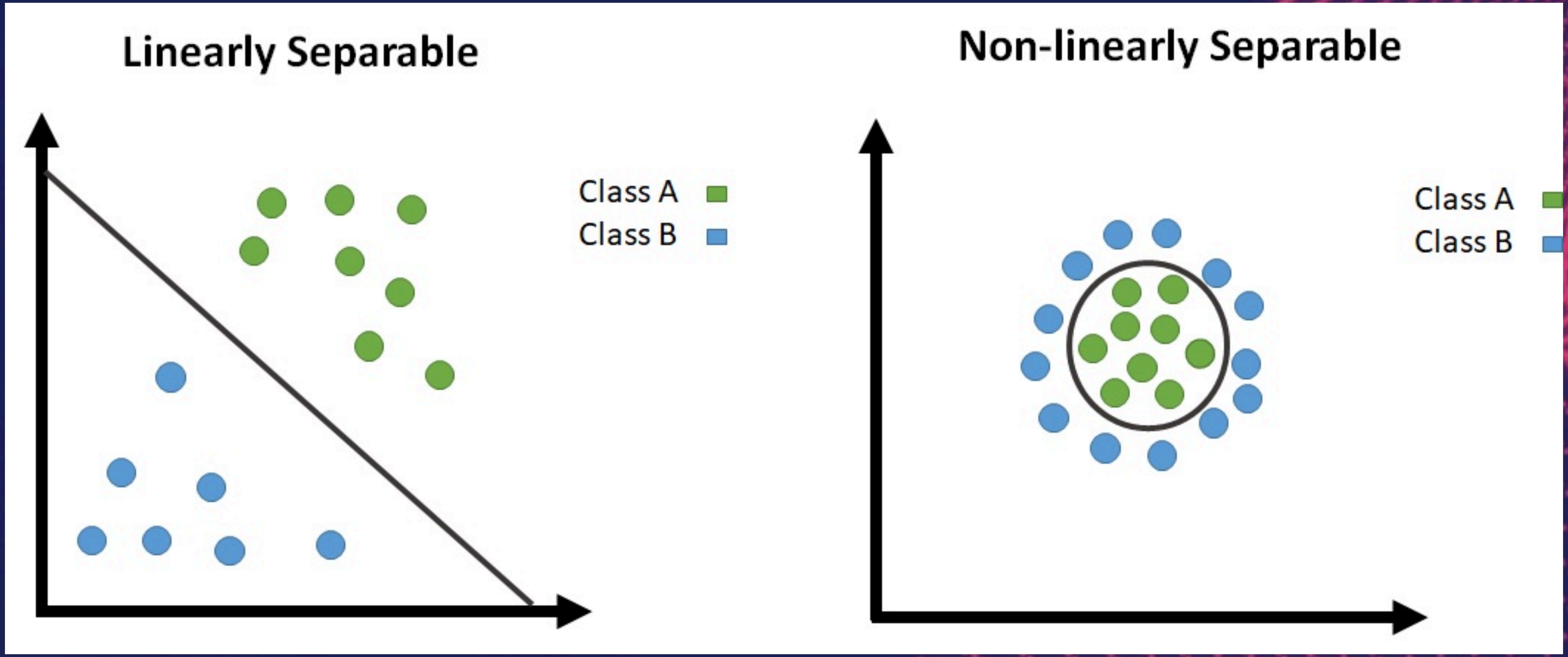
Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks.

HYPERPLANES IN DIFFERENT DIMENSIONS

Dimension of Hyper Plane depends on the number of features in the input data



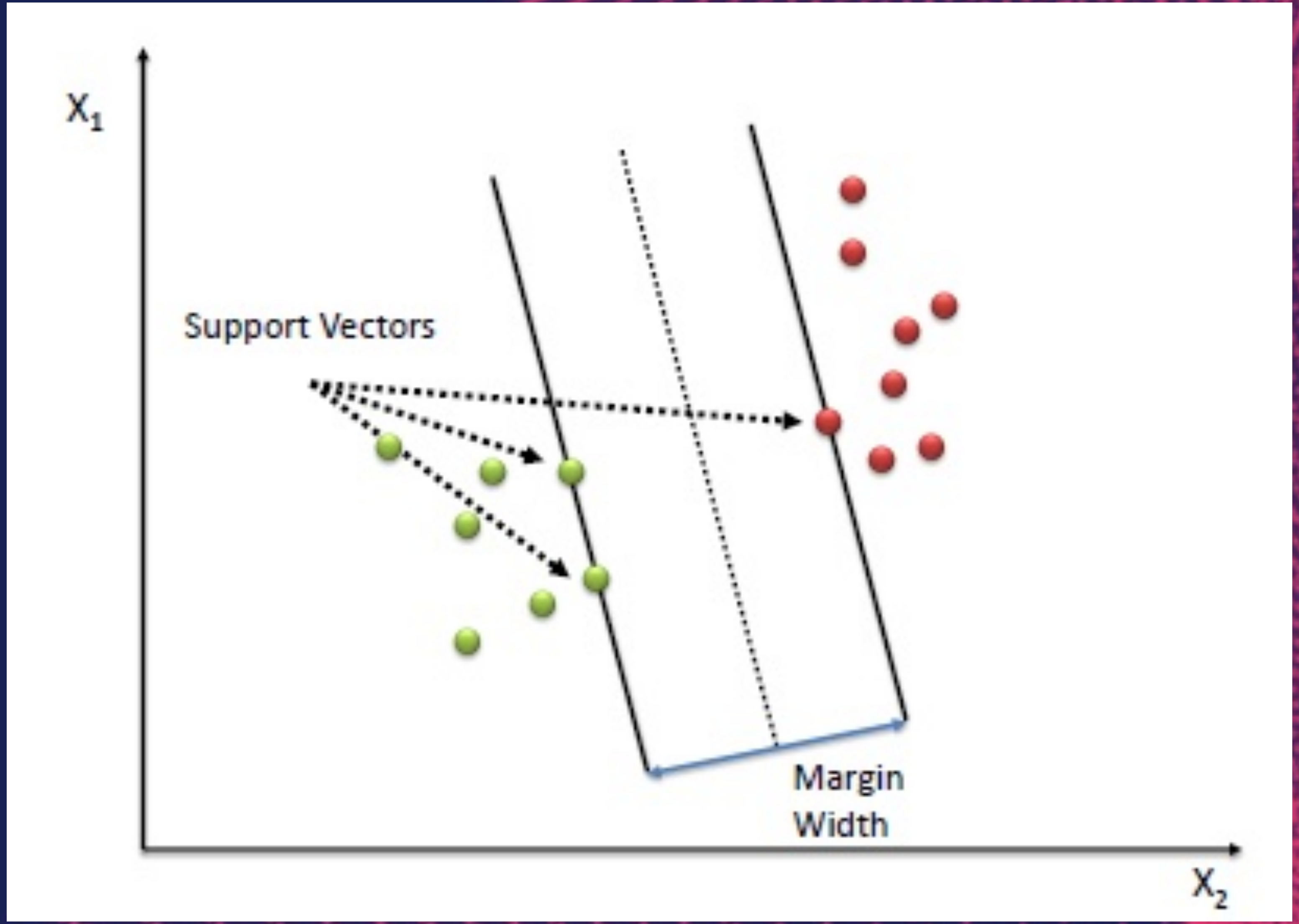
LINEAR AND NON LINEAR DATA



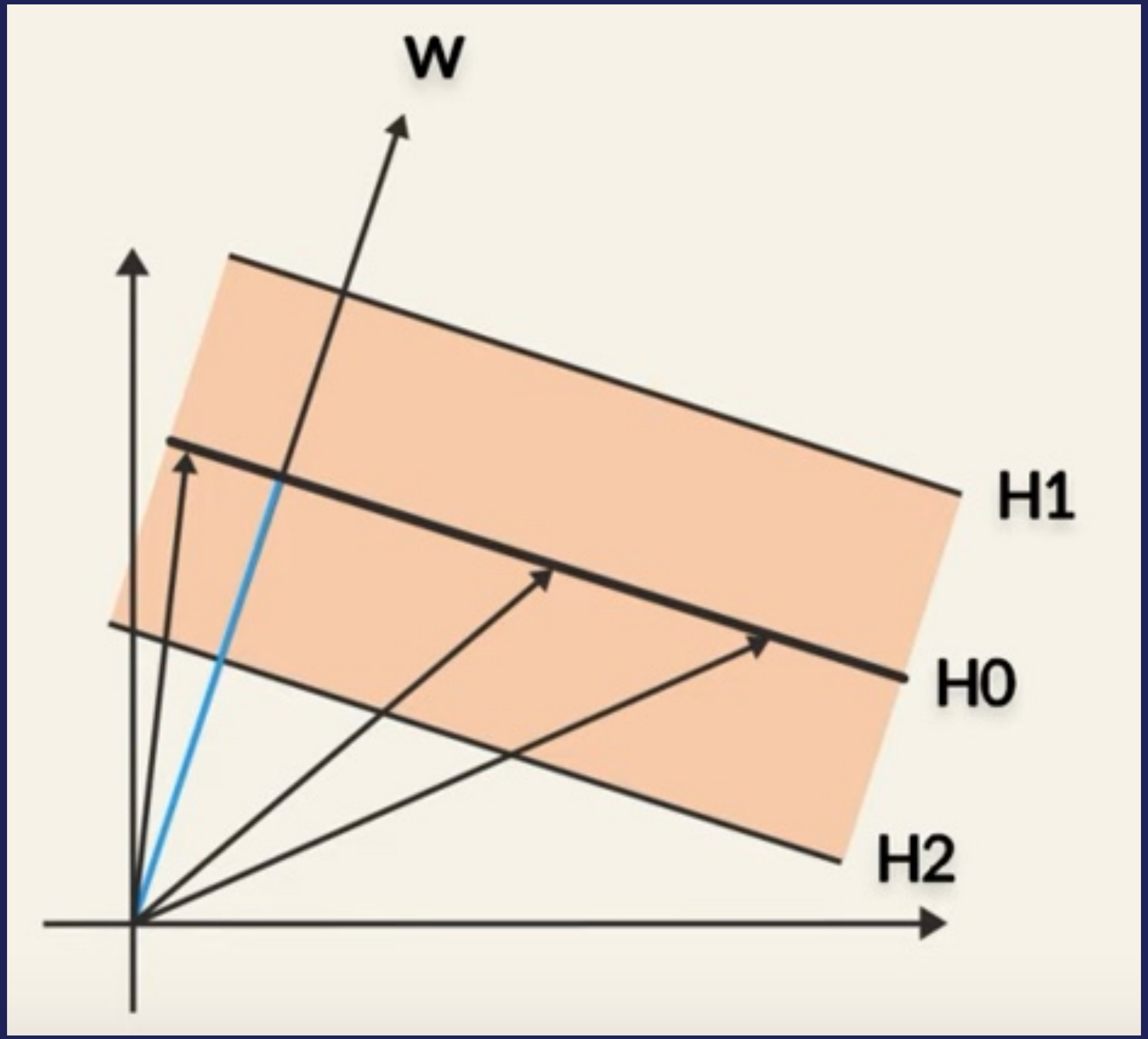
DEFINITIONS

SUPPORT VECTORS: Points that are closer to hyperplane. Used to find the HP.

MARGIN: Distance between the hyperplane and the support vectors. Larger the margin better the SVM.



LINEAR CASE: OPTIMAL SOLN



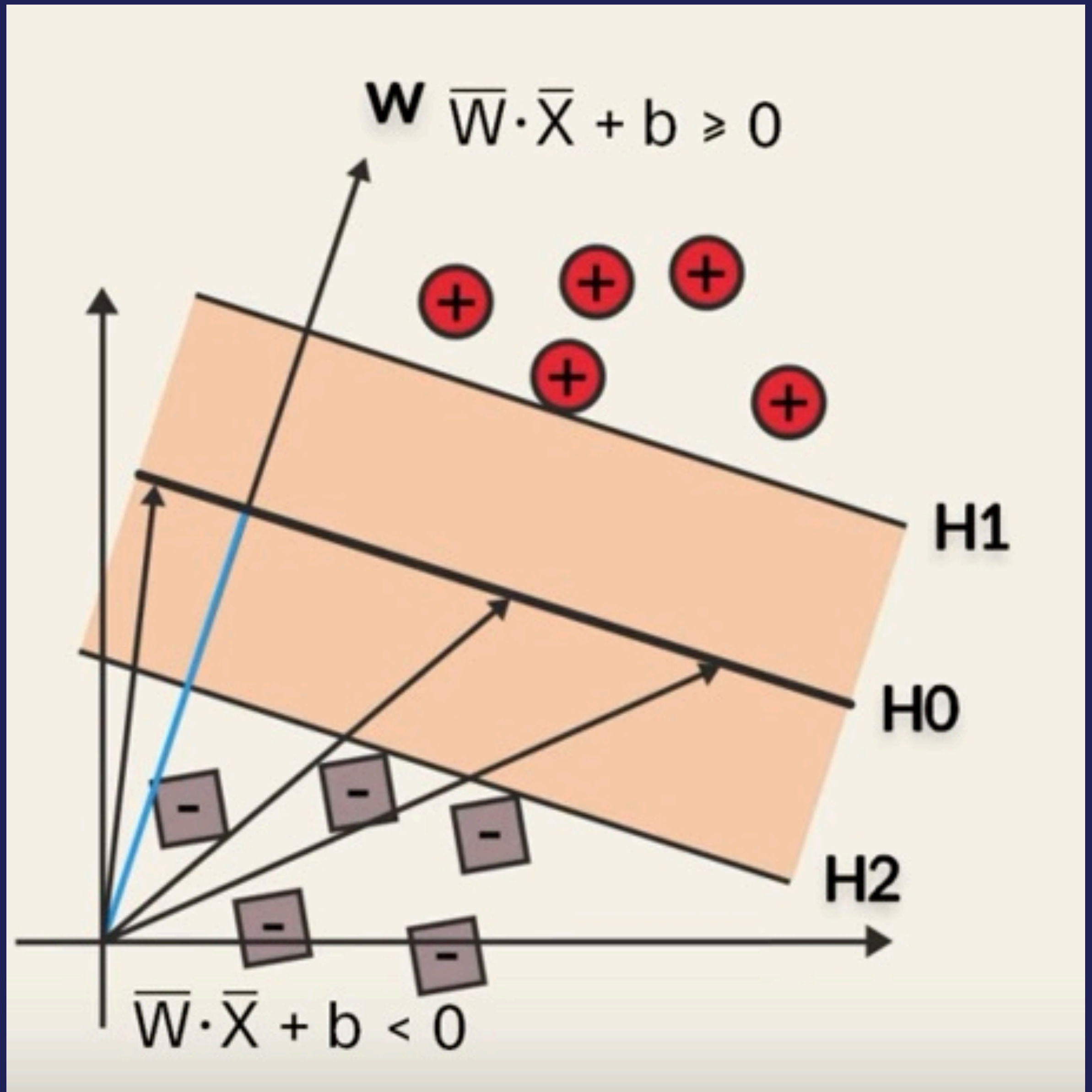
W: Vector perpendicular to hyperplane

X: Position vector of a data point

$$W * X = |W|\,|X| \cos \underline{\theta} = |W| * c$$

$$W * X + b = 0 \text{ (For point on the plane)}$$

LINEAR CASE: OPTIMAL SOLN



$W * X + b > 0$ (For a point above the plane)

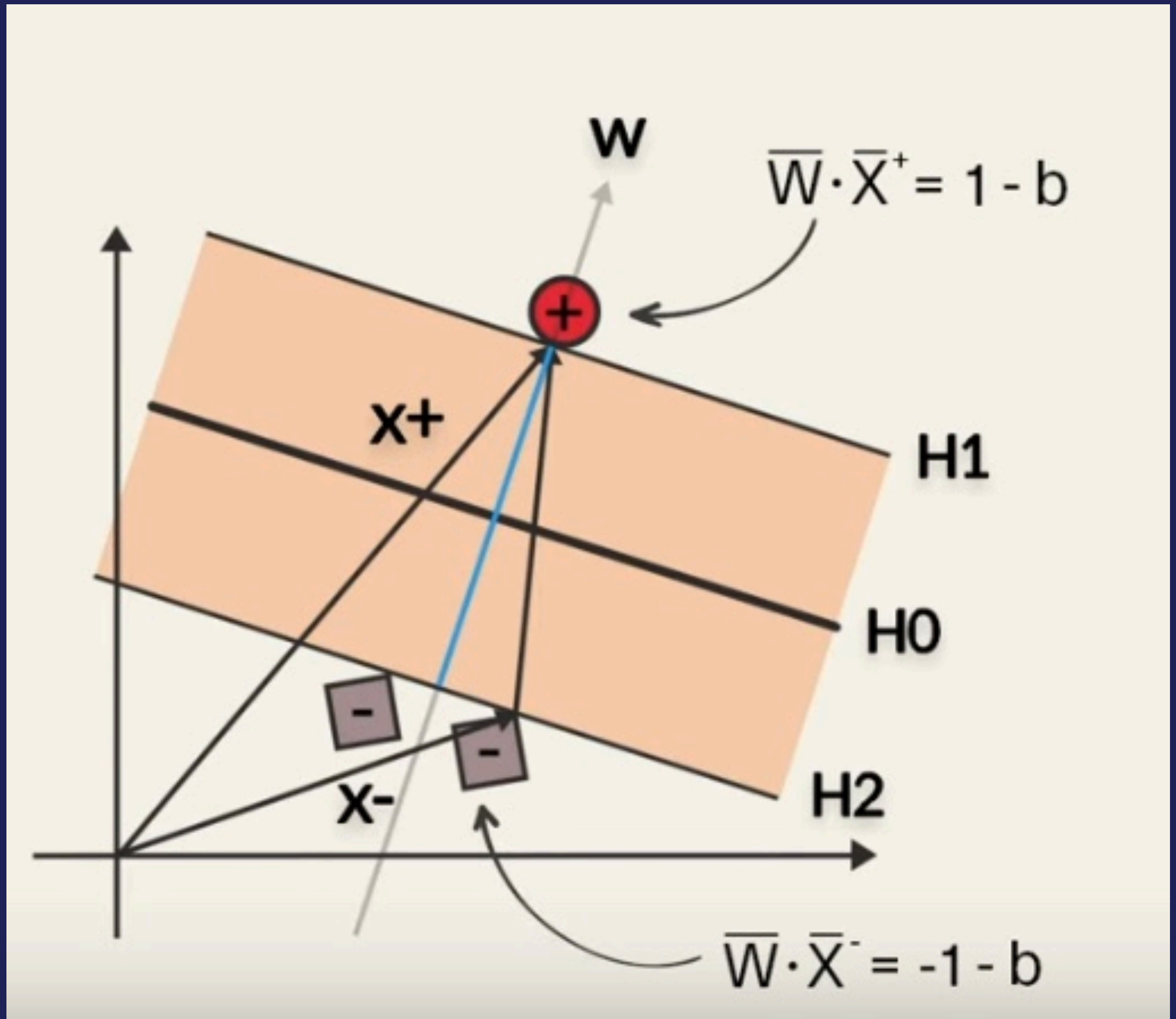
$W * X + b < 0$ (For a point below the plane)

$W * X + b = 1$ (H1 equation)

$W * X + b = -1$ (H2 equation)

(W and b are rescaled to make it 1)

MARGIN CALCULATION



margin = ((X+ - X-) * W) / |W|

= (W*X+ - W*X-)/|W|

=(1-b - (-1-b))/|W|

= 2/|W|

MARGIN MAXIMISATION AND CONSTRAINTS

We need to maximize the margin = 2/|W| i.e. we need to minimize |W|/2 with following constraints :

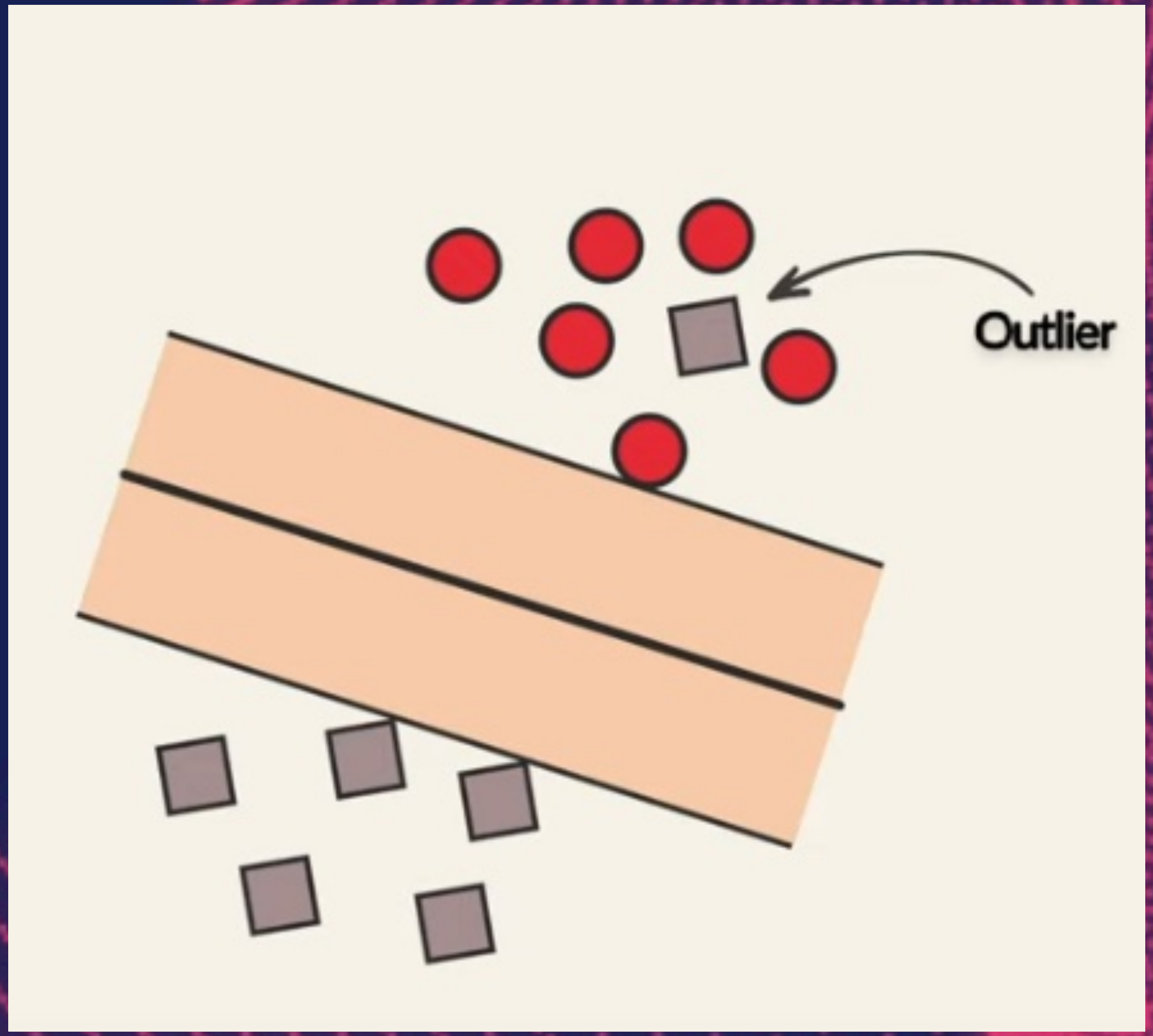
For a point above the hyperplane y = 1 and W*X + b > 1
For a point below the hyperplane y = -1 and W*X + b < -1

So the final constraint we get is y * (W*X + b) >= 1 for all data points.

ATTENDANCE QR

NOISE IN LINEAR DATA

If there is some noise in the linear data it will no longer be linear so we need to introduce something called SOFT MARGIN. Earlier margin is called HARD MARGIN.



SOFT MARGIN AND REGULARISATION

New Constraint: $y*(W*X + b) >= 1 - \zeta$

New Loss Function: $|W|^2 / 2 + c \Sigma \zeta$

ζ **here is** $\max(0, 1 - y*(W*X + b))$

Final Loss Function: $|W|^2 / 2 + c \Sigma \max(0, 1 - y*(W*X + b))$

EFFECT OF C

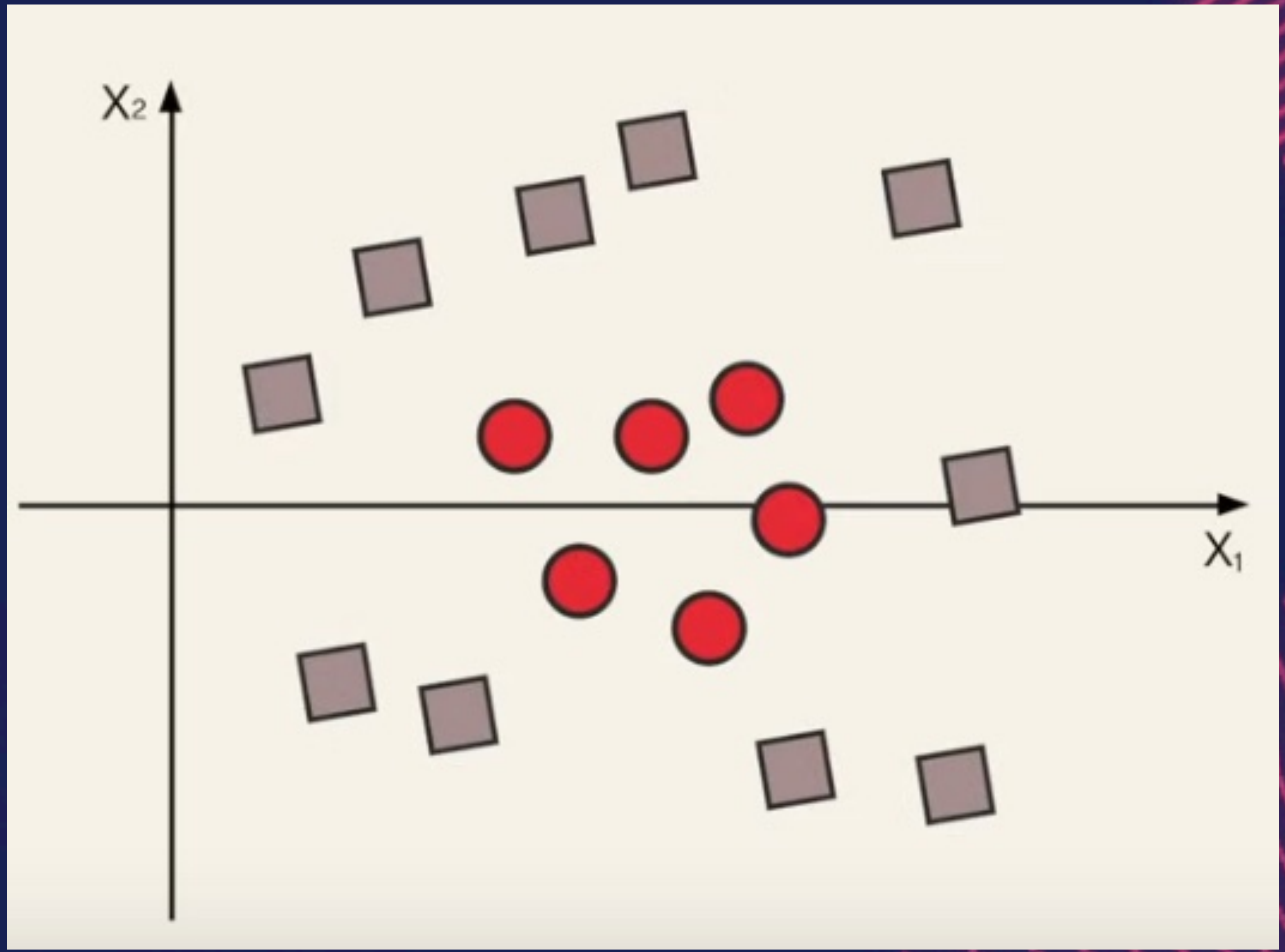
Large Value of C:

- Low Training Error
- Acts Like Hard Margin
- Over fits on training data

Small Value of C:

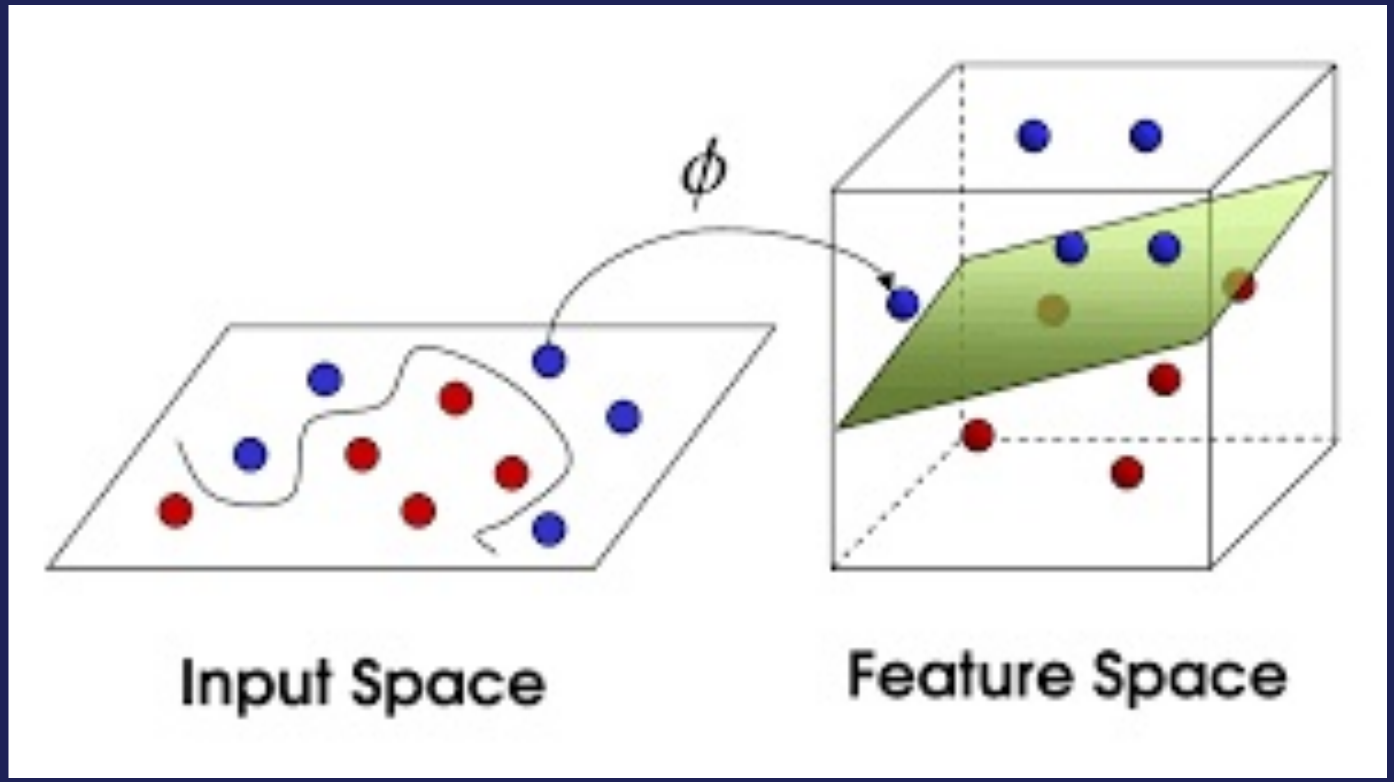
- Higher Training Error
- Acts like Soft Margin
- Under fits on training data

ACTUAL NON LINEAR DATA



To map for actual non-linear data we use something called the KERNEL METHOD

KERNEL METHOD



In kernel method we map points of certain dimension to higher dimension so that it becomes a regular SVM problem.
We have two types of kernel methods: POLYNOMIAL AND RADIAL

KERNEL COMPUTATION

Doing Lagrangian Multipliers method on the constraint y(W*X + b) >= 1 and representing the W and b in terms of one variable alpha we get the following equation that we need to optimize:

$$\text{maximise} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

Here we notice the function we need to optimize depends only on the dot product of the feature vector

KERNEL COMPUTATION

Computational Efficiency

- **Only support vectors alpha is non-zero (key idea of SVM)**
- **Computation of dot product is enough**

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

POLYNOMIAL KERNEL

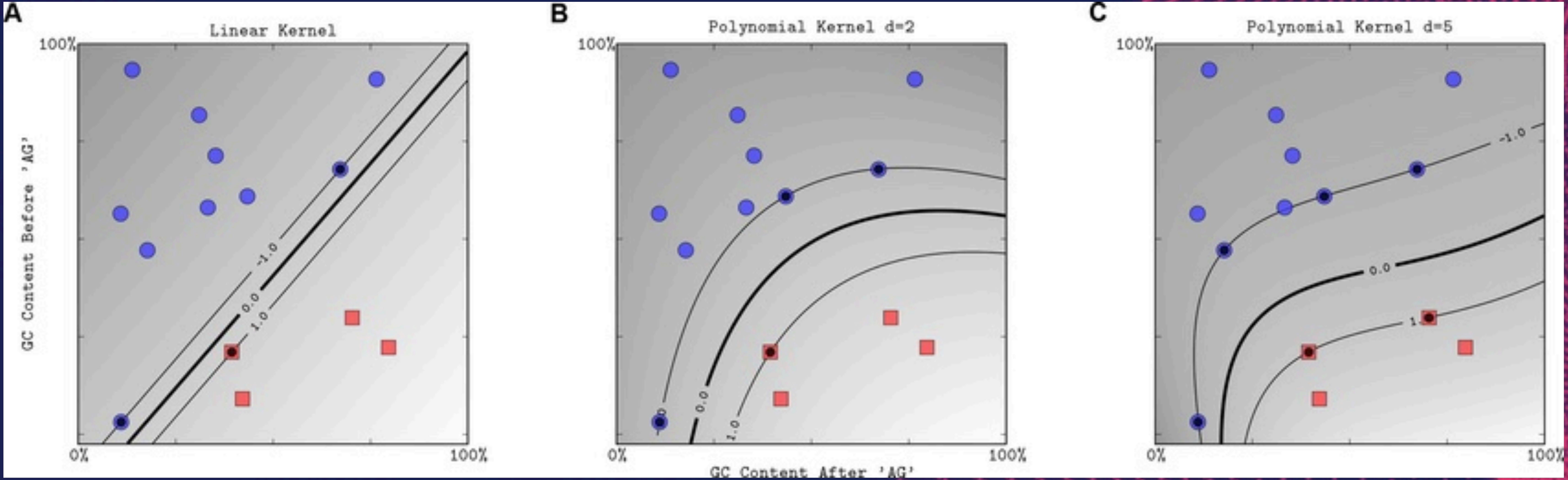
Kernel function:

$$(\gamma\langle x, x'\rangle + r)^d$$

- **r is the coefficient of the polynomial**
- **d is the degree of the polynomial**

- **Larger c implies low training error i.e. over fitting and vice versa**
- **Larger d implies high degree data leads to over fitting as complexity increases.**

POLYNOMIAL KERNEL



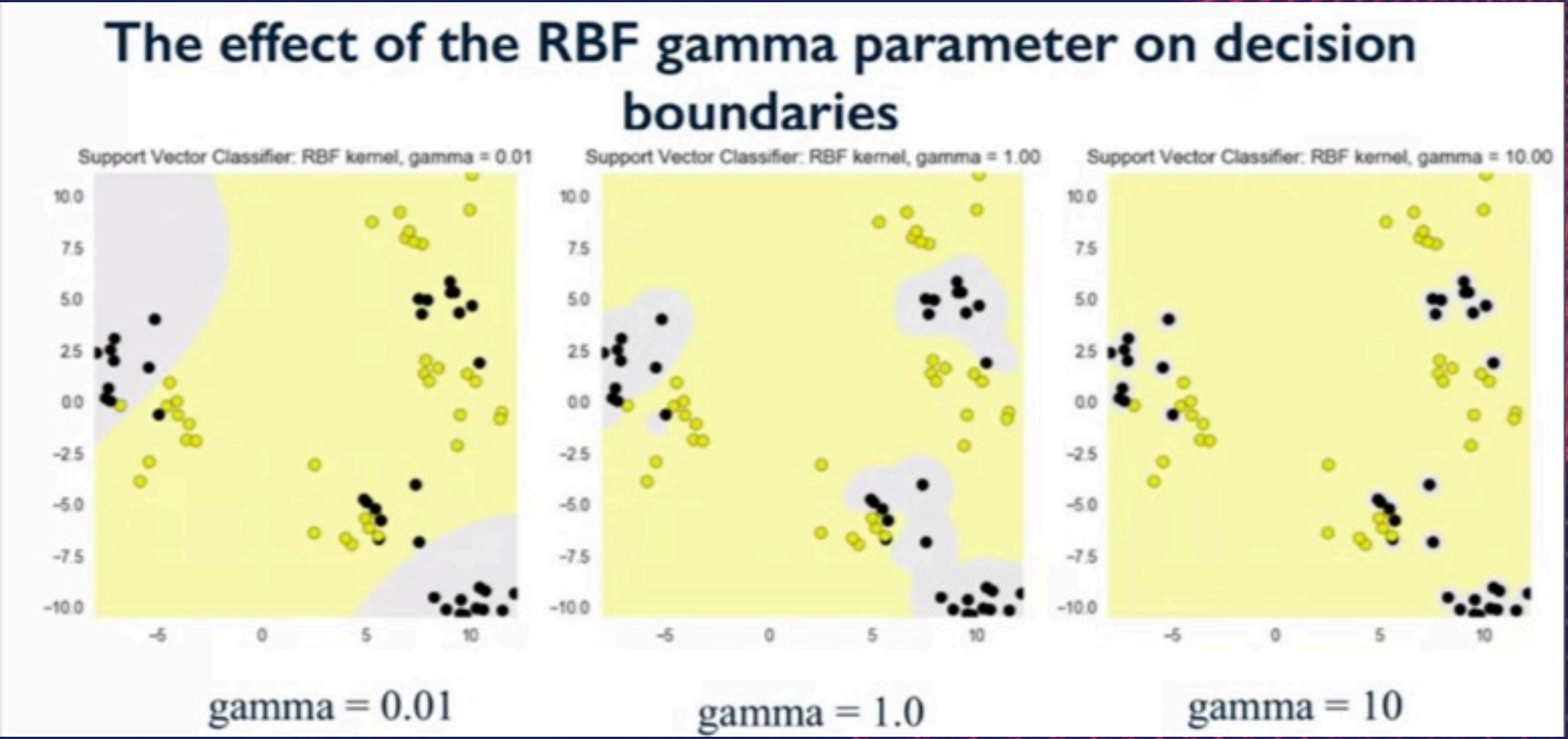
RADIAL BASIS KERNEL

Kernel function: $$K(X_1, X_2) = \exp(-\gamma \, \|X_1 - X_2\|^2)$$

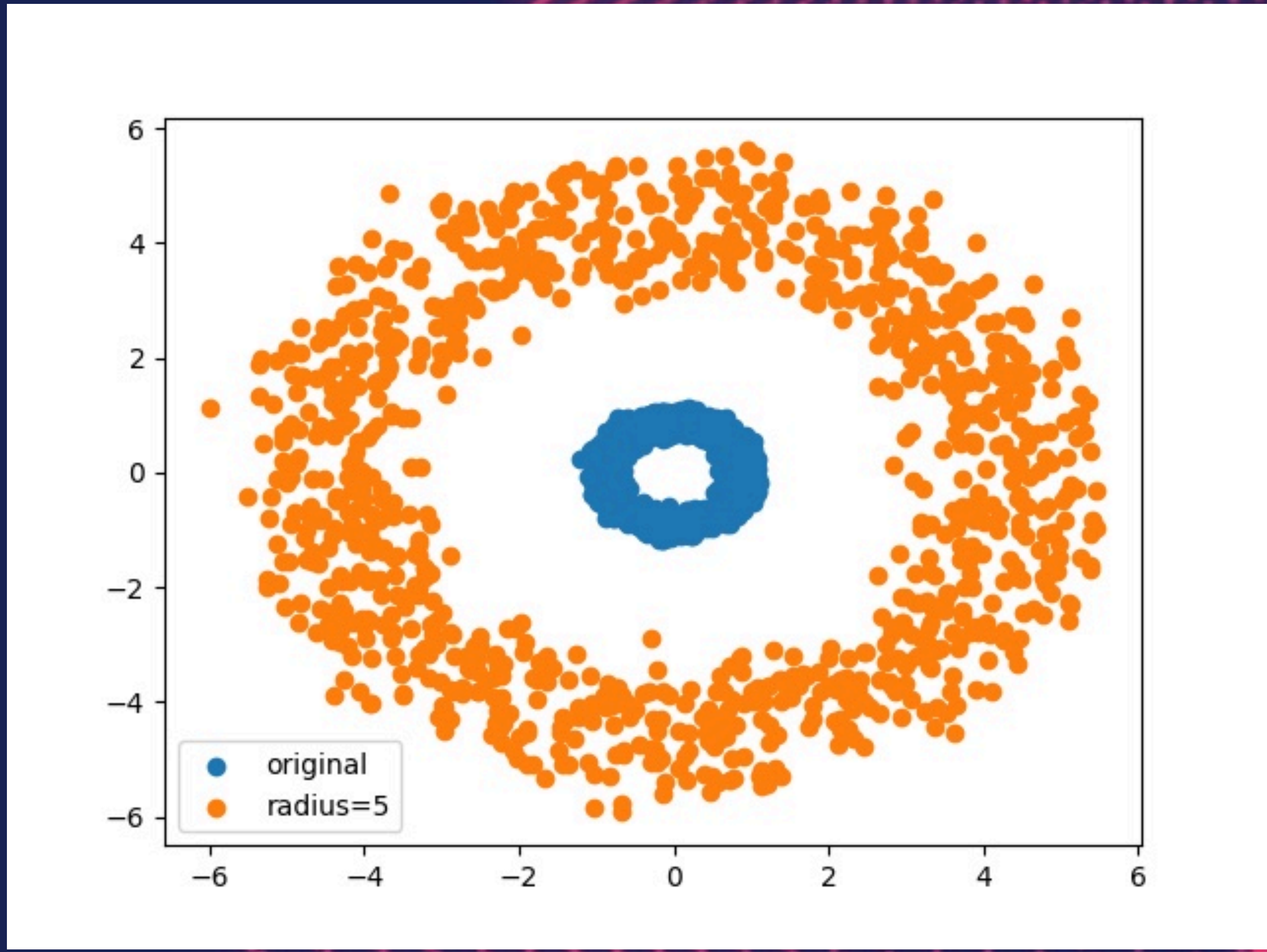
- **‖X1 - X2‖ computes the distance between X1 and X2**

- **Gamma gives the influence of each training example**
- **Higher gamma leads to over fitting and lower gamma leads to under fitting**

RADIAL BASIS KERNEL



SLIDO QUESTION



Answer the question with the help of the figure.

Join at slido.com using code 5945864

CODE IMPLEMENTATION

Google Colab

google.com