AI VIETNAM
All-in-One Course
(TA Session)

Support Vector Machine

Extra Class

AI VIET NAM
@aivietnam.edu.vn

Dinh-Thang Duong – TA

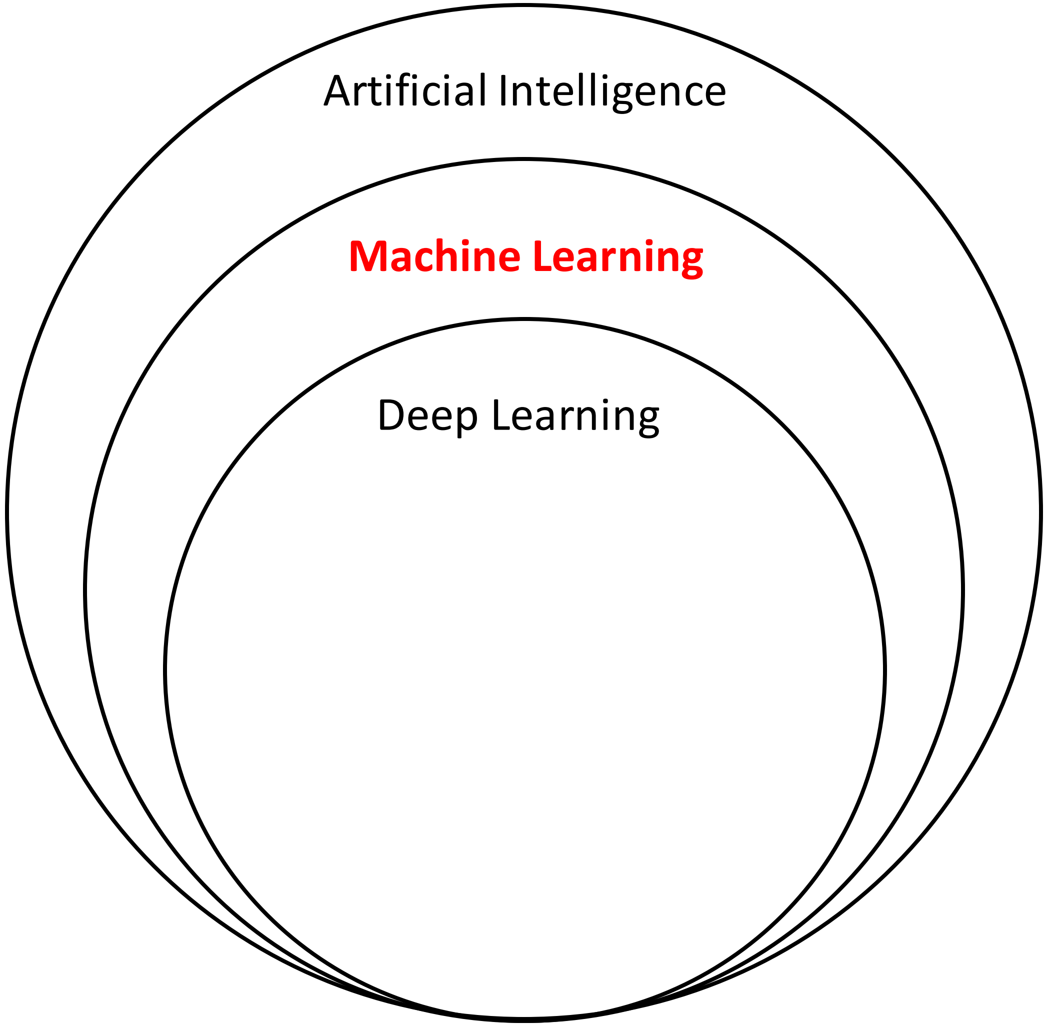
Year 2023

Outline

- Introduction
- Support Vector Machine
- Code Examples
- Question

Introduction

❖ Getting Started



Machine Learning (ML): A branch of AI and Computer Science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

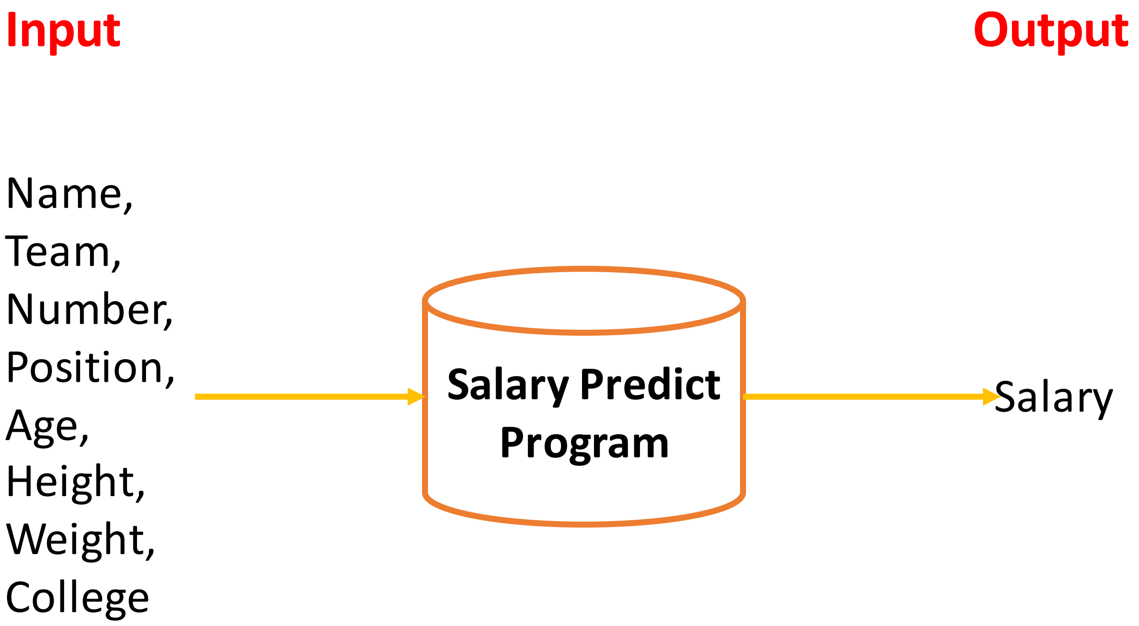
https://www.ibm.com/topics/machine-learning

Introduction

❖ Getting Started

Suppose you got some dataset:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	Boston Celtics	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
11	Isaiah Thomas	Boston Celtics	4.0	PG	27.0	5-9	185.0	Washington	6912869.0
12	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0
13	James Young	Boston Celtics	13.0	SG	20.0	6-6	215.0	Kentucky	1749840.0



And you want to make a program to automatically predict value of 1 column based on others.

X(Name, Team, Number, Position, Age, Height, Weight, College) -> Y(Salary)

Introduction

❖ Getting Started

Input X | Output Y

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	Boston Celtics	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
11	Isaiah Thomas	Boston Celtics	4.0	PG	27.0	5-9	185.0	Washington	6912869.0
12	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0
13	James Young	Boston Celtics	13.0	SG	20.0	6-6	215.0	Kentucky	1749840.0

Using this data to "train" an ML Model

ML Model

Input | **Output**

Use the trained model to predict Salary based on any given input

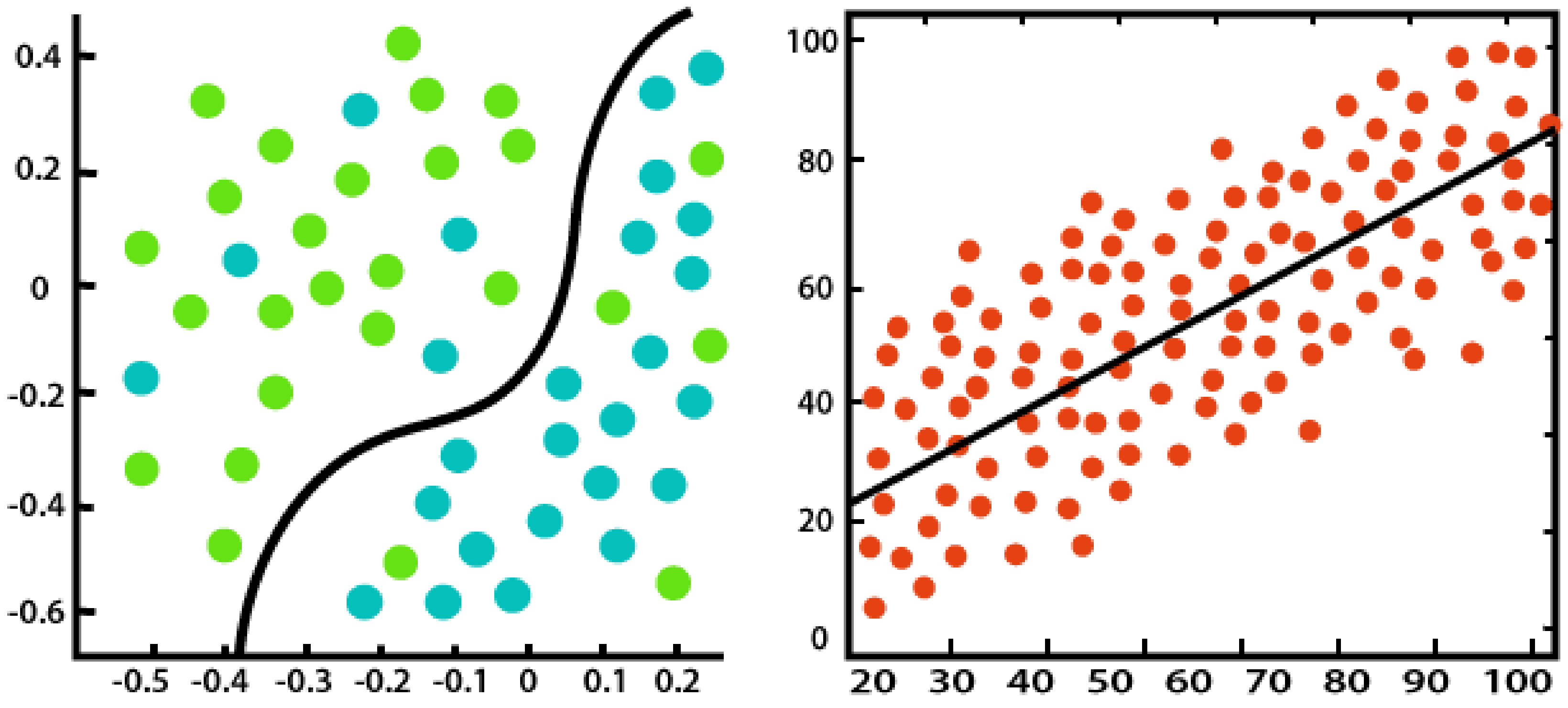
Name, Team, Number, Position, Age, Height, Weight, College → **Trained ML Model** → Salary

Since we **use a labeled dataset to train** ML Model.

=> This is called **Supervised-learning**.

Introduction

❖ Supervised Learning



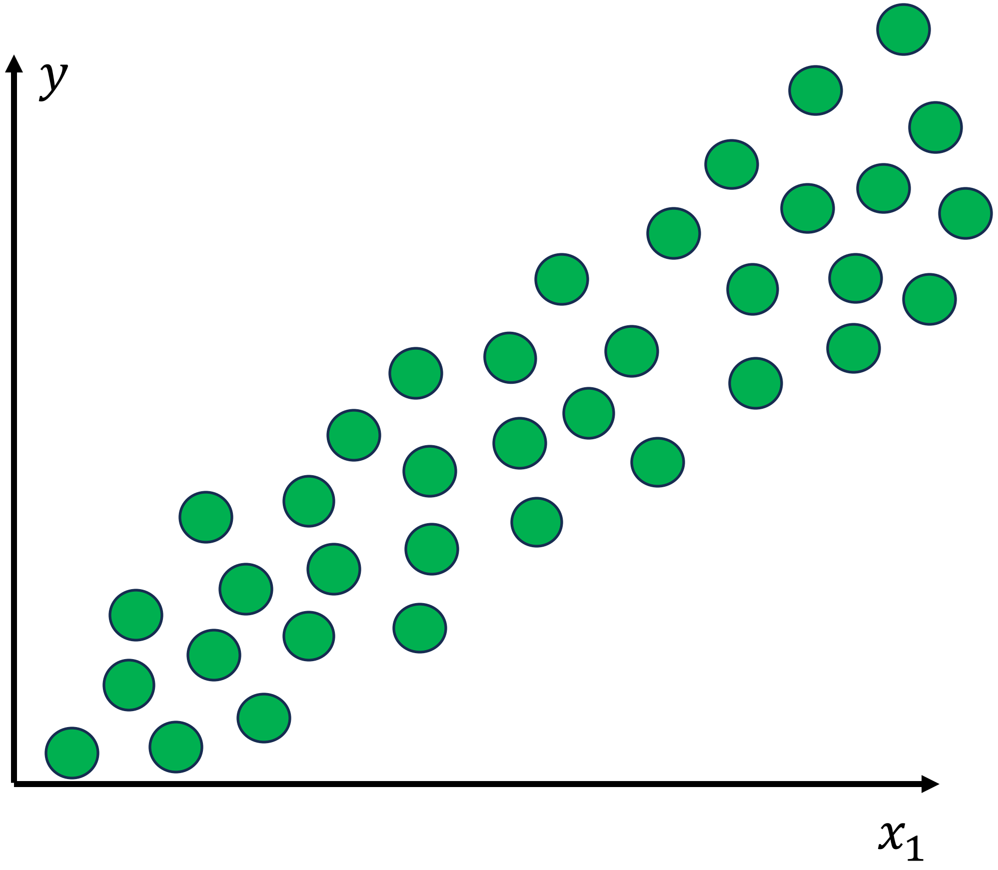
Classification Regression

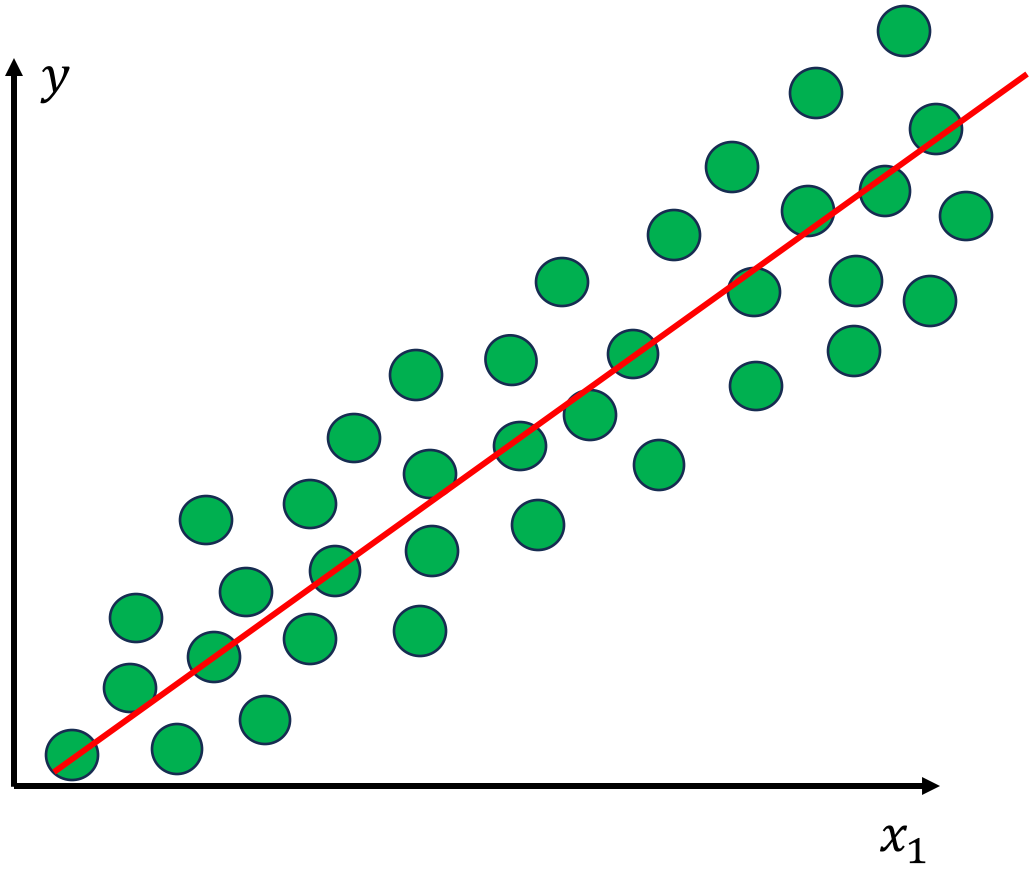
In ML Superivsed-learning algorithms, we often deal with Regression and Classification

Introduction

❖ Supervised Learning: Regression

Regression: A task involving predicting a **continuous value** based on given inputs.



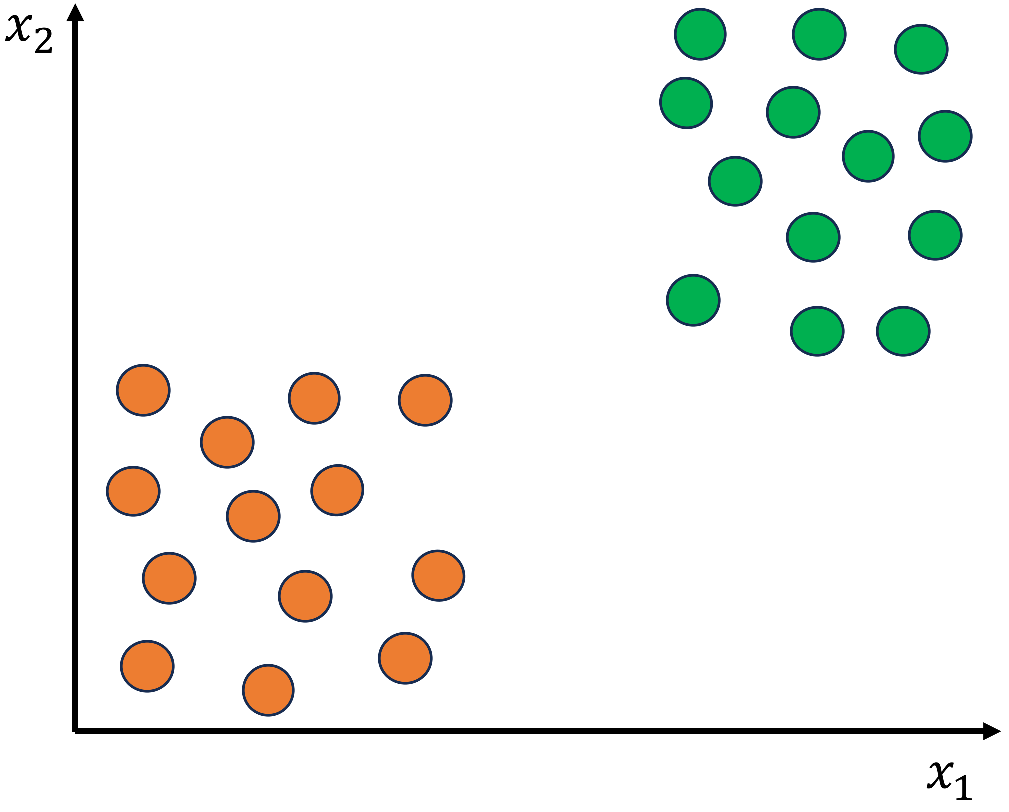


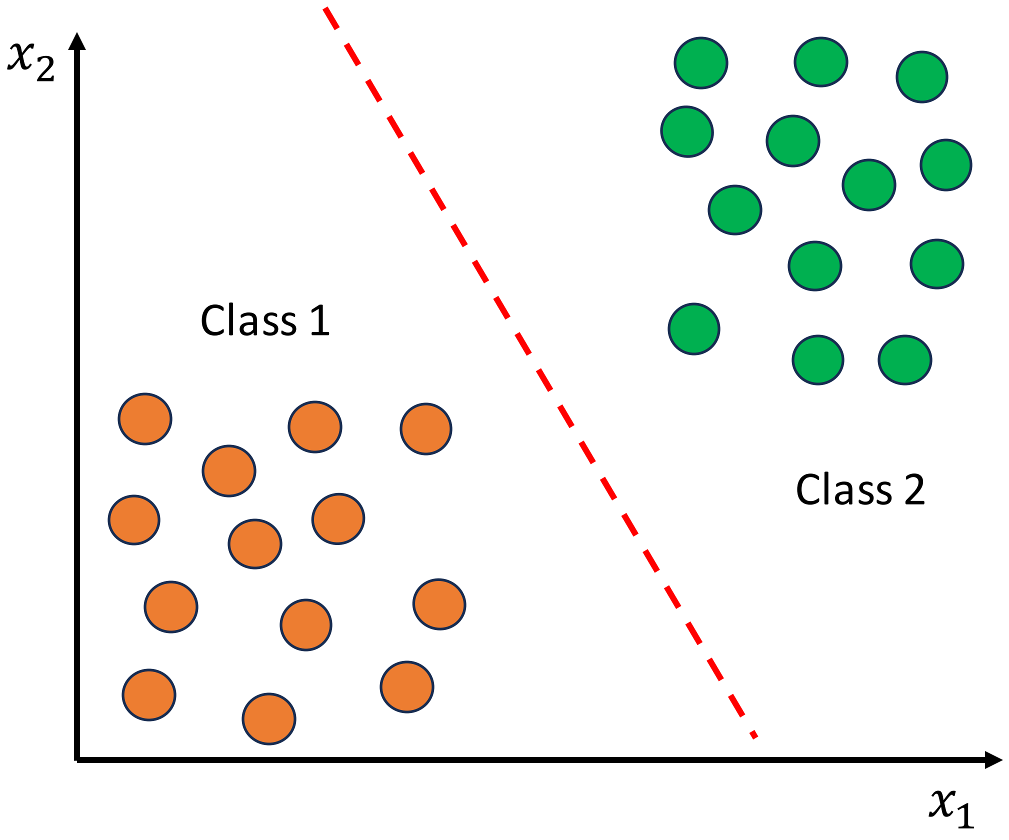
In general, we want to find the line that best fit the data distribution.

Introduction

❖ Supervised Learning: Classification

Classification: A task involving predicting a **discrete (categorical) value** based on given inputs.

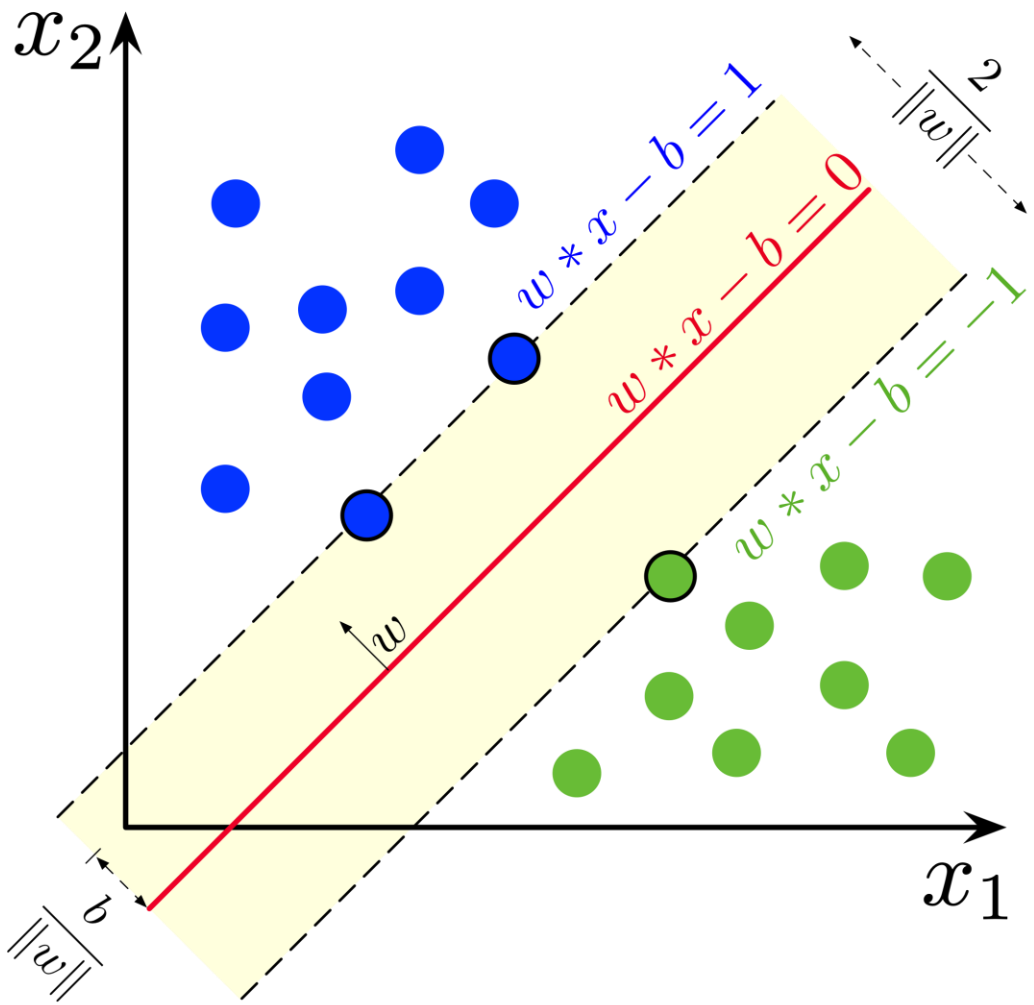




In general, we want to find the line that best separates the dataset into classes.

Support Vector Machine

❖ Introduction

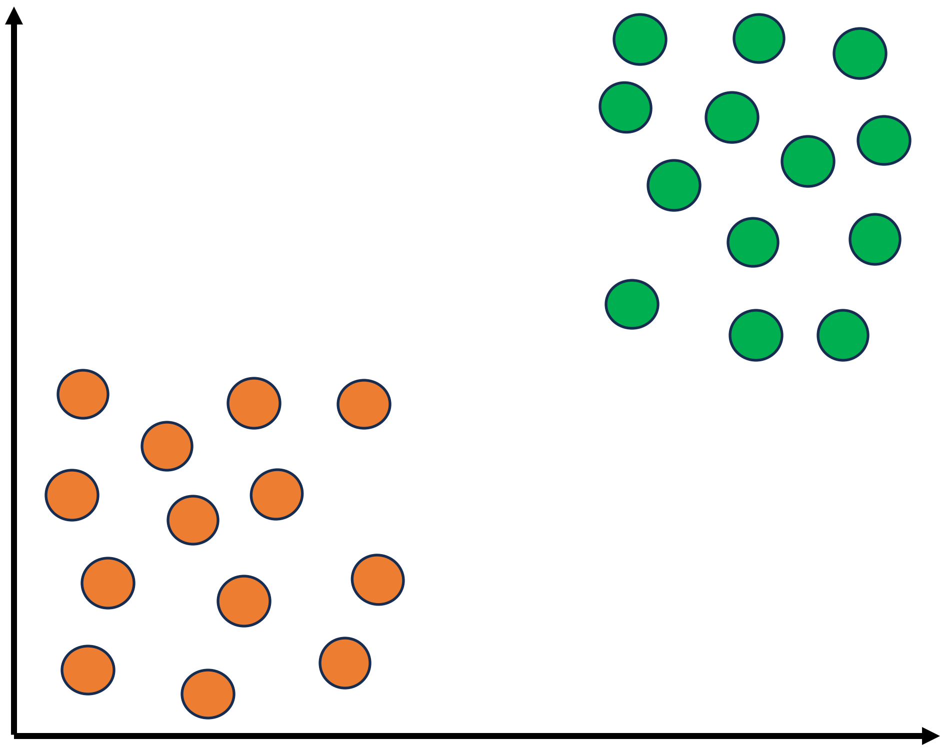


Support Vector Machine (SVM): A supervised-learning ML algorithm that works by identifying the optimal hyperplane that best separates data into different classes.

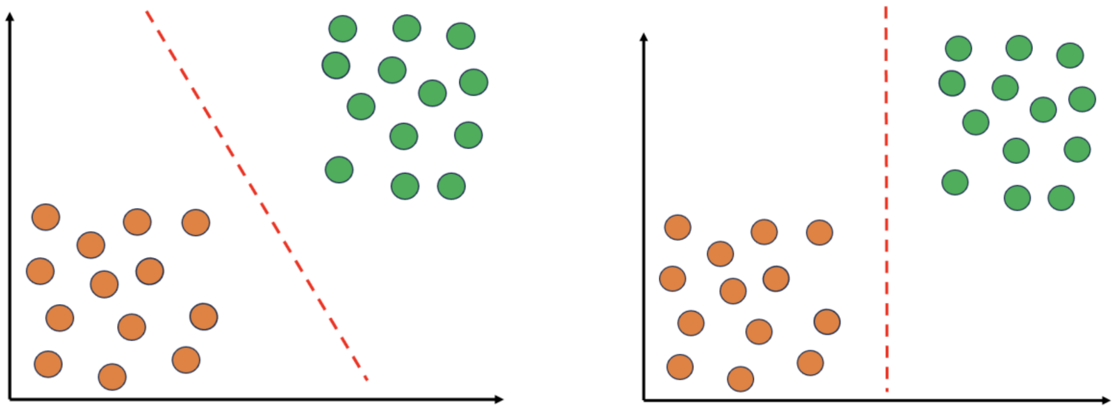
SVM was originally built for classification task (SVC) but was later modified to fit for regression task (SVR) too.

Support Vector Machine

❖ Getting Started



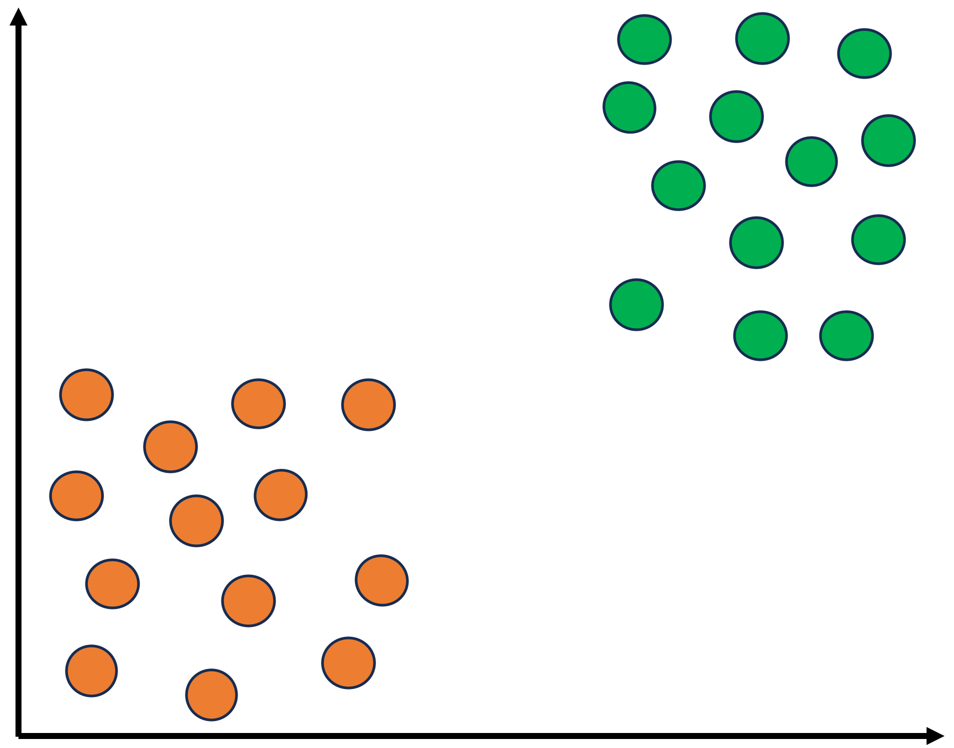
Assume we have a linearly separable dataset



Linearly separable data: A dataset that can be fully separated into classes using a single line.

Support Vector Machine

❖ Getting Started

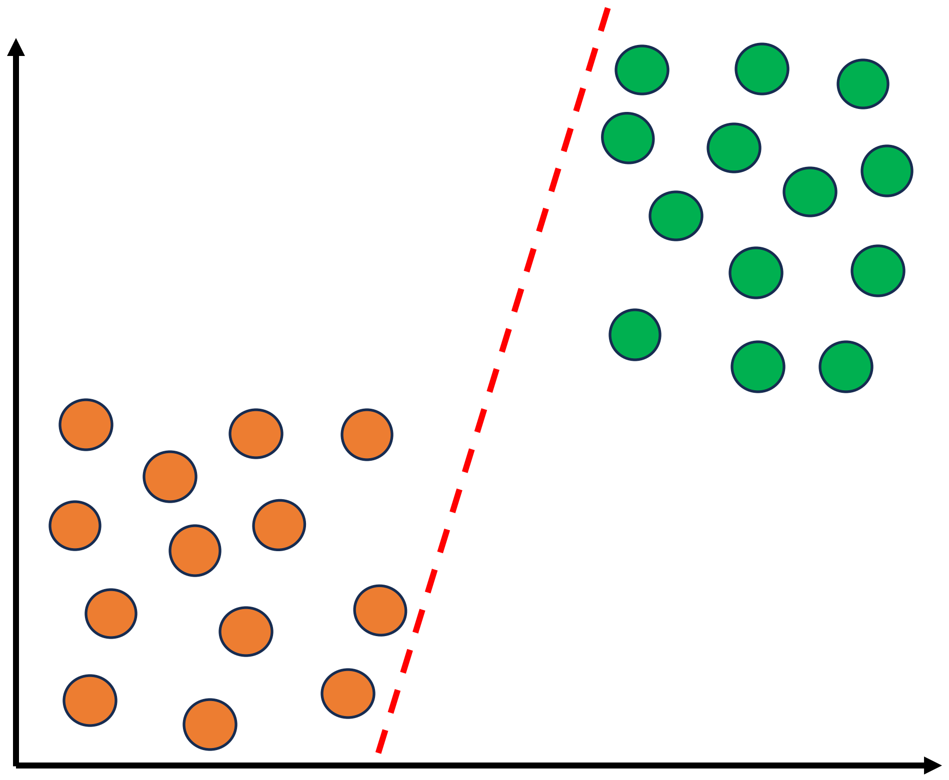


How should we draw a line so that we can perfectly separate this dataset into 2 classes?

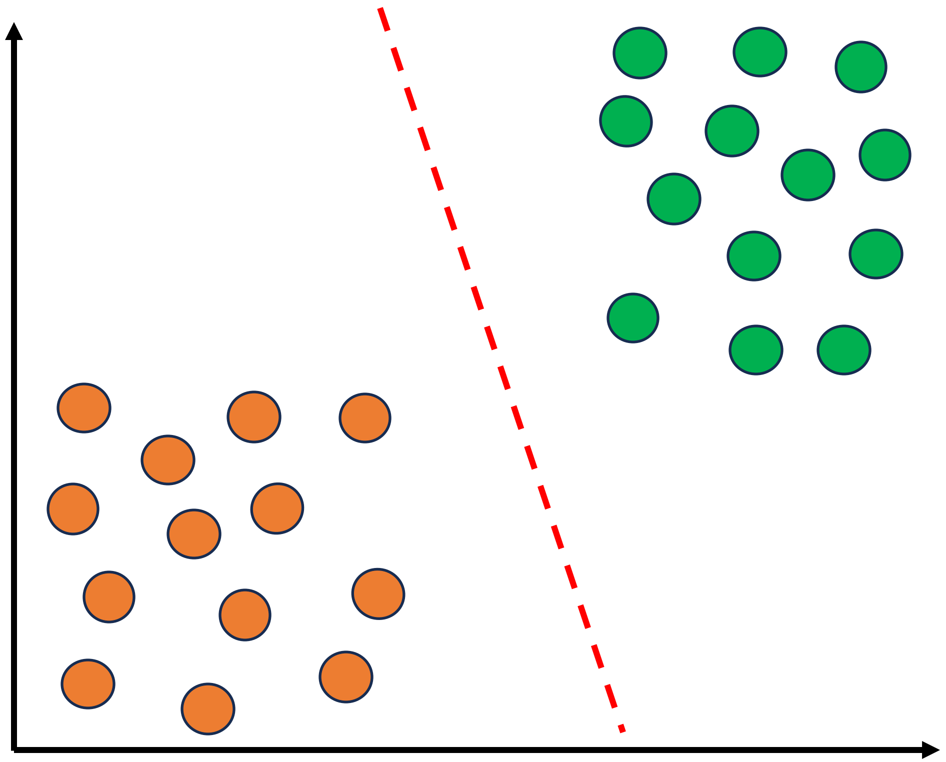
Support Vector Machine

❖ Getting Started

Example 1



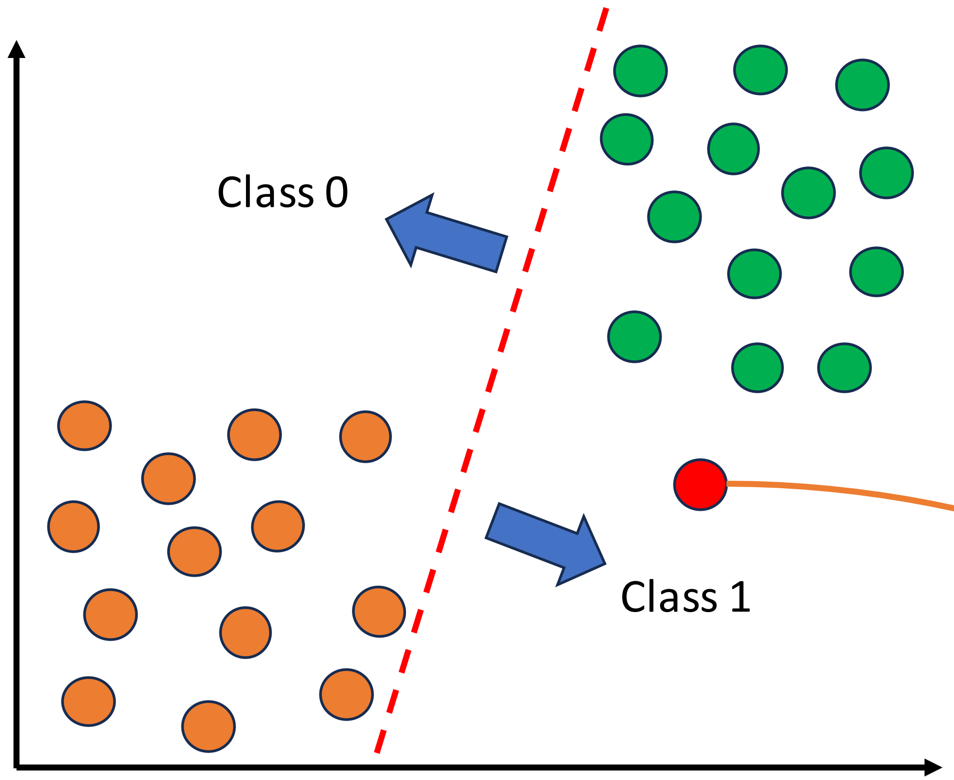
Example 2



There are many ways to draw the line

Support Vector Machine

❖ Getting Started

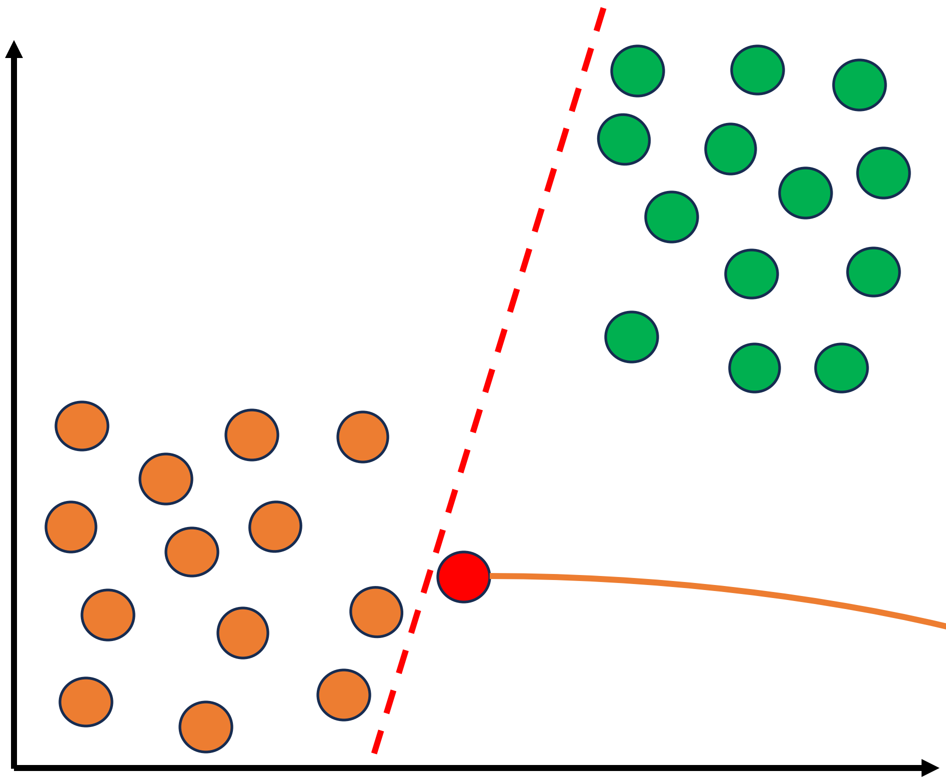


With this line, we can now determine whether a new data point belongs to Class 0 or Class 1 based on which side of the line it falls on.

This point is classified as **Class 1** since it lies on right hand side of the line.

Support Vector Machine

❖ Getting Started



However, in this situation, the result seems wrong since the point is more closer to Class 0.

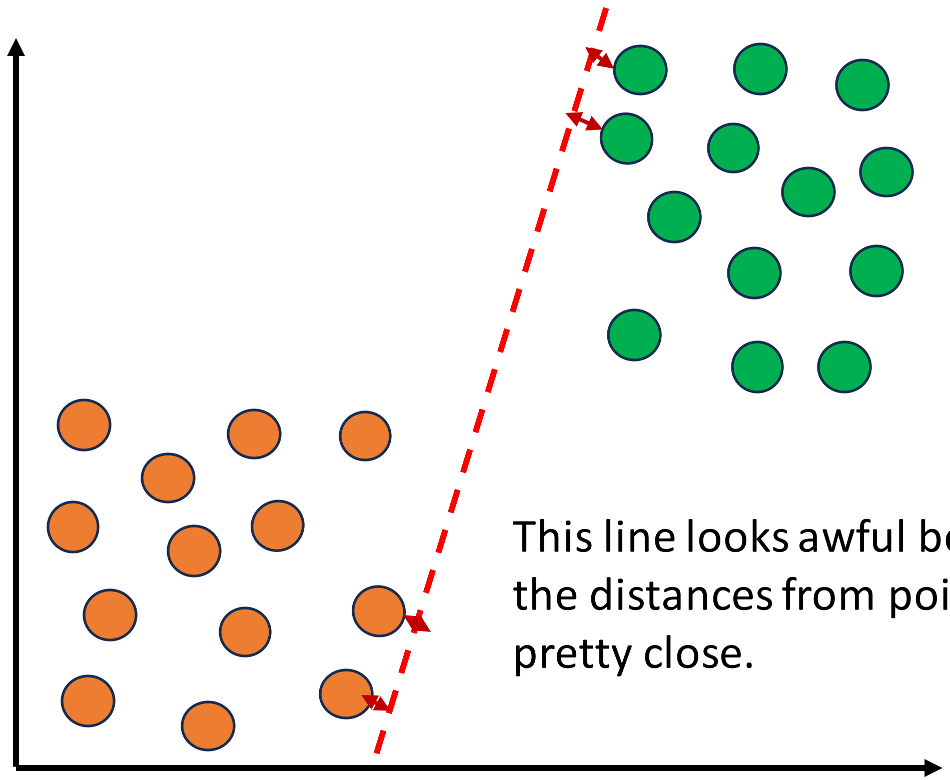
=> This line is not really optimal.

What would be the best line?

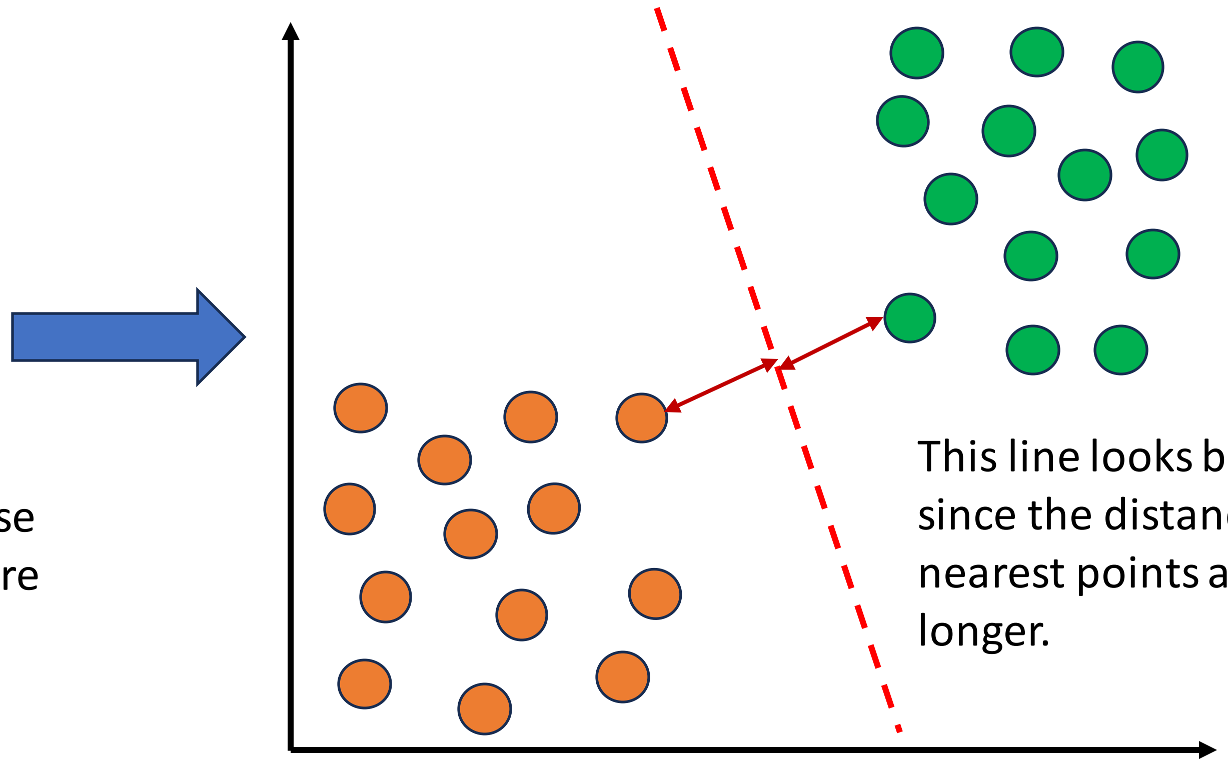
This point is classified as **Class 1** since it lies on right hand side of the line.

Support Vector Machine

❖ Getting Started



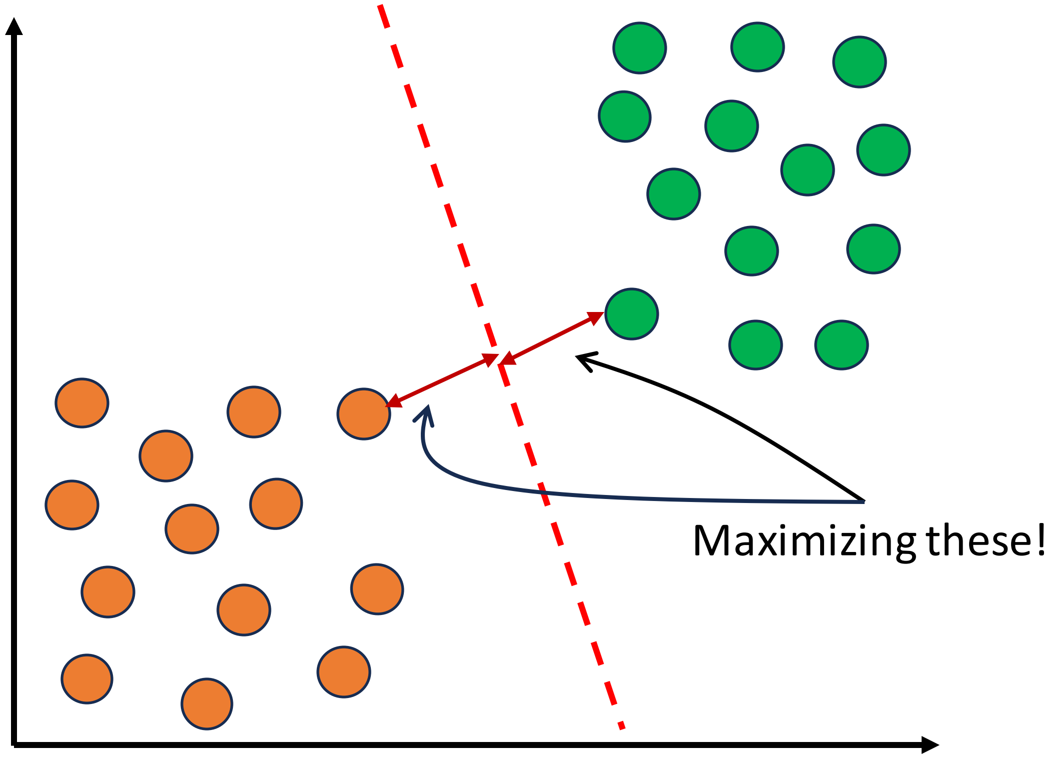
This line looks awful because the distances from points are pretty close.



This line looks better since the distances to nearest points are much longer.

Support Vector Machine

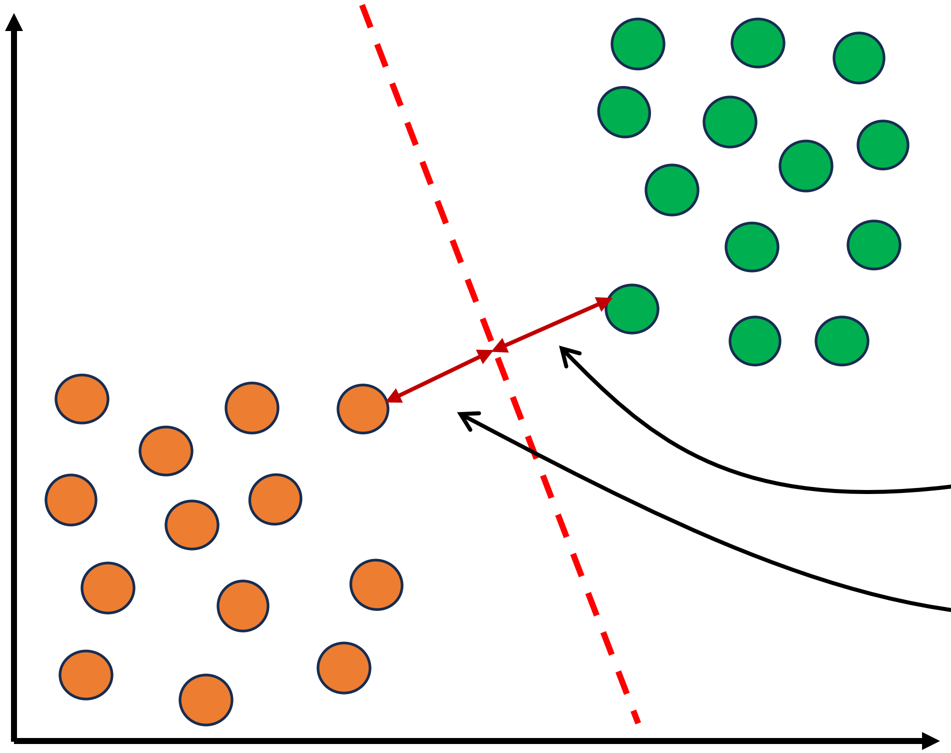
❖ Idea



Idea: Find the line that best separates the data into classes while maximizing the distances between nearest points.

Support Vector Machine

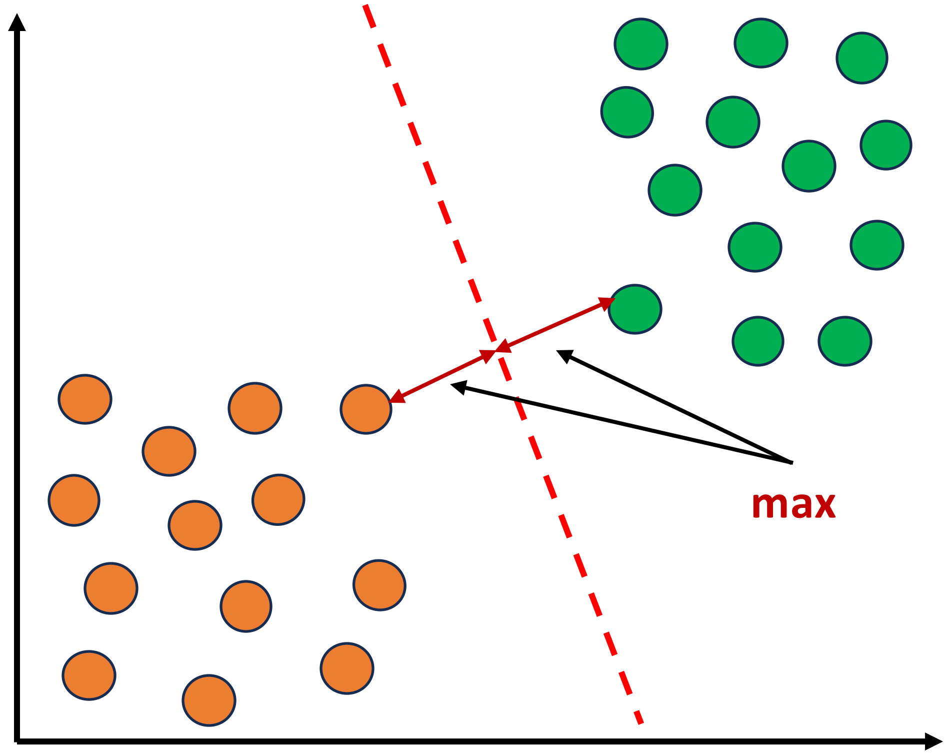
❖ Margin



The distance between the line and the nearest data point from either of the two classes is called **margin**.

Support Vector Machine

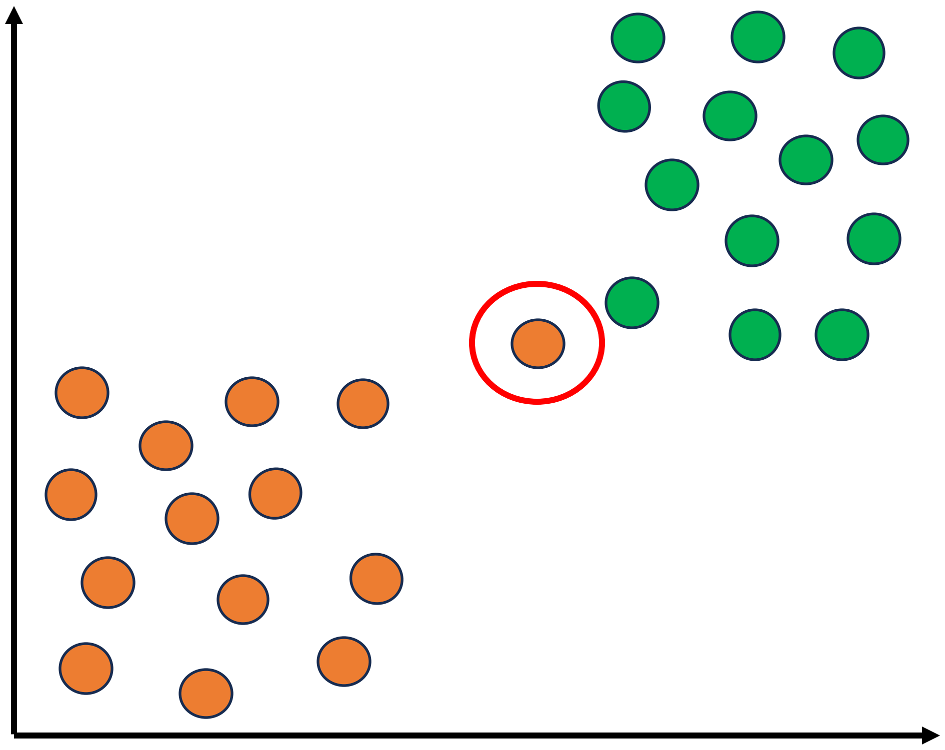
❖ Hard Margin Classifier Idea



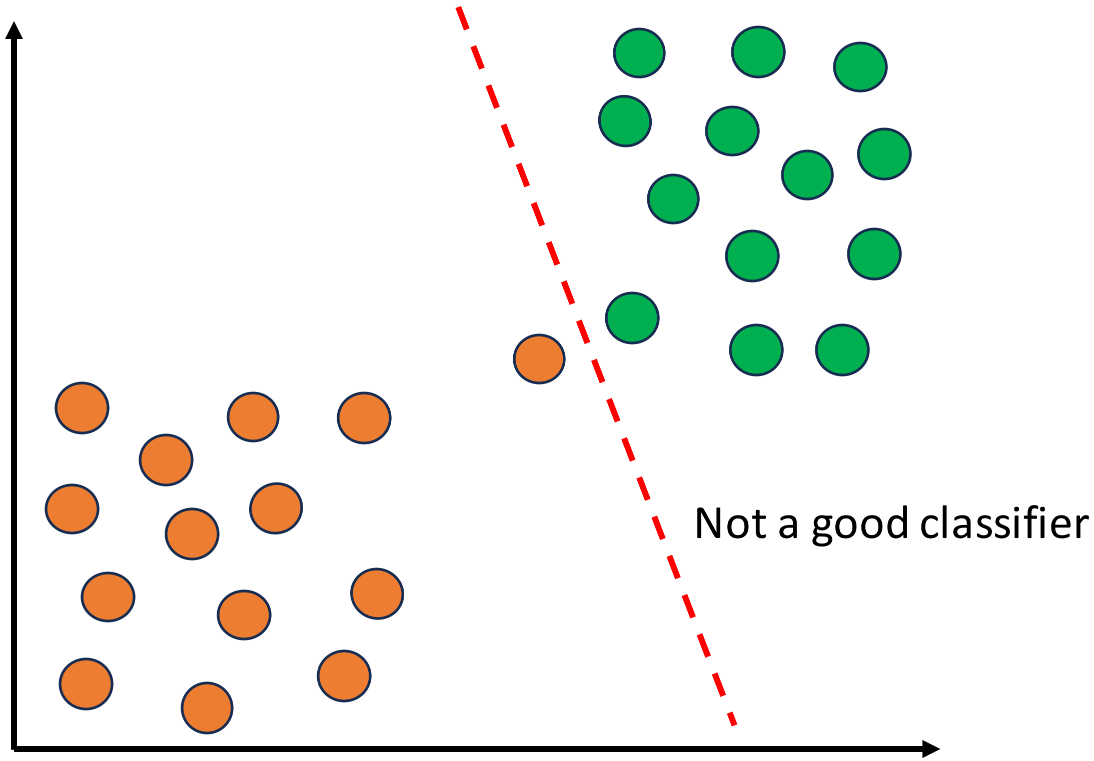
Idea: Find the line that best separates the data into classes while **maximizing the margin**. This is called **Hard Margin Classifier**.

Support Vector Machine

❖ Hard Margin Classifier Problem



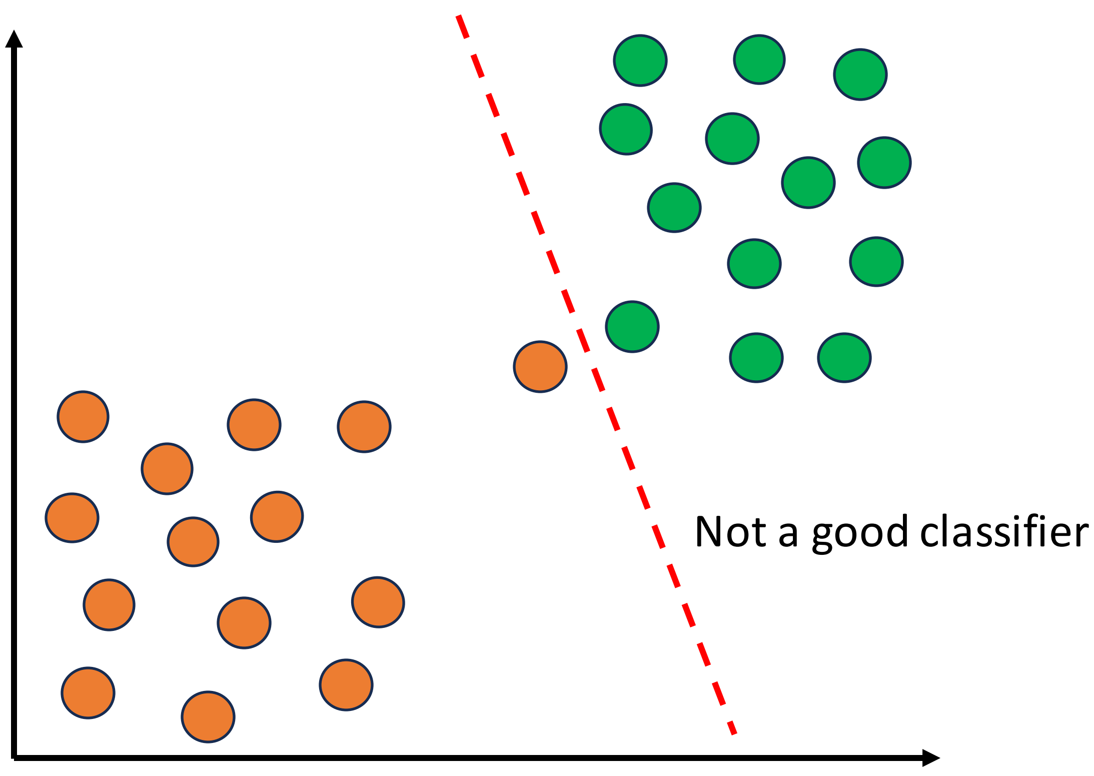
However, assume we have **an outlier**

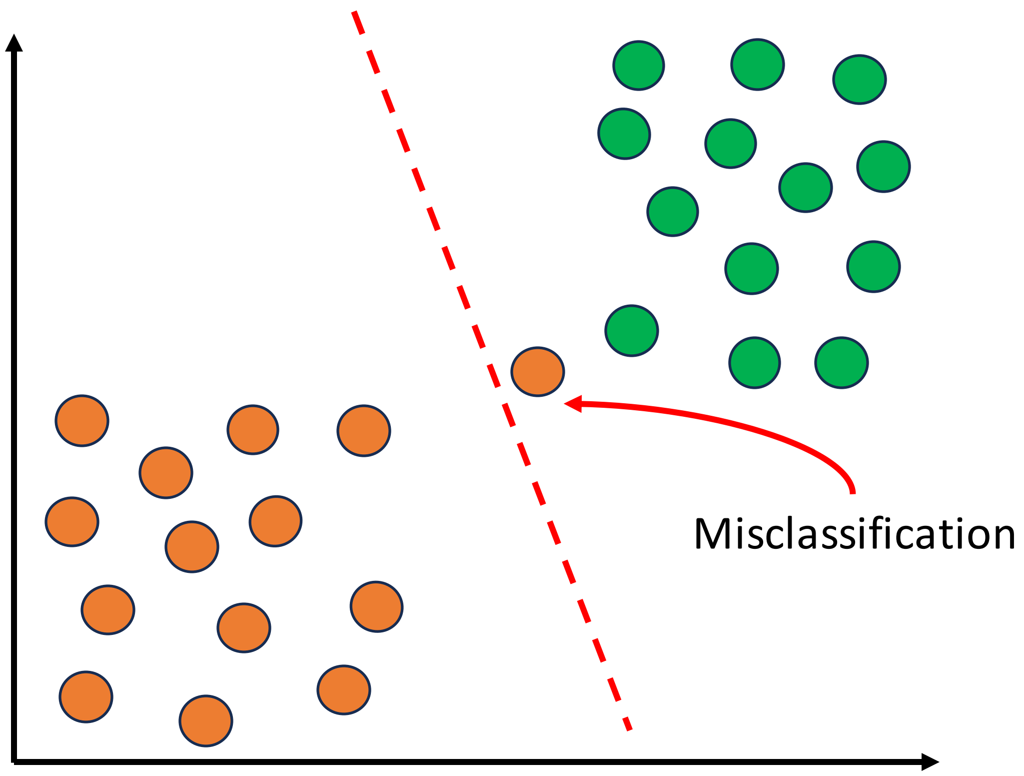


Using Hard Margin Classifier, we might have a line like this.

Support Vector Machine

❖ Soft Margin Classifier



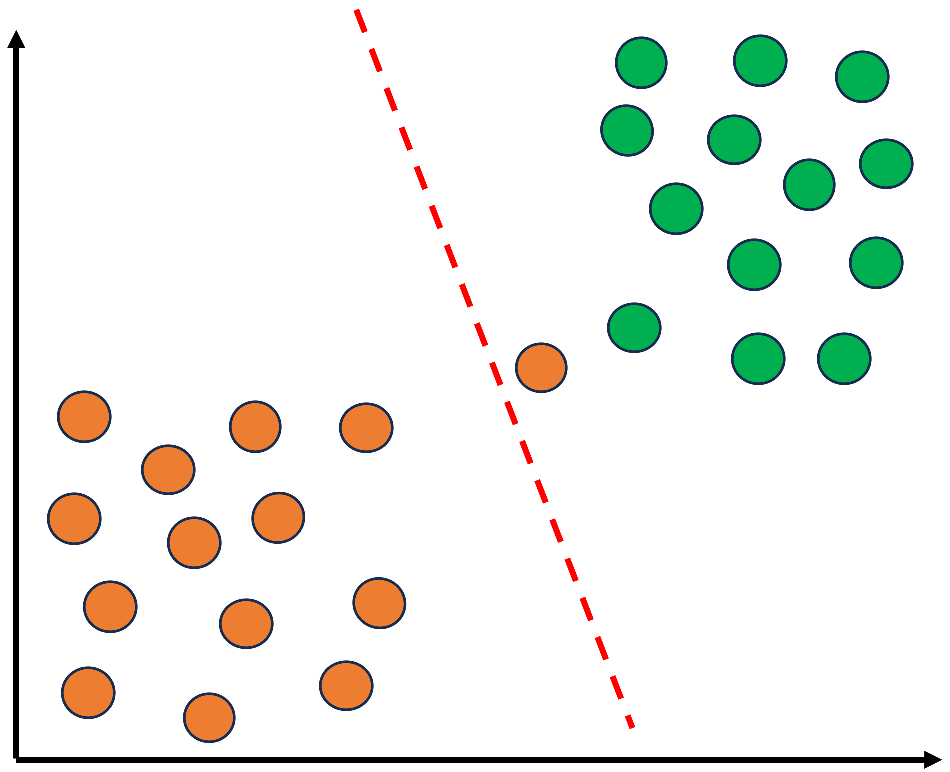


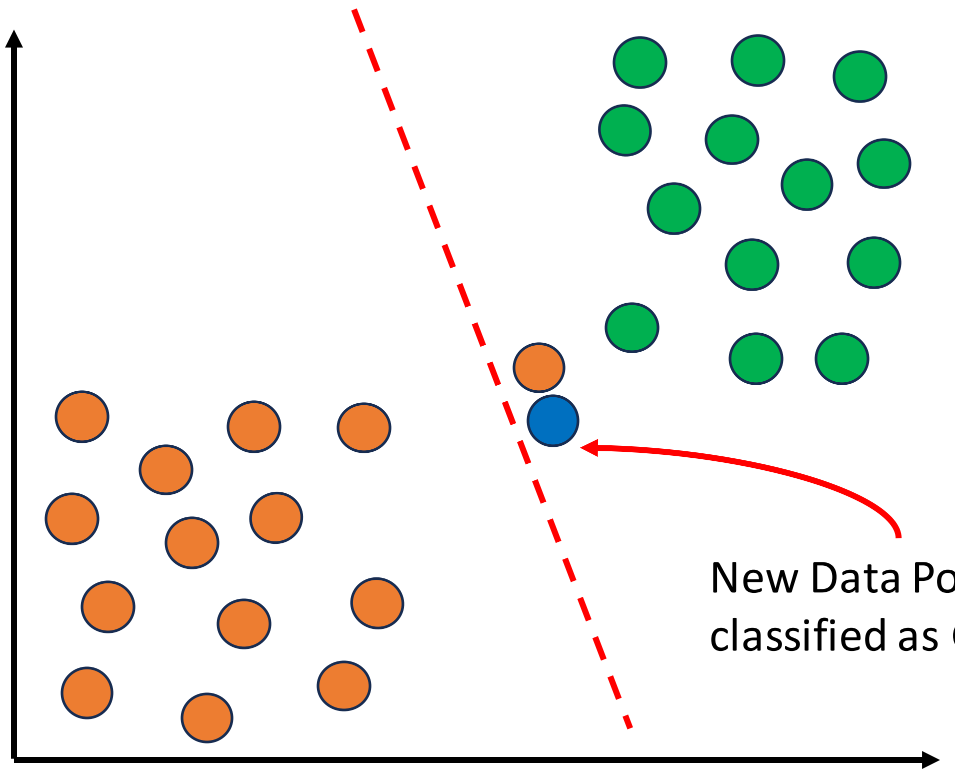
How to avoid this case?

To avoid this, we should **allow misclassification**

Support Vector Machine

❖ Soft Margin Classifier

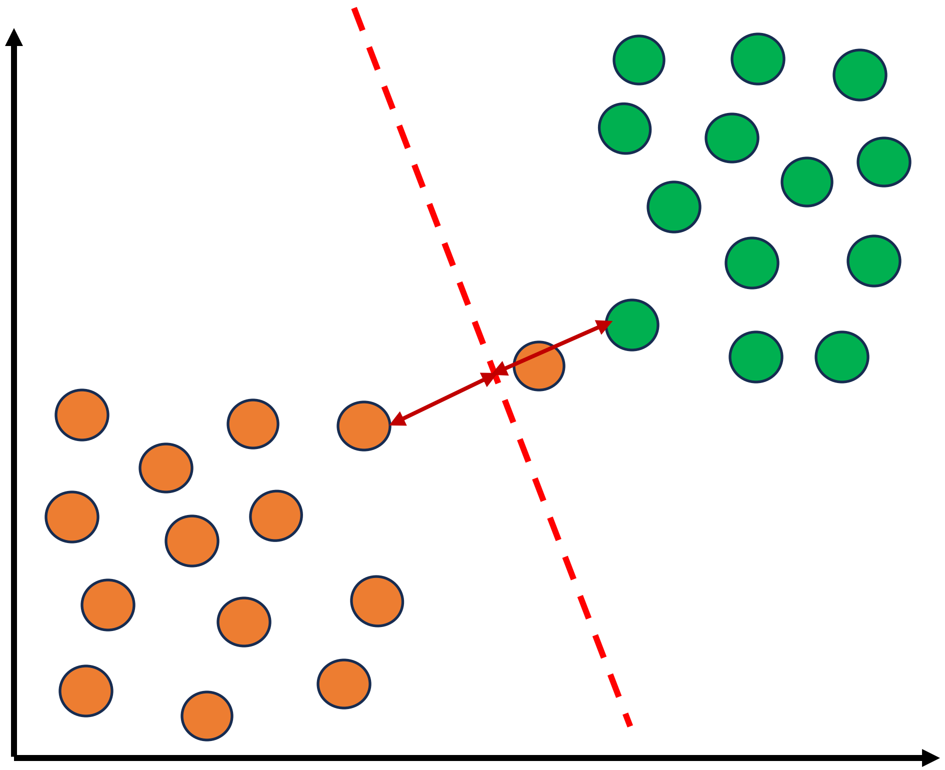




However, when we have a new data point, we might get it right.

Support Vector Machine

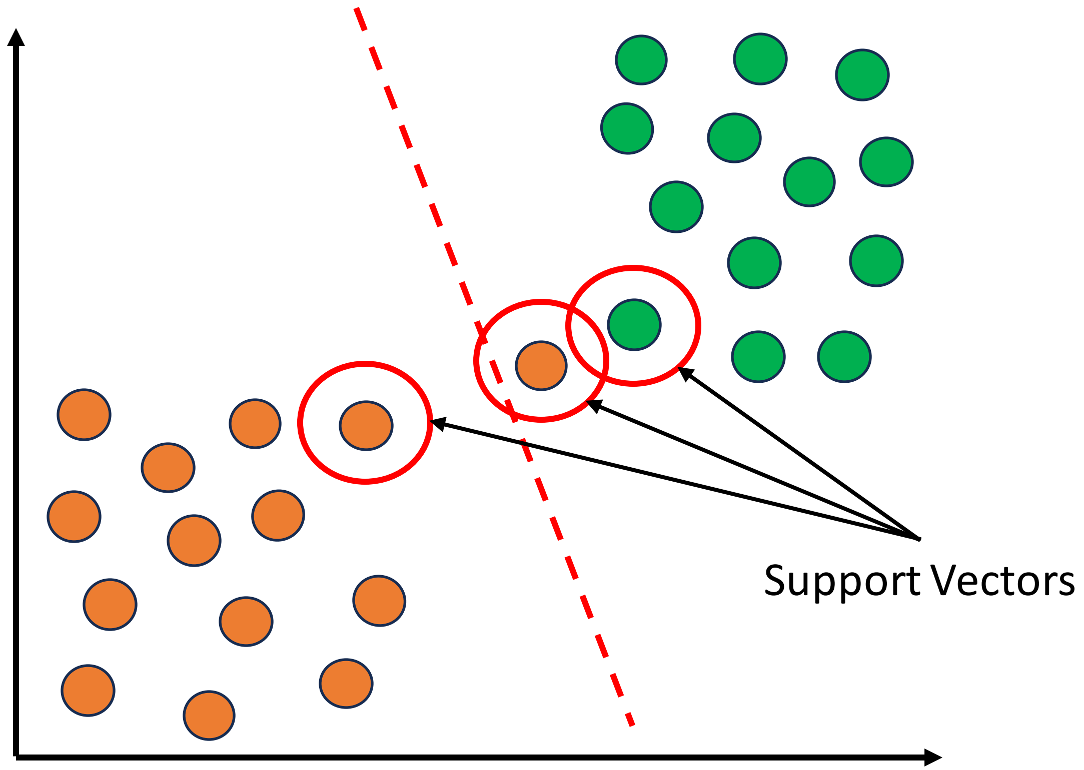
❖ Soft Margin Classifier



When we **allow misclassifications**, the distance between the observations and the decision boundary is called **Soft Margin** **=> Soft Margin Classifier (Support Vector Classifier).**

Support Vector Machine

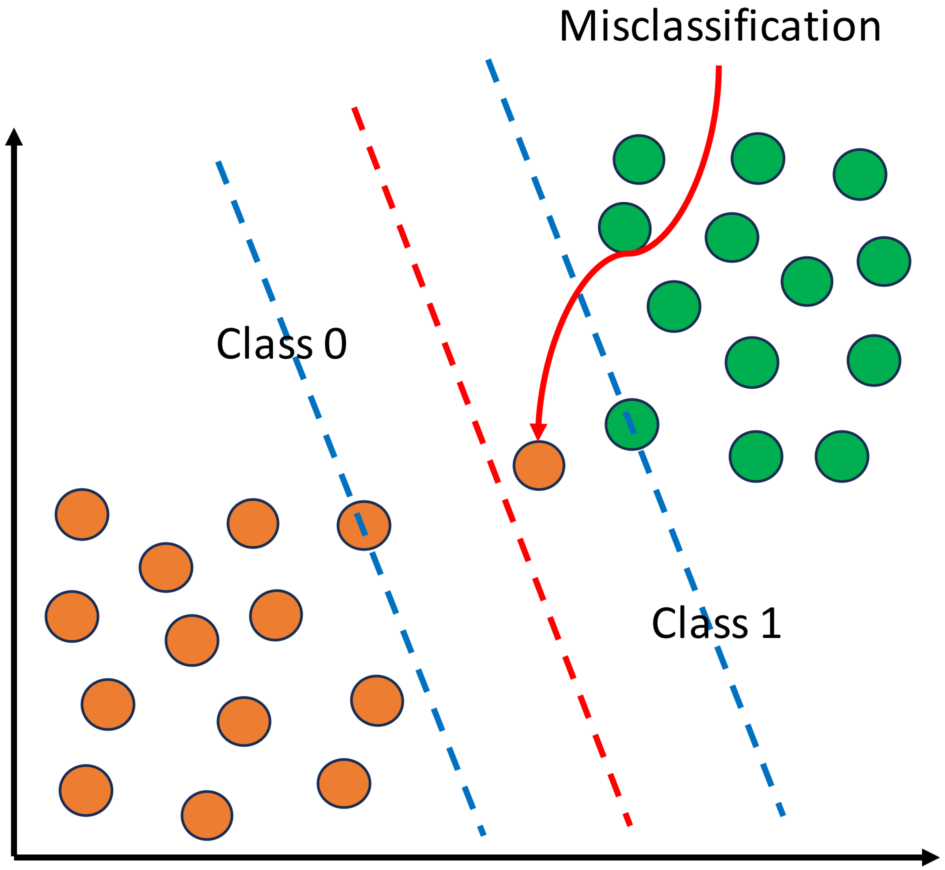
❖ Why "Support" Vector Classifier



We called "Support" Vector Classifier because the **data points on the edge and within the Soft Margin** are called Support Vectors.

Support Vector Machine

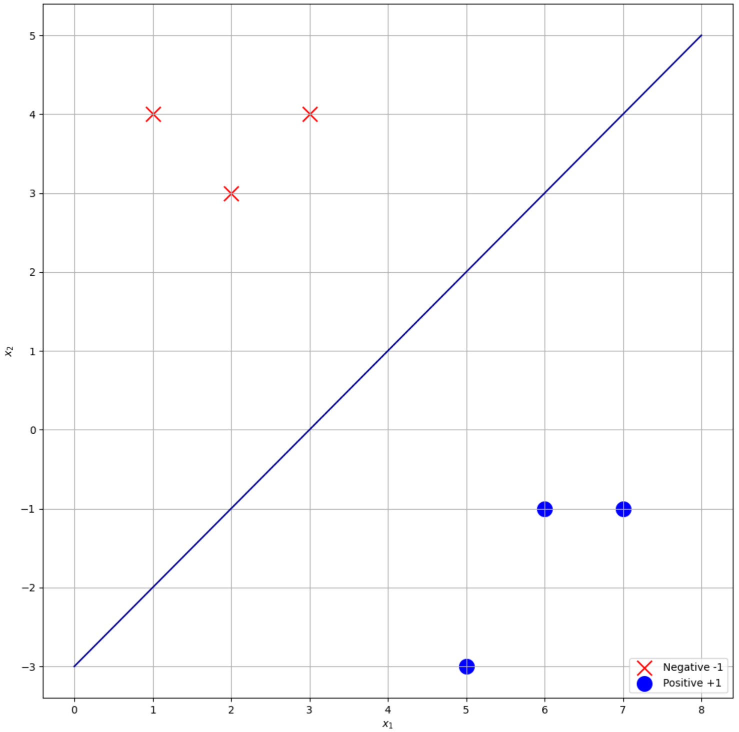
❖ Why "Support" Vector Classifier



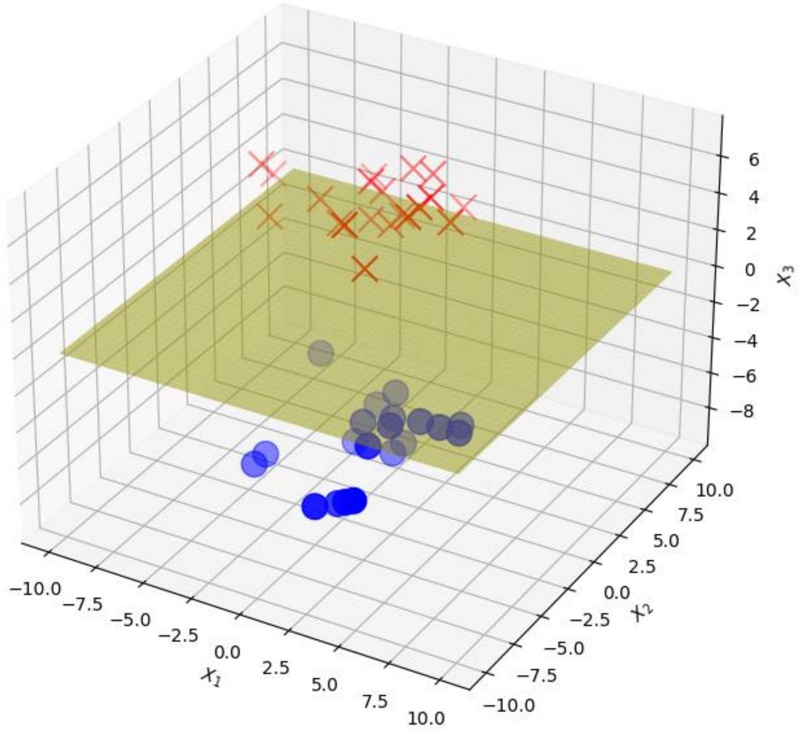
To better have a sense of relation between data points and Soft Margin, we draw two parallel lines to the Decision Boundary on Support Vectors.

Support Vector Machine

❖ SVC: Hyperplane



In 2D space, decision boundary is a line

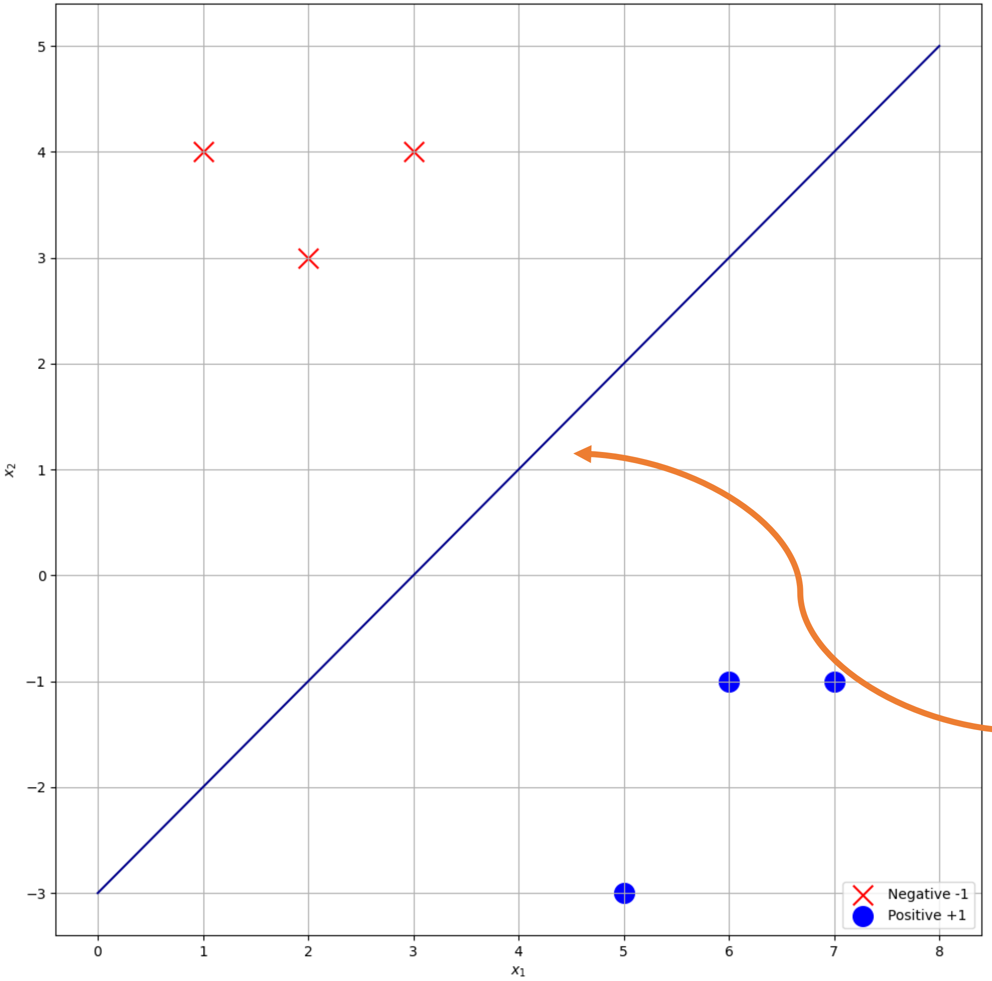


But in 3D, decision boundary is instead a plane

2D → line
3D → plane
Higher → Hyperplane

Support Vector Machine

❖ SVC: Hyperplane



Equation of Hyperplane

$$w \cdot x + b = 0$$

Hypothesis Function *h(x)*

- $h(x_i) = \begin{cases} +1 \; if \; w \cdot x + b \geq 0 \\ -1 \; if \; w \cdot x + b < 0 \end{cases}$

- $h(x_i) = sign(w \cdot x + b)$

With w = (1, -1) and b = -3 we get this hyperplane.

We use the hypothesis function to predict the class of a data point.

Support Vector Machine

❖ SVC: Prediction

With w = (1, -1) and b = -3, the equation of hyperplane becomes:

$$w \cdot x + b = x_1 - x_2 - 3 = 0$$

X1	X2	Y
3	4	-1
1	4	-1
2	3	-1
6	-1	1
7	-1	1
5	-3	1

(1*3) + (-1*4) + (-3) = -7 < 0 → $y_{predict} = -1$

Classifying a data point using the hyperplane.

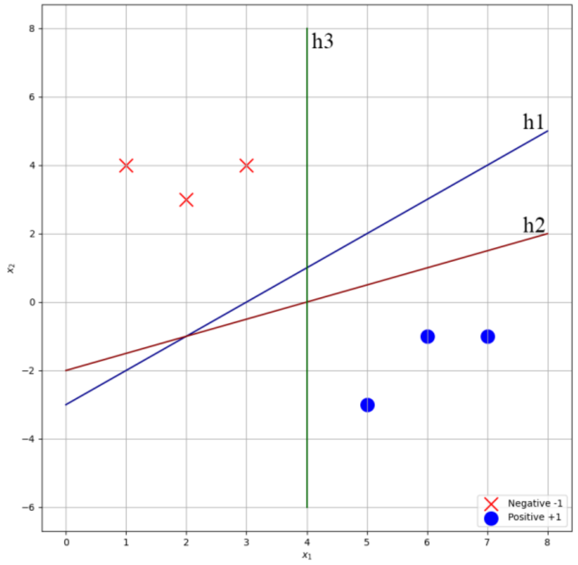
(1*7) + (-1*-1) + (-3) = 5 > 0 → $y_{predict} = +1$

In this example, we use X1 and X2 to predict Y.

Support Vector Machine

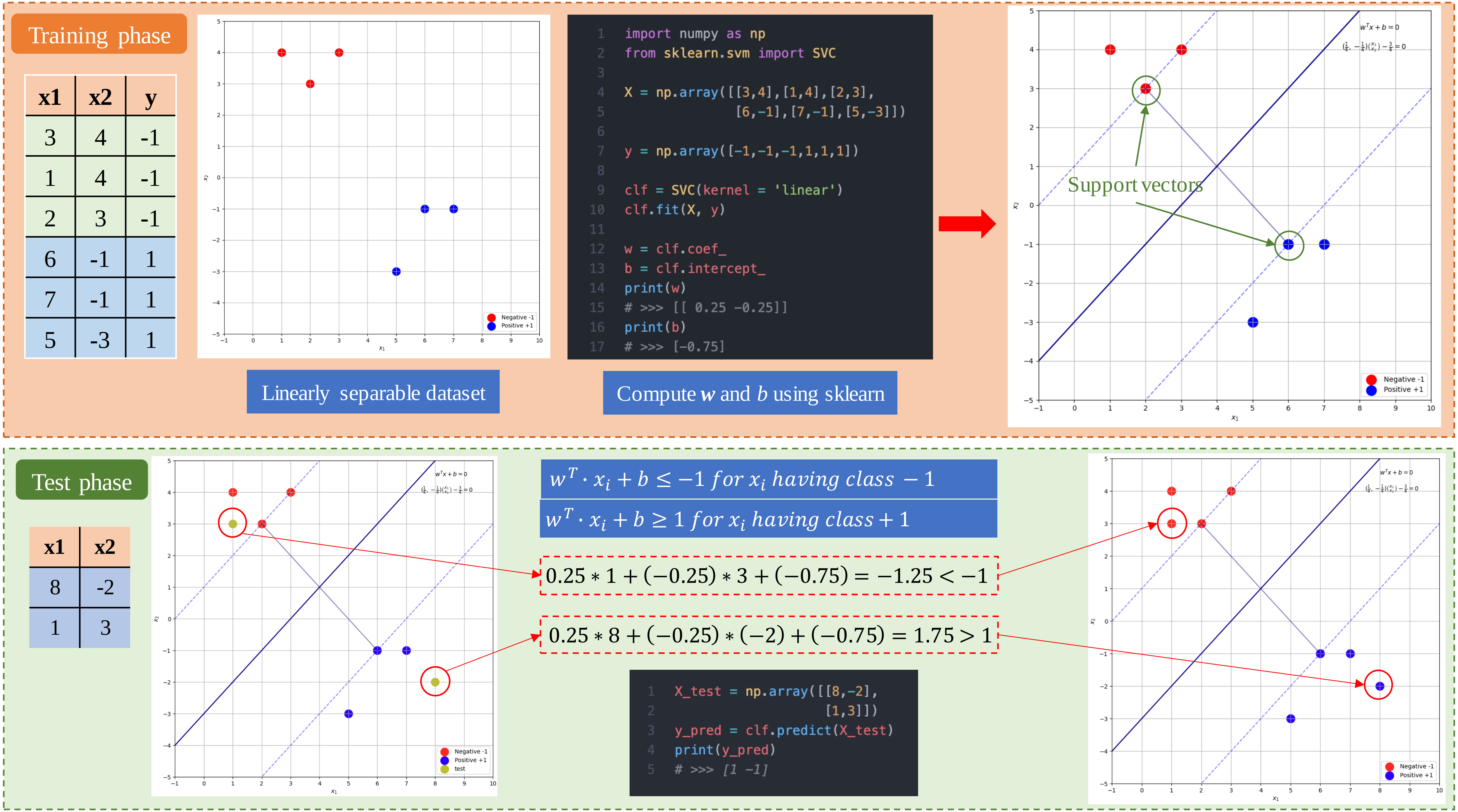
❖ SVC: Hyperplane

Changing the value of **W** gives us different hyperplanes



Where is the optimal hyperplane?

How do we find it?

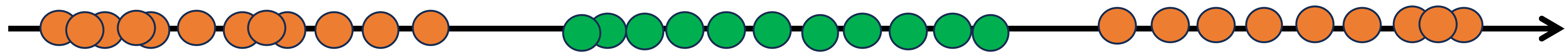


Training phase
x1 x2 y
3 4 -1
1 4 -1
2 3 -1
6 -1 1
7 -1 1
5 -3 1
Linearly separable dataset
import numpy as np
from sklearn.svm import SVC
X = np.array([[3,4],[1,4],[2,3],
[6,-1],[7,-1],[5,-3]])
y = np.array([-1,-1,-1,1,1,1])
clf = SVC(kernel = 'linear')
clf.fit(X, y)
w = clf.coef_
b = clf.intercept_
print(w)
# >>> [[0.25 -0.25]]
print(b)
# >>> [-0.75]
Compute w and b using sklearn
Support vectors
$w^Tx + b = 0$
Negative -1
Positive +1
Test phase
x1 x2
8 -2
1 3
test
$w^T \cdot x_i + b \leq -1$ for x_i having class -1
$w^T \cdot x_i + b \geq 1$ for x_i having class $+1$
$0.25 * 1 + (-0.25) * 3 + (-0.75) = -1.25 < -1$
$0.25 * 8 + (-0.25) * (-2) + (-0.75) = 1.75 > 1$
X_test = np.array([[8,-2],
[1,3]])
y_pred = clf.predict(X_test)
print(y_pred)
# >>> [1 -1]

Support Vector Machine

❖ SVC Problem

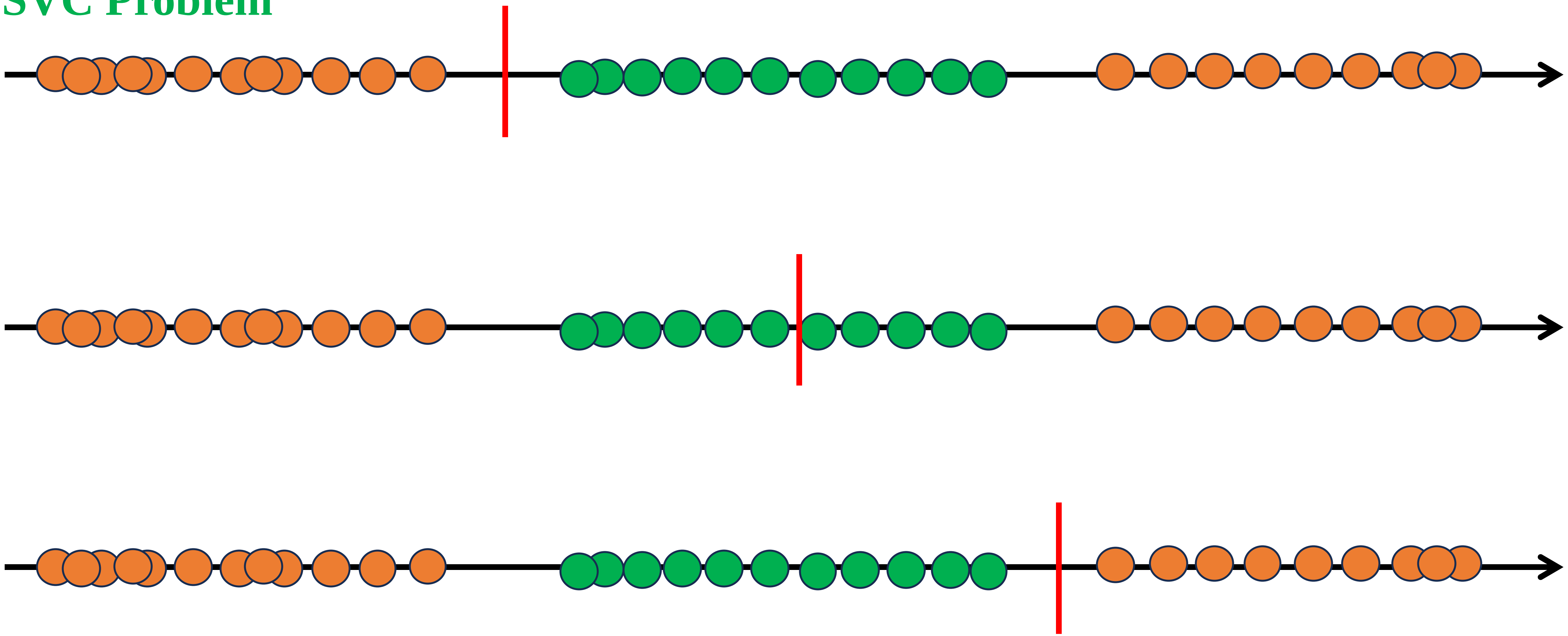
1-Dimensional Space



Can SVC handle this kind of data?

Support Vector Machine

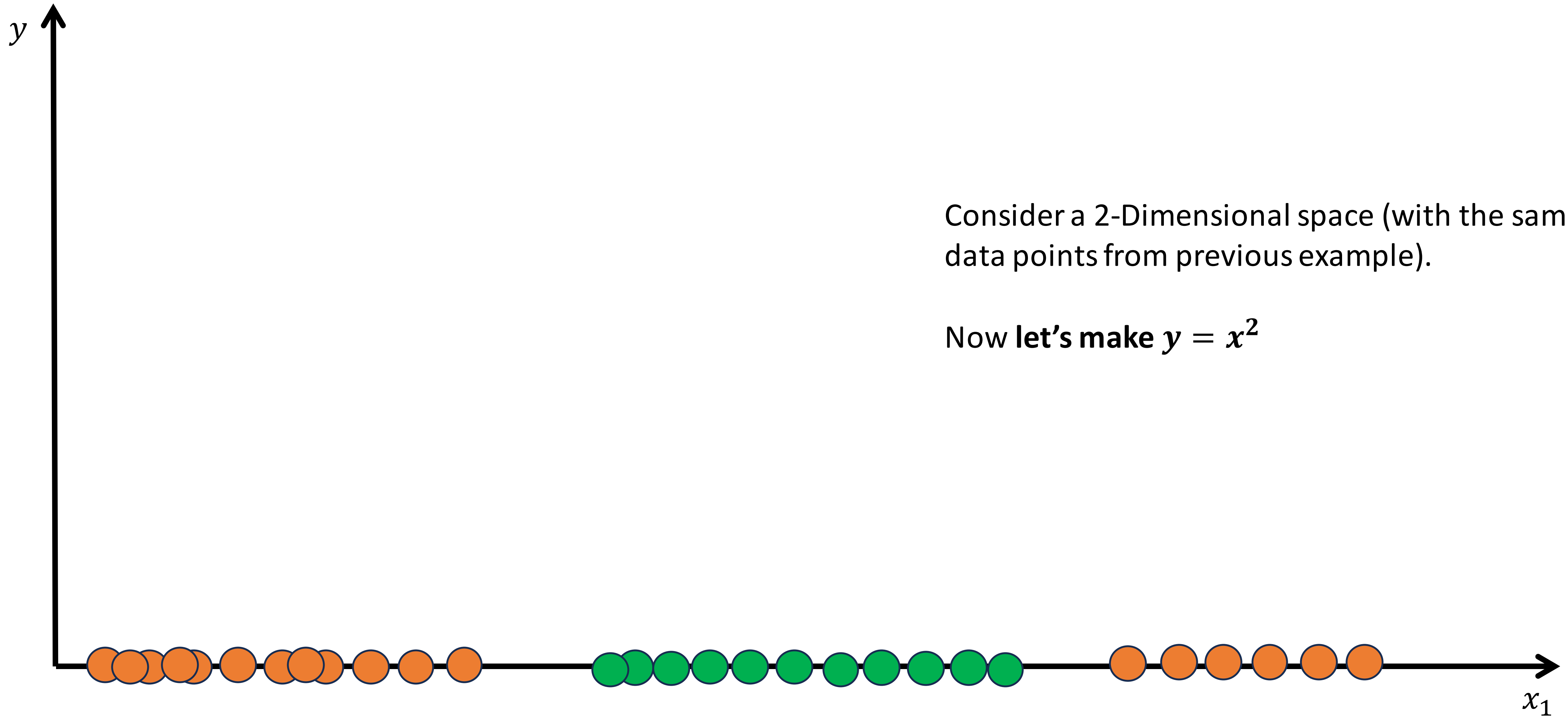
❖ SVC Problem



In general, it is hard for Hard/Soft Margin Classifier to handle this kind of data

Support Vector Machine

❖ SVM Idea

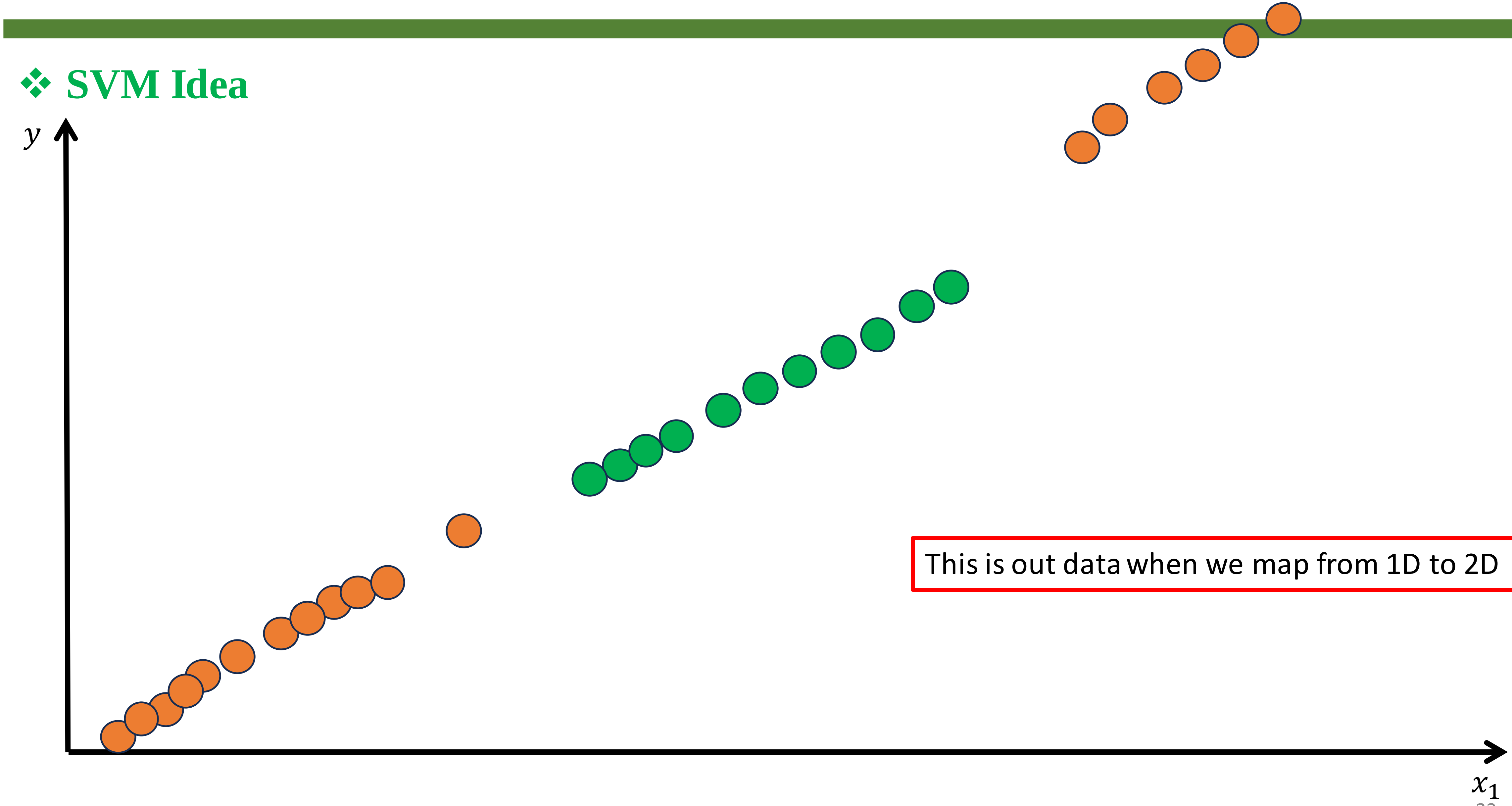


Consider a 2-Dimensional space (with the same data points from previous example).

Now **let's make** $\boldsymbol{y = x^2}$

Support Vector Machine

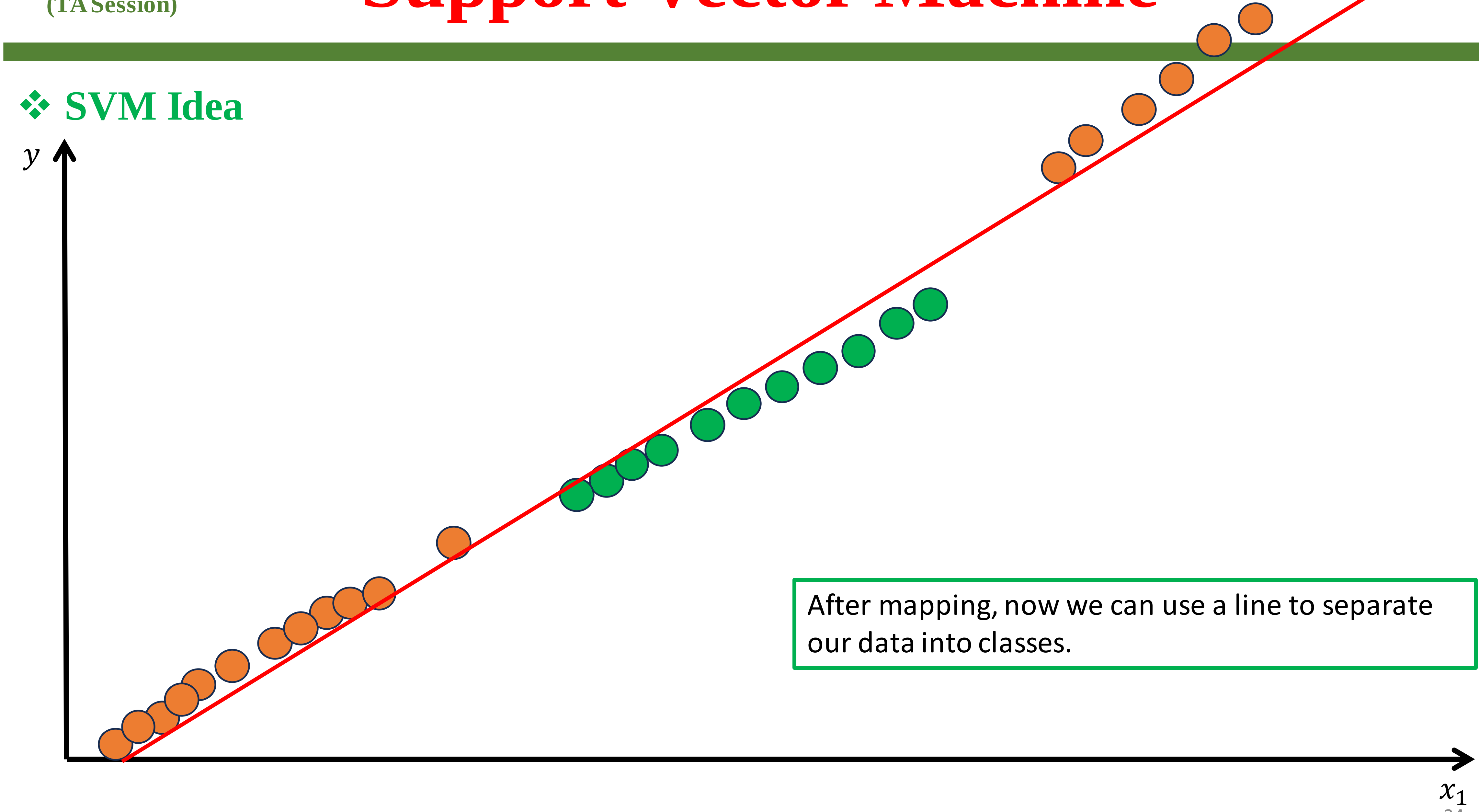
❖ SVM Idea



This is out data when we map from 1D to 2D

Support Vector Machine

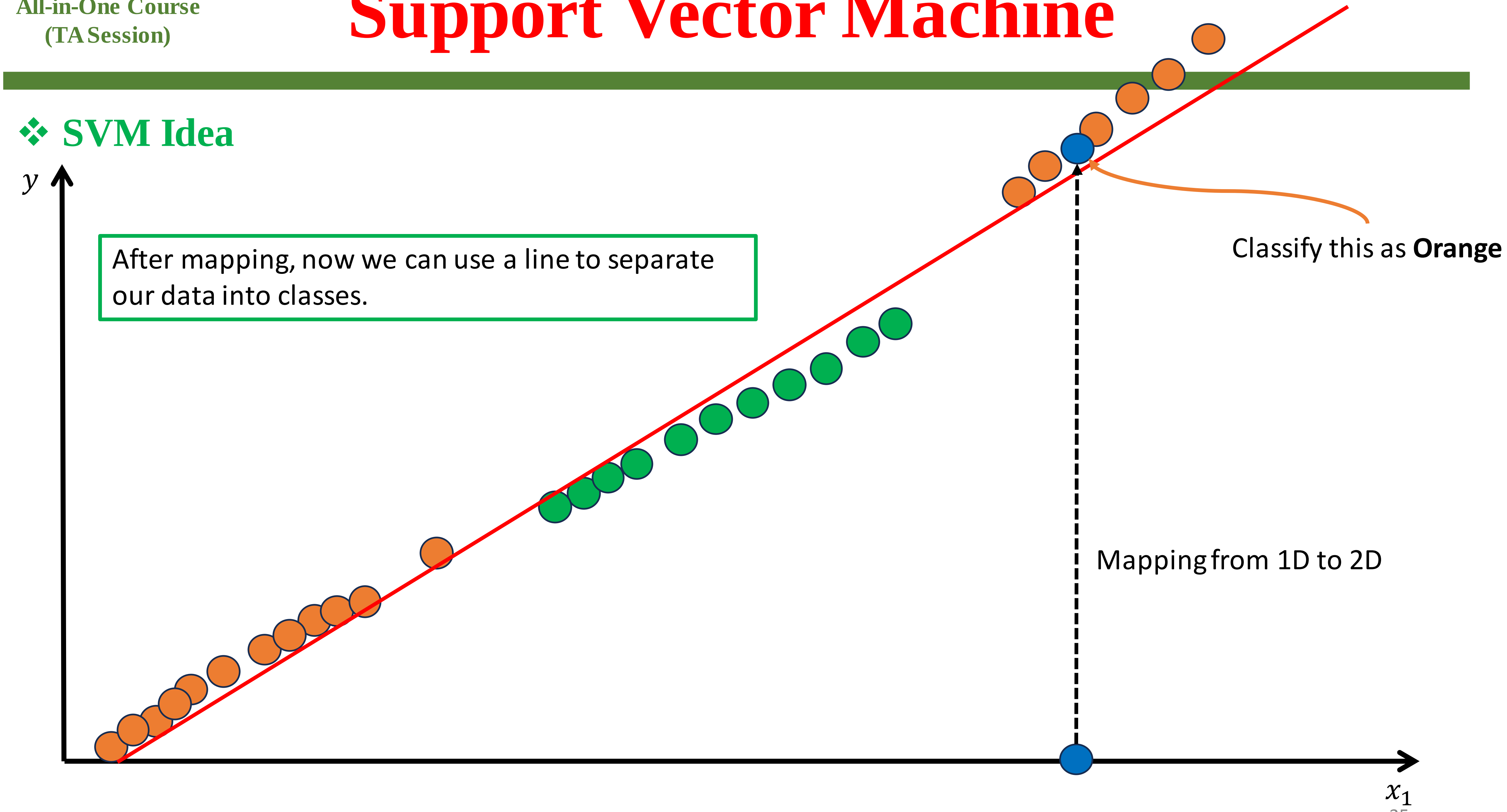
❖ SVM Idea



After mapping, now we can use a line to separate our data into classes.

Support Vector Machine

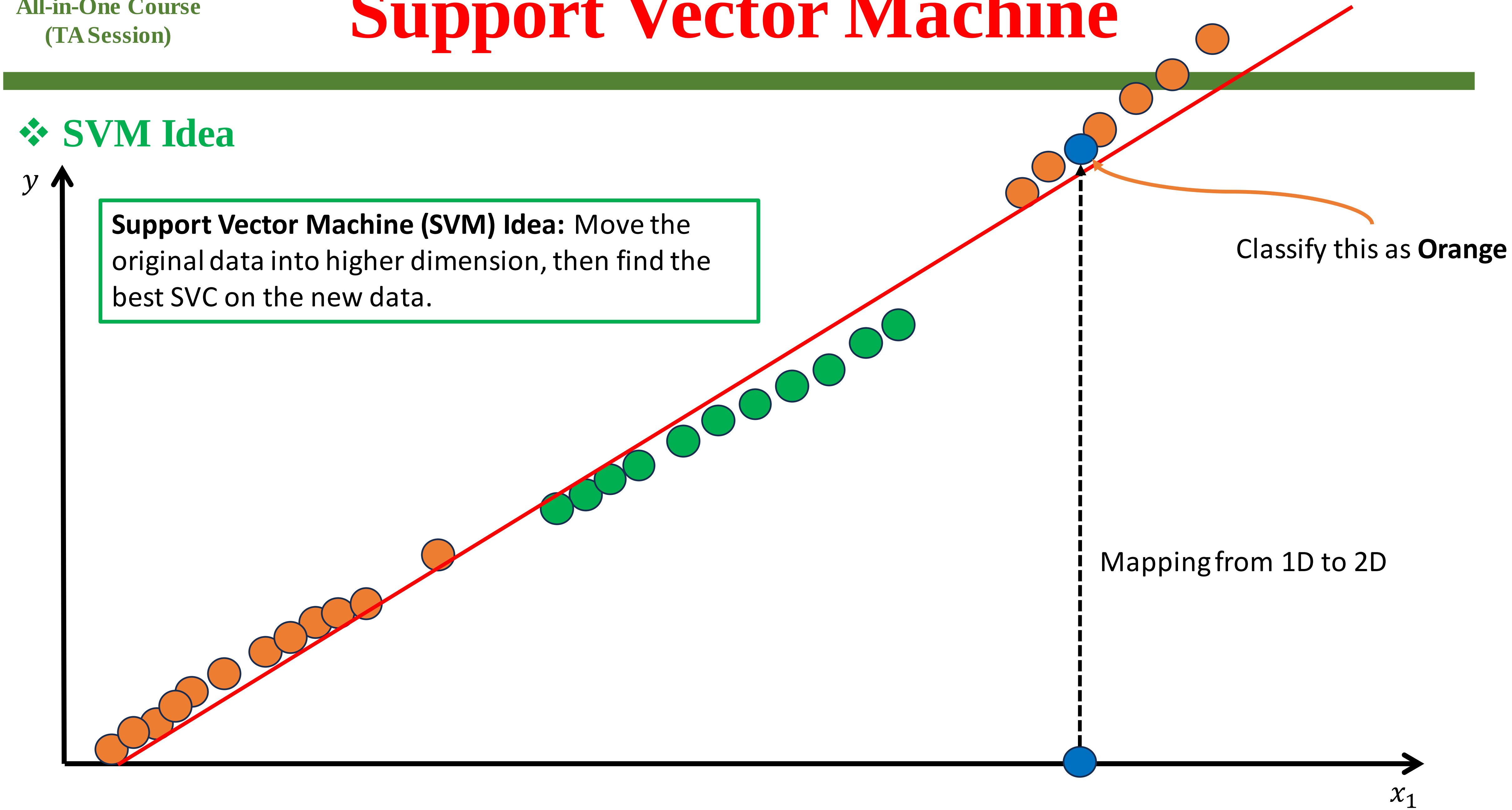
❖ SVM Idea



Support Vector Machine

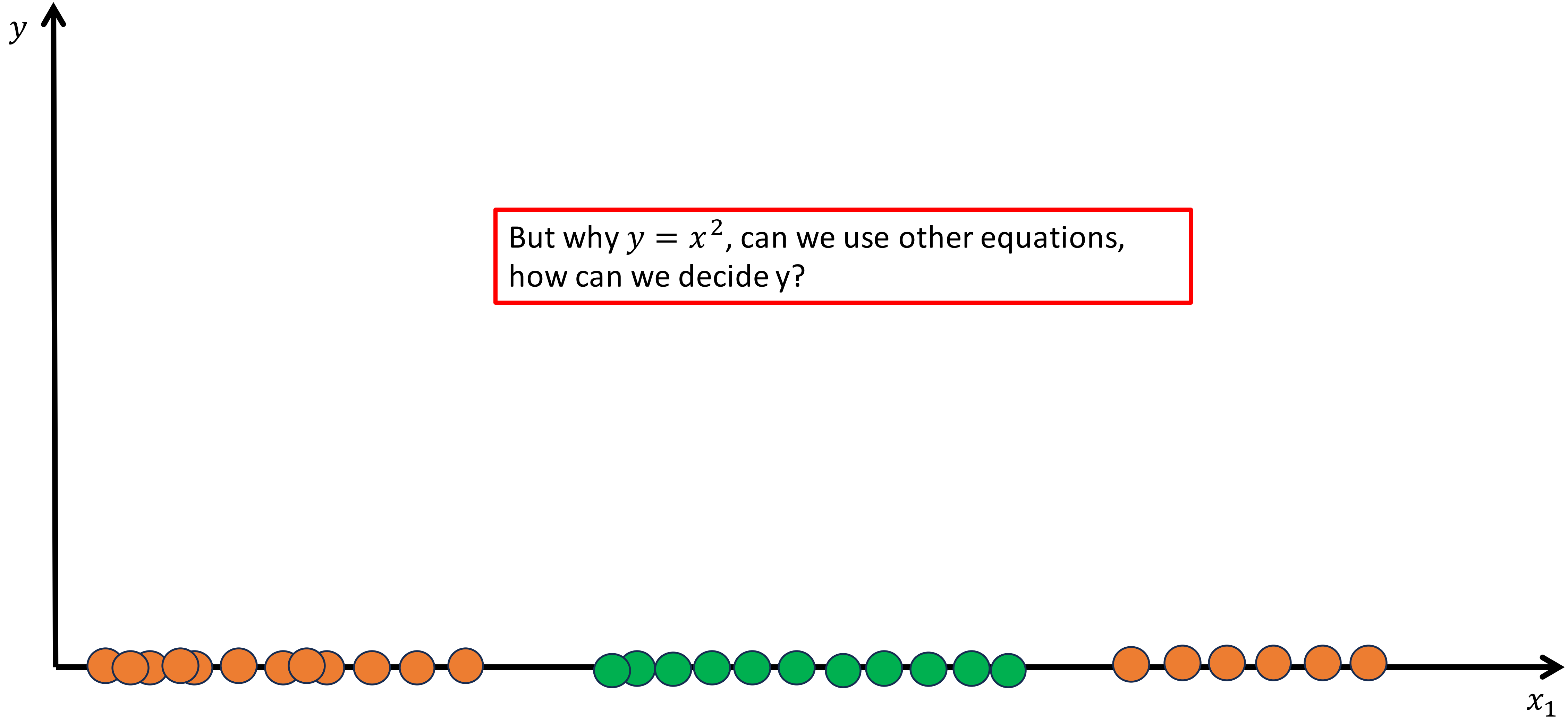
❖ SVM Idea

Support Vector Machine (SVM) Idea: Move the original data into higher dimension, then find the best SVC on the new data.



Support Vector Machine

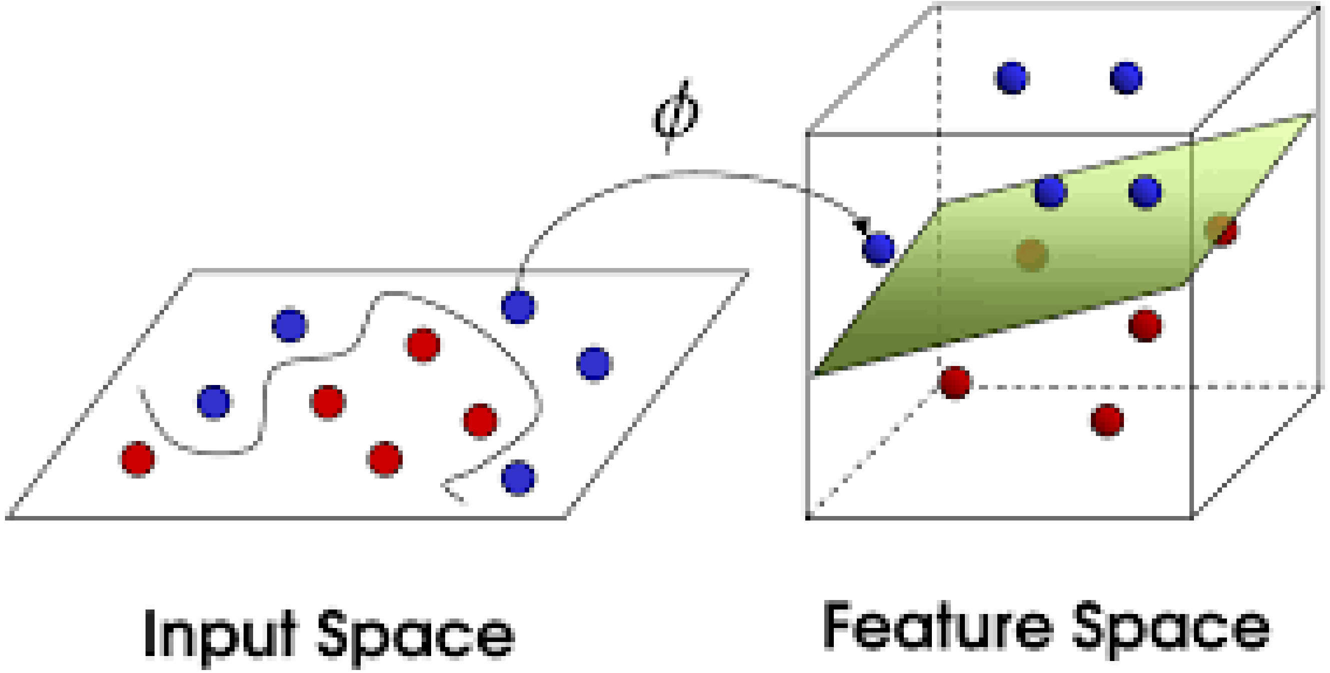
❖ SVM Idea



But why $y = x^2$, can we use other equations, how can we decide y?

Support Vector Machine

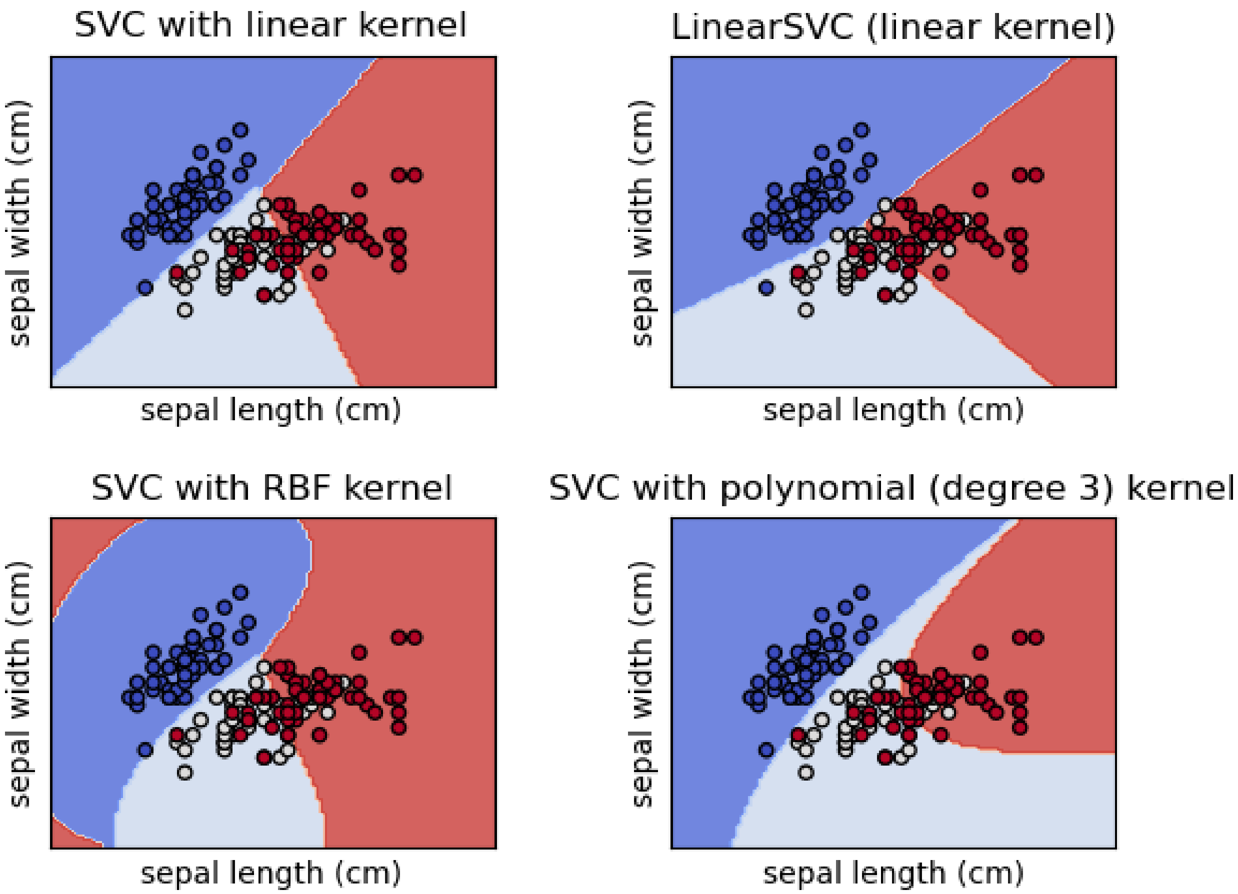
❖ Kernel



To decide the y, or to decide SVC in higher dimensions, we use **Kernel Functions.**

Support Vector Machine

❖ Type of kernels



Different results from different kernels using sklearn

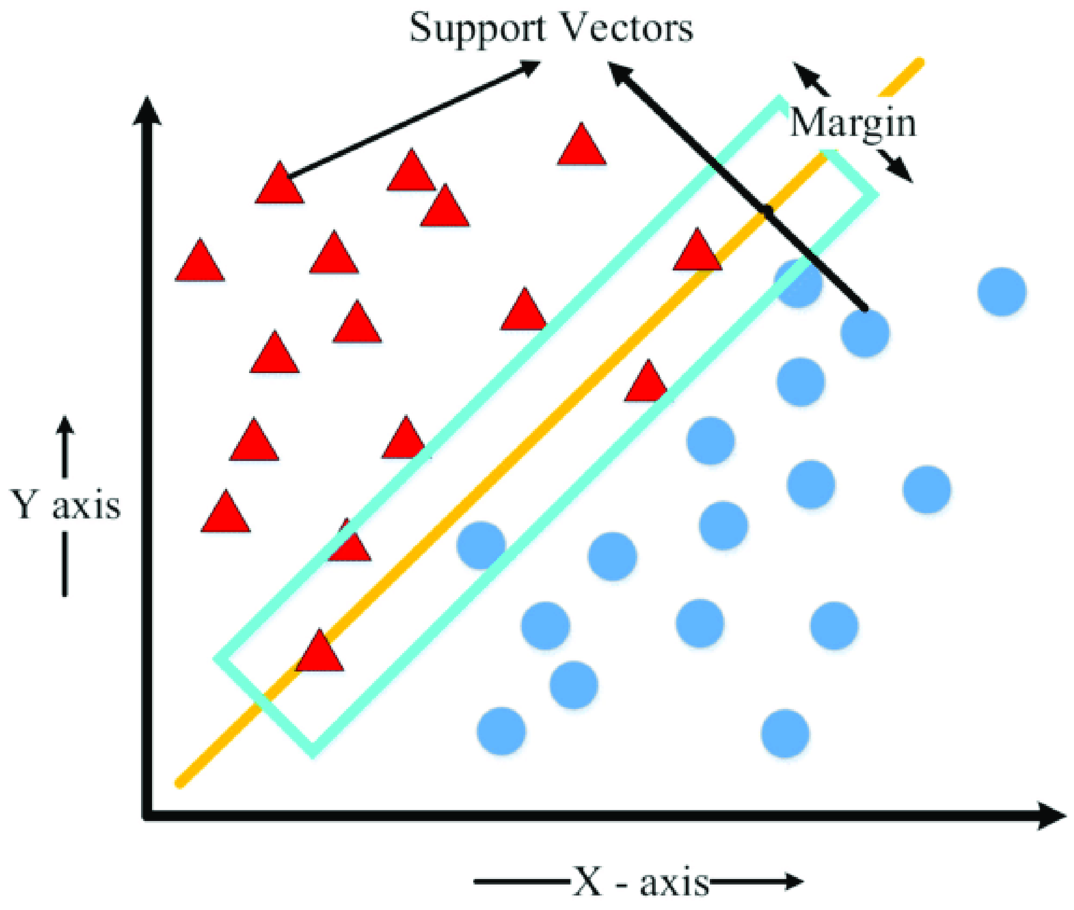
In general, we have some kernel types:

- Linear
- Polynomial
- Radial Basis Function (RBF)
- Sigmoid

Code Examples

❖ Introduction

Description: Build a binary classifier with SVM using scikit-learn library.



Code Examples

❖ Step 1: Import libraries

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Code Examples

❖ Step 2: Download and load dataset

1. Download the dataset here.
2. Using pandas.read_csv() to read the dataset.

```
dataset_path = './linear_separable_bin_clf.csv'
df = pd.read_csv(
    dataset_path,
    names=['label', 'features_1', 'features_2']
)
df
```

	label	features_1	features_2
0	1.0	2.6487	4.5192
1	1.0	1.5438	2.4443
2	1.0	1.8990	4.2409
3	1.0	2.4711	5.8097
4	1.0	3.3590	6.4423
...	...	...	...
95	-1.0	7.3641	5.9868
96	-1.0	6.2592	4.6711
97	-1.0	8.3703	7.5810
98	-1.0	8.5676	4.6457
99	-1.0	8.1676	4.6457

100 rows × 3 columns

Code Examples

❖ Step 3: Get some detail information

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   label       100 non-null    float64
 1   features_1  100 non-null    float64
 2   features_2  100 non-null    float64
dtypes: float64(3)
memory usage: 2.5 KB
```

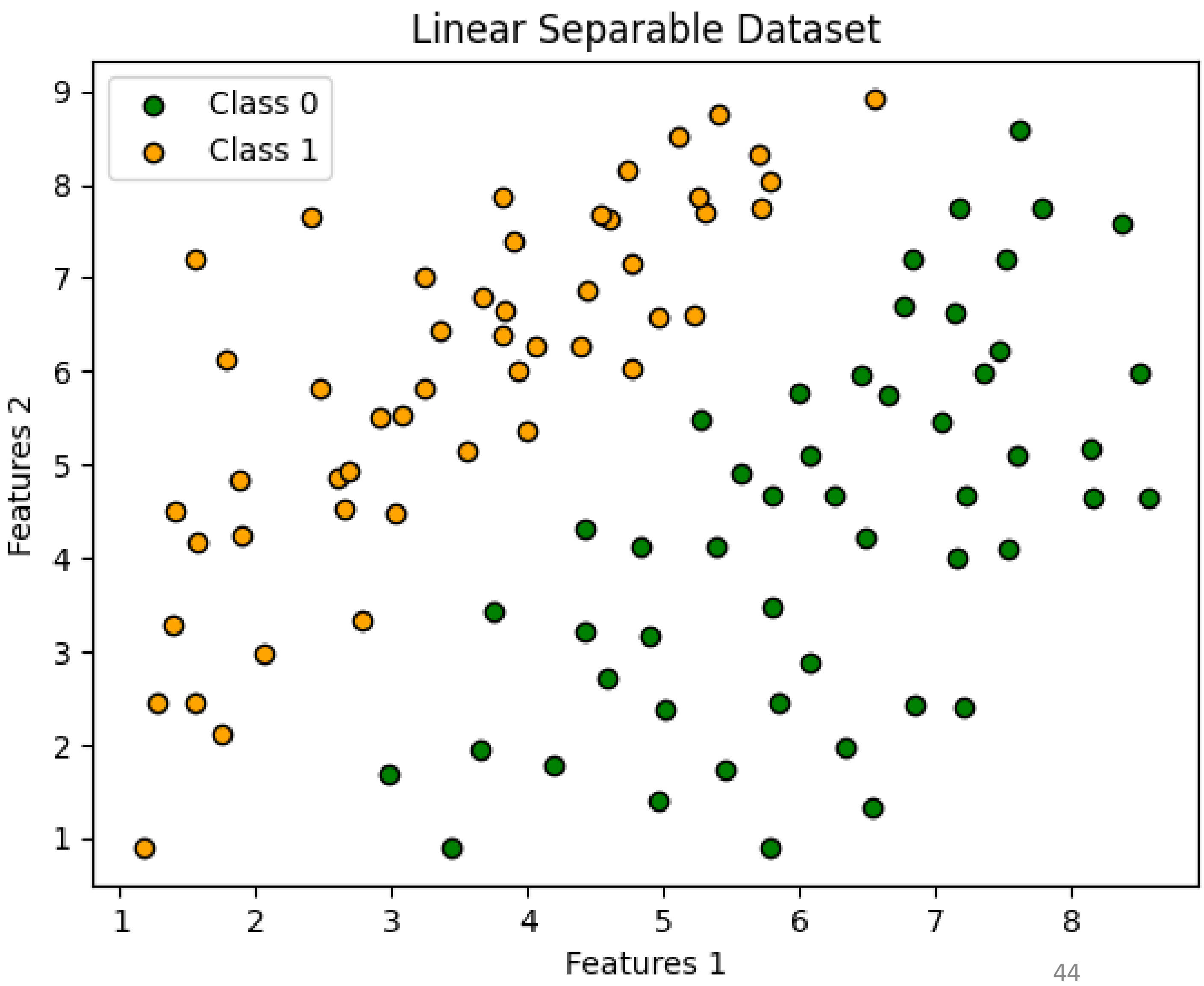
```
1 df.describe()
```

	label	features_1	features_2
count	100.000000	100.000000	100.000000
mean	0.000000	4.866669	5.144230
std	1.005038	1.964860	2.103965
min	-1.000000	1.169000	0.900800
25%	-1.000000	3.418175	3.469150
50%	0.000000	4.927500	5.265700
75%	1.000000	6.466375	6.720625
max	1.000000	8.567600	8.922100

Code Examples

❖ Step 4: Plot the dataset

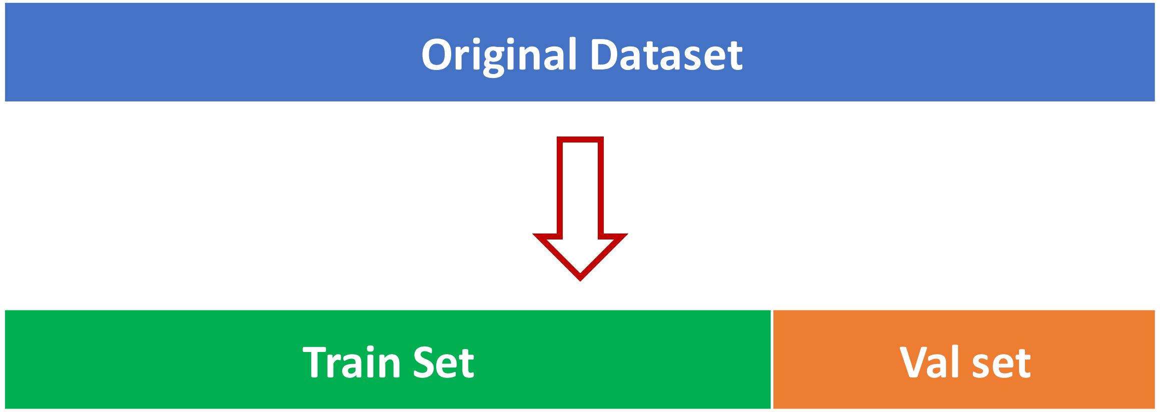
```python
class_0 = df[df['label'] == -1]
class_1 = df[df['label'] == 1]
plt.scatter(
    class_0['features_1'],
    class_0['features_2'],
    edgecolor="black",
    marker='o',
    color='green',
    label='Class 0'
)
plt.scatter(
    class_1['features_1'],
    class_1['features_2'],
    edgecolor="black",
    marker='o',
    color='orange',
    label='Class 1'
)
plt.xlabel('Features 1')
plt.ylabel('Features 2')
plt.title('Linear Separable Dataset')
plt.legend()
plt.show()
```



Code Examples

❖ Step 5: Split train val dataset

```
dataset_arr = df.to_numpy()
X, y = dataset_arr[:, 1:], dataset_arr[:, 0]

test_size = 0.3
random_state = 1
is_shuffle = True
X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=test_size,
    random_state=random_state,
    shuffle=is_shuffle
)
```




```
print(f'Number of training samples: {X_train.shape[0]}')
print(f'Number of val samples: {X_val.shape[0]}')
```

```
Number of training samples: 70
Number of val samples: 30
```

Code Examples

❖ Step 6: Train SVM

In this problem, we do classification, so we will use SVC module.

```
classifier = SVC(
    kernel='linear',
    random_state=random_state
)
classifier.fit(X_train, y_train)
```

```
▼ SVC
SVC(kernel='linear', random_state=1)
```

sklearn.svm.SVC

class `sklearn.svm.SVC`(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `LinearSVC` or `SGDClassifier` instead, possibly after a `Nystroem` transformer or other Kernel Approximation.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how `gamma`, `coef0` and `degree` affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Read more about SVC here

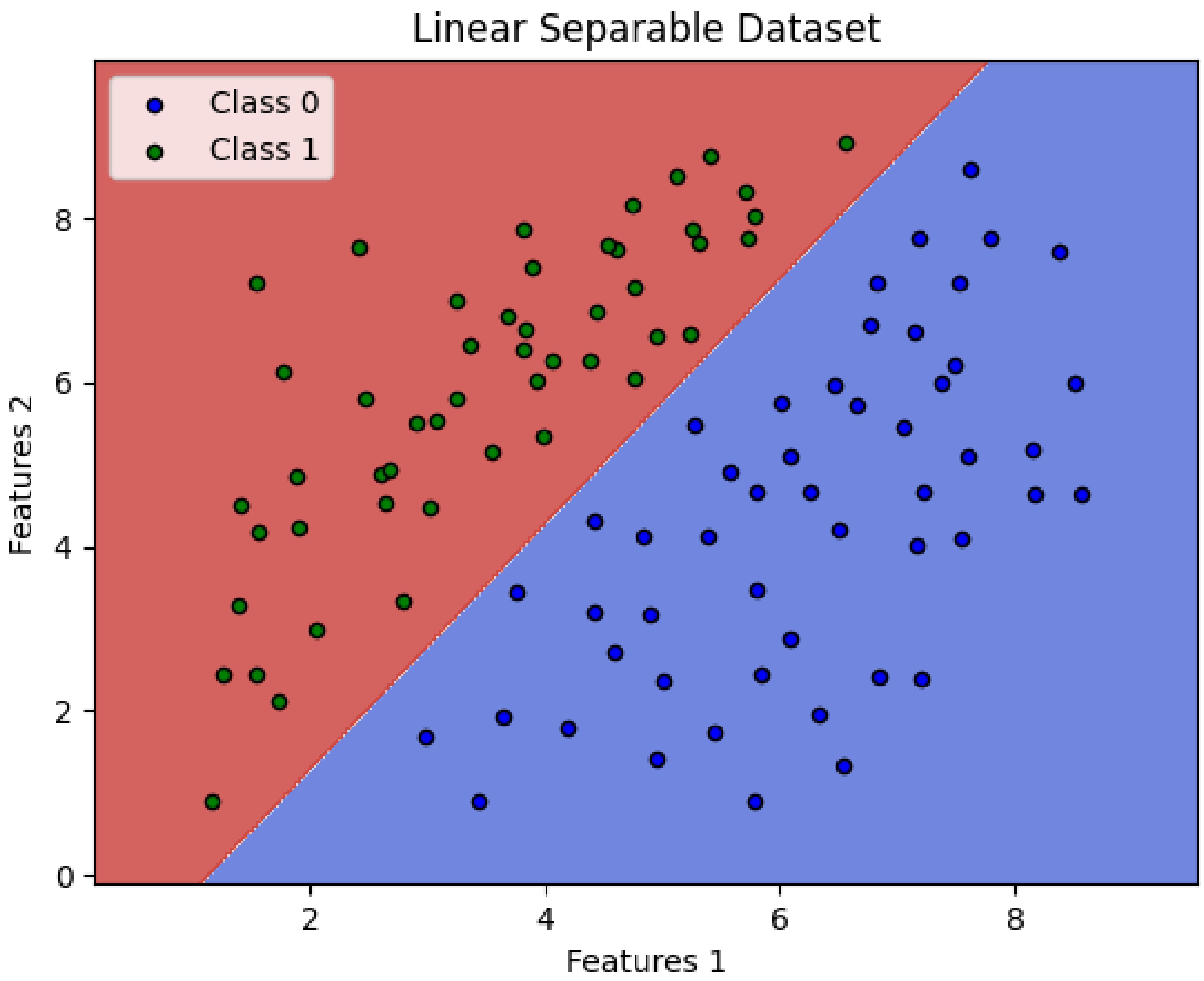
Code Examples

❖ Step 7: Evaluation

Evaluate trained SVM on val set:

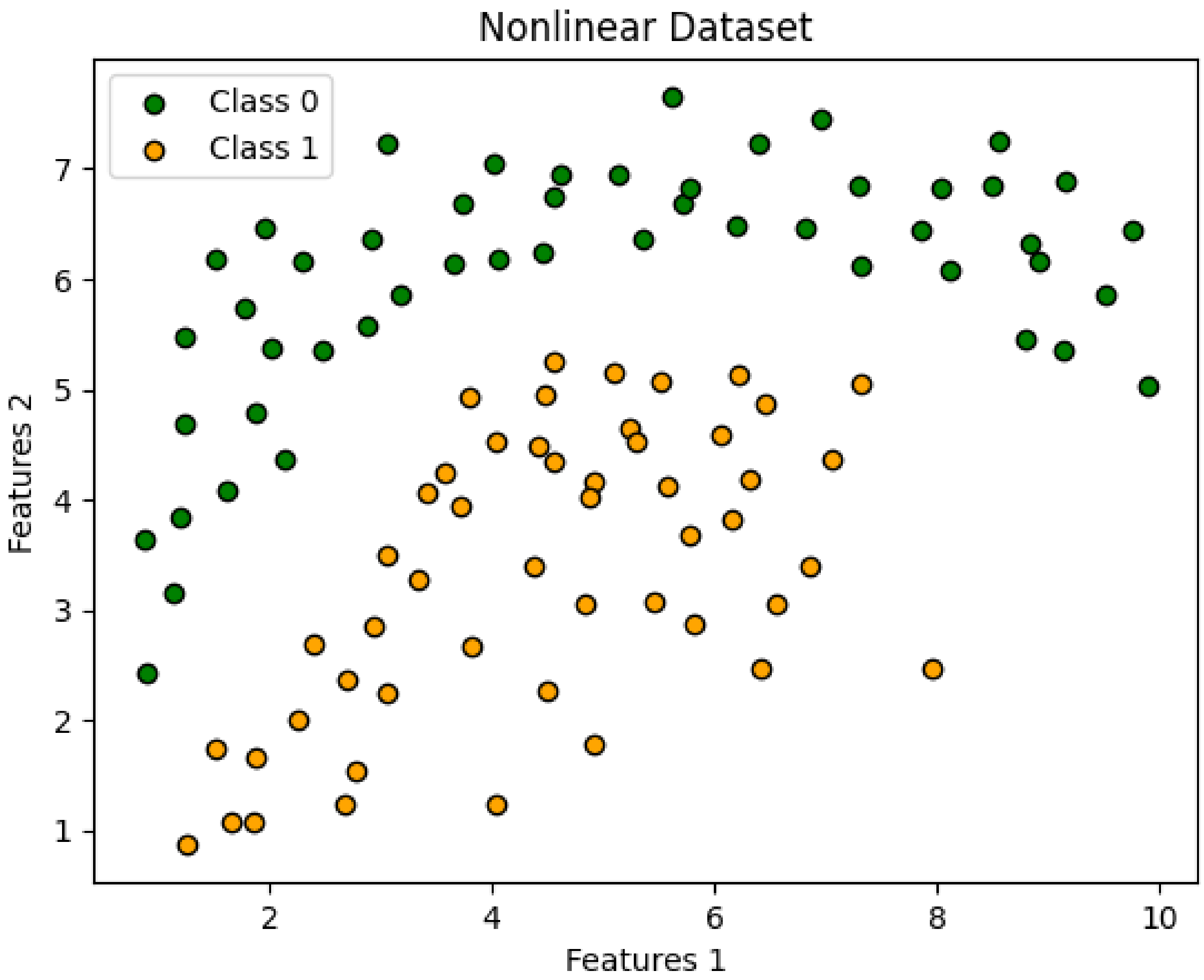
```
y_pred = classifier.predict(X_val)
scores = accuracy_score(y_pred, y_val)

print('Evaluation results on validation set:')
print(f'Accuracy: {scores}')
```

```
Evaluation results on validation set:
Accuracy: 1.0
```



Code Examples

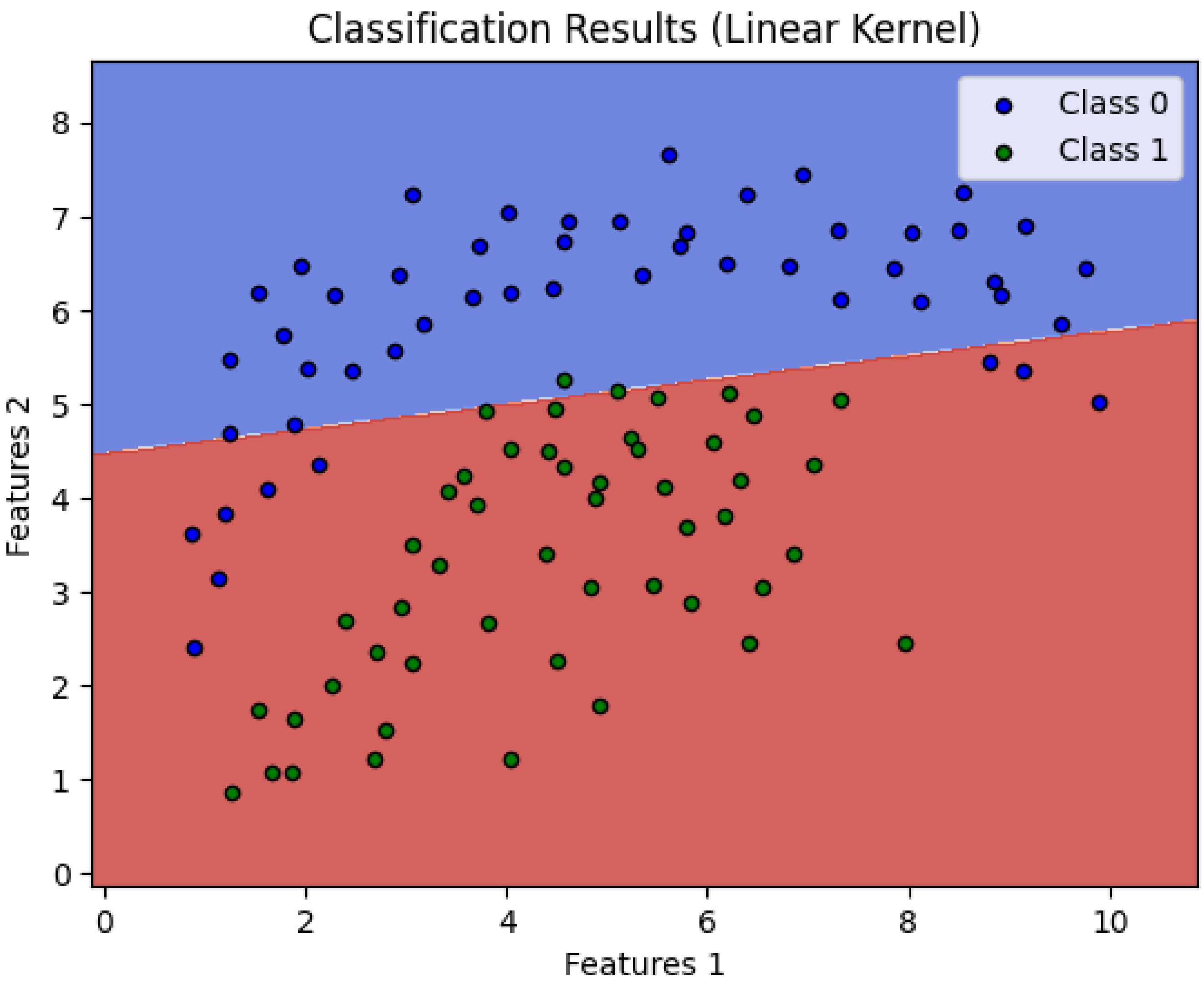
❖ Linear kernel with non-linear dataset ? (Download here)



If we apply the same previous SVC code, will it still work?

Code Examples

❖ Linear kernel with non-linear dataset ?




```python
linear_clf = SVC(
    kernel='linear',
    random_state=random_state
)
linear_clf.fit(X_train, y_train)
```

▼ SVC

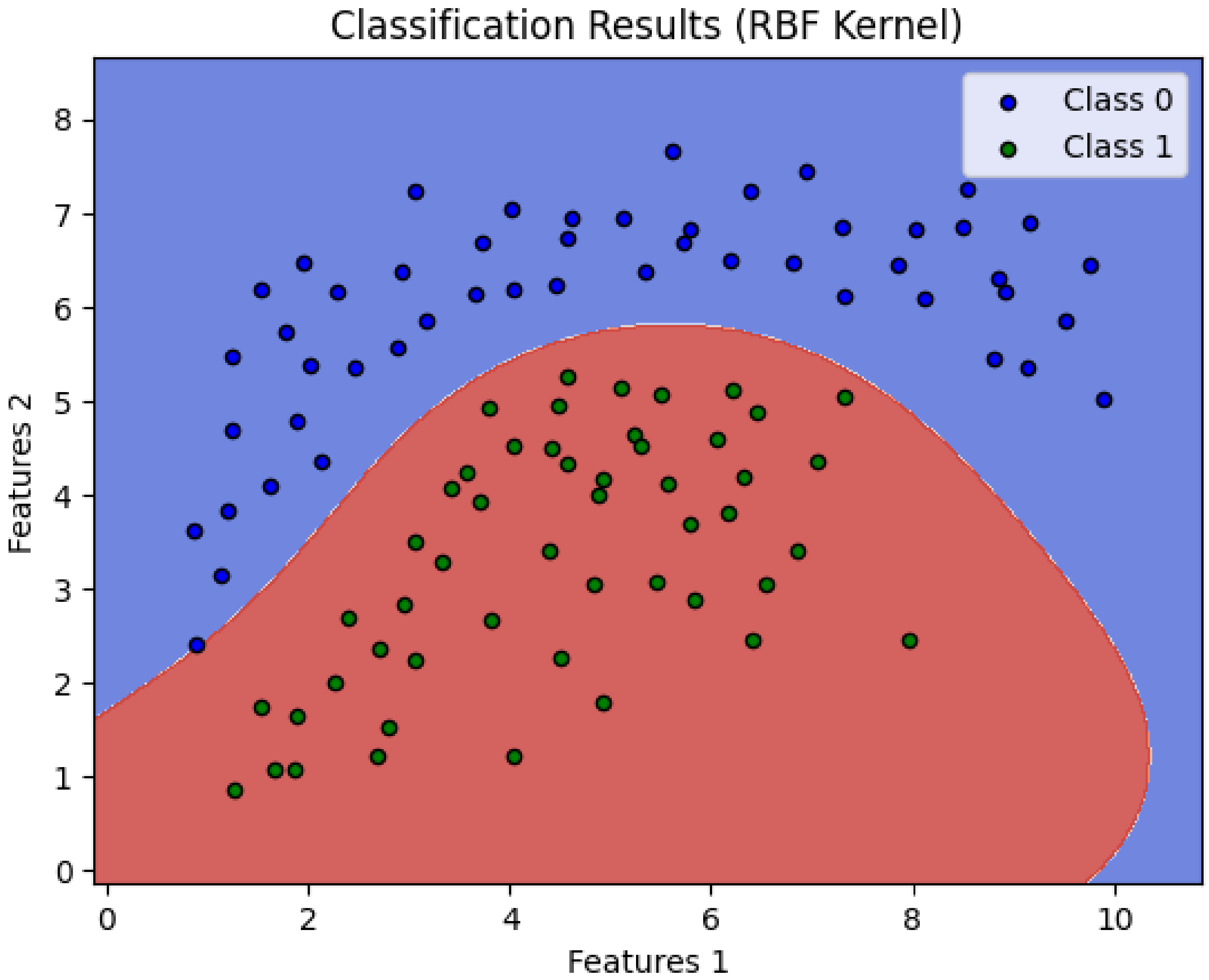
SVC(kernel='linear', random_state=1)

```python
linear_scores = accuracy_score(y_pred_linear, y_val)
print('Evaluation results on validation set:')
print(f'Linear kernel accuracy: {linear_scores}')
```

```
Evaluation results on validation set:
Linear kernel accuracy: 0.8666666666666667
```

Code Examples

❖ How about other kernels ?




```
rbf_clf = SVC(
    kernel='rbf',
    random_state=random_state
)
rbf_clf.fit(X_train, y_train)
```

```
SVC
SVC(random_state=1)
```

```
rbf_scores = accuracy_score(y_pred_rbf, y_val)
print('Evaluation results on validation set:')
print(f'RBF kernel accuracy: {rbf_scores}')
```

```
Evaluation results on validation set:
RBF kernel accuracy: 1.0
```

It is important to tune proper hyperparameters of SVM for the best result.

Question

?


Thank you!