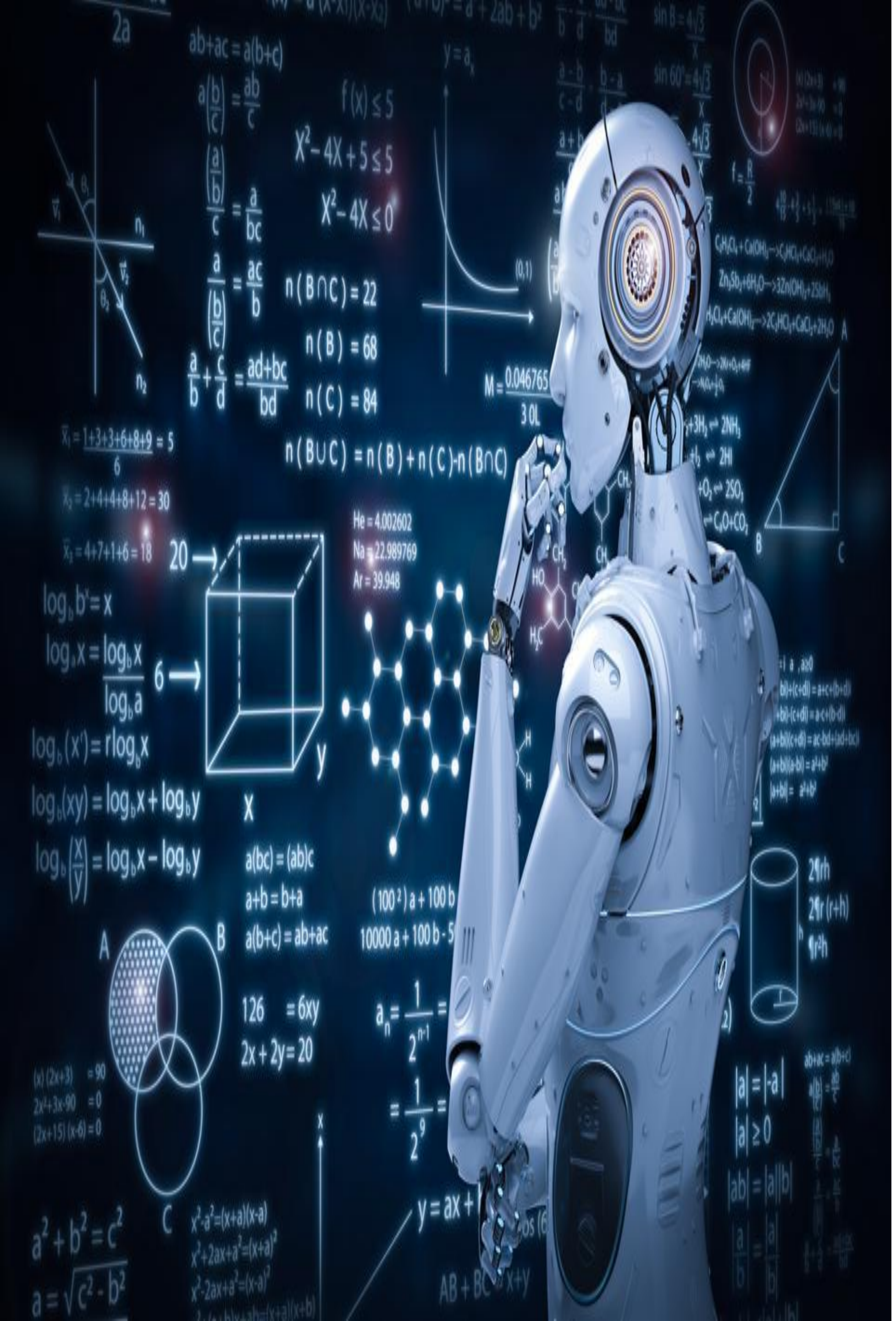


Practical Machine Learning

Day 13: Sep22 DBDA

Kiran Waghmare

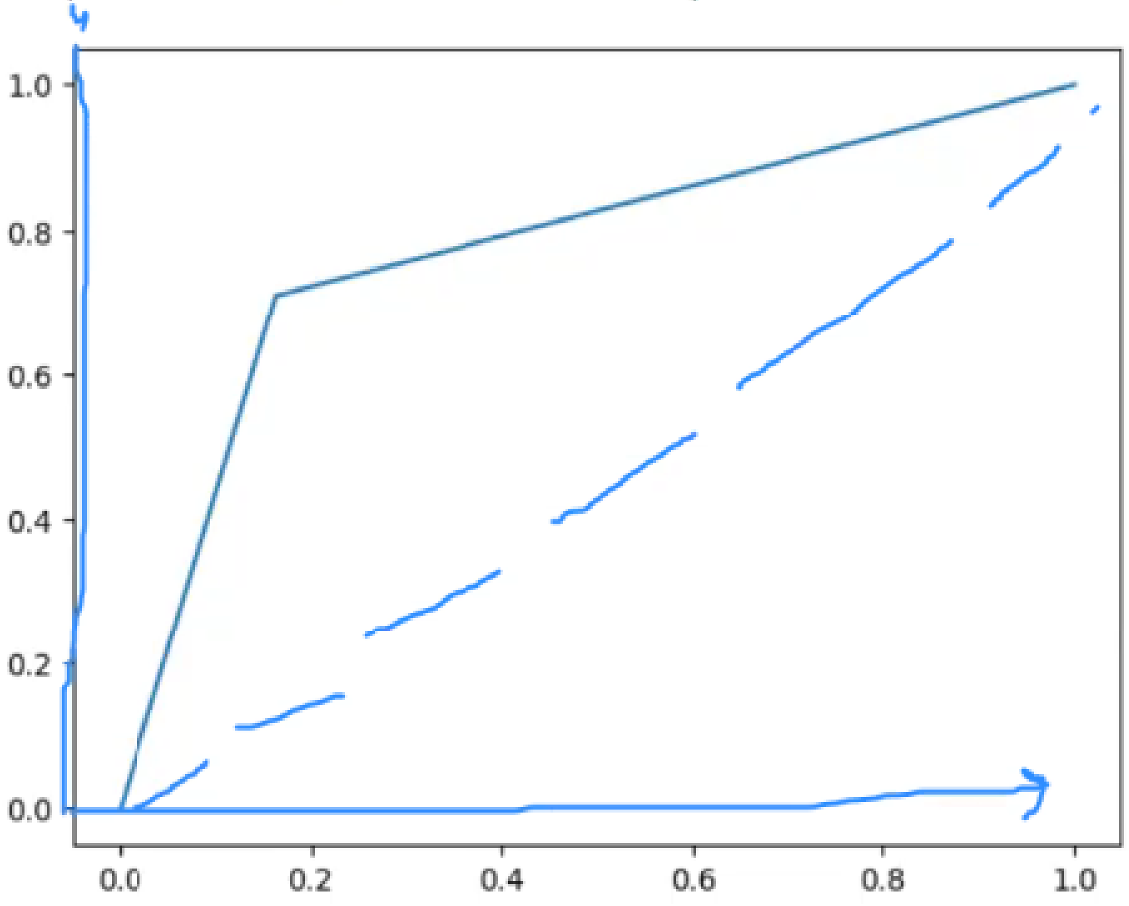
```
In [65]: from sklearn import model_selection
         from sklearn.model_selection import cross_val_score
         kfold = model_selection.KFold(n_splits=10)
         model2 = tree.DecisionTreeClassifier()
         model2

Out[65]: DecisionTreeClassifier()

In [66]: accuracy_score(y_test,y_pred)

Out[66]: 0.7877094972067039

In [68]: model3 = model_selection.cross_val_score(model2, X_train, y_train, cv=kfold)
         model3.mean()

Out[68]: 0.7894366197183098

In [ ]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm = confusion_matrix(y_test, y_pred)
         cm
```

DT

DT with cross validation

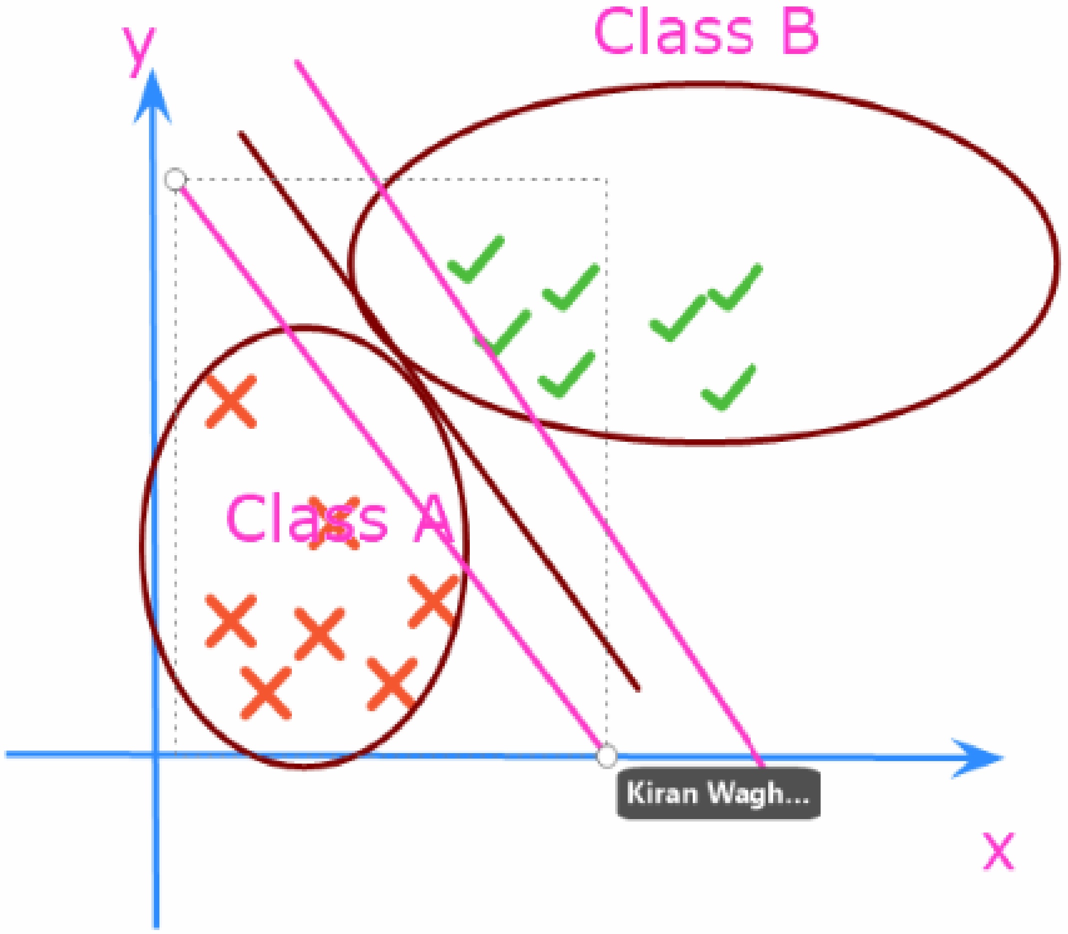
In [72]:
```
roc_auc_score(y_test,y_pred)
fpr,tpr,threshold = roc_curve(y_test, y_pred)
plt.plot(fpr,tpr)
```

Out[72]: `[<matplotlib.lines.Line2D at 0x1e04f4bfd00>]`



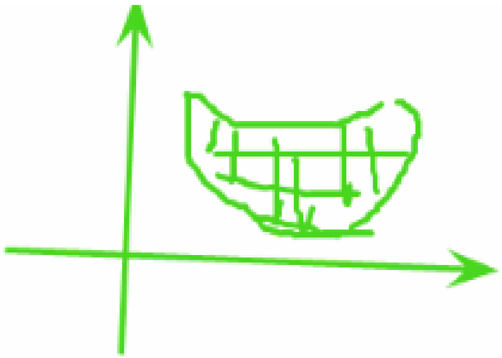
Agenda

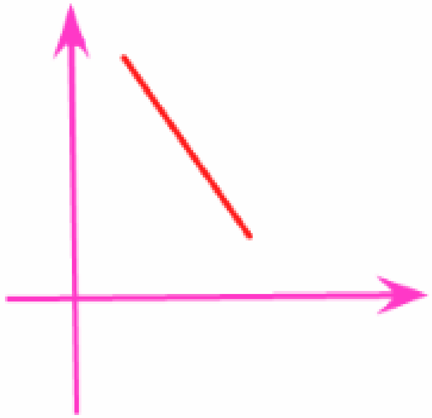
- SVM
- SVM-Kernel

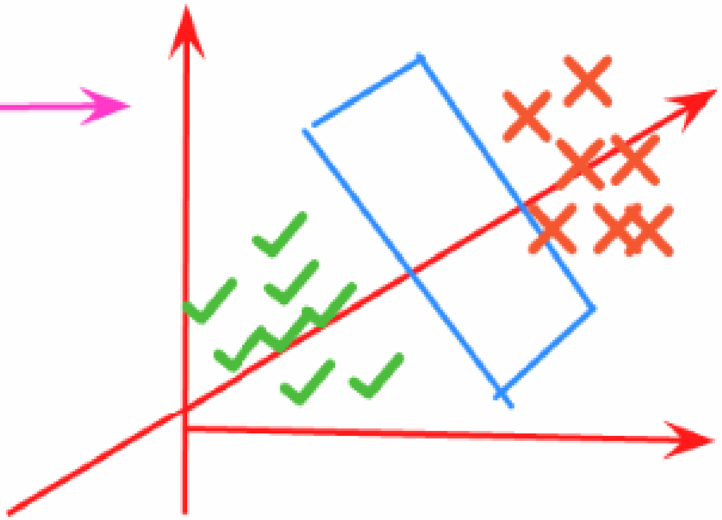


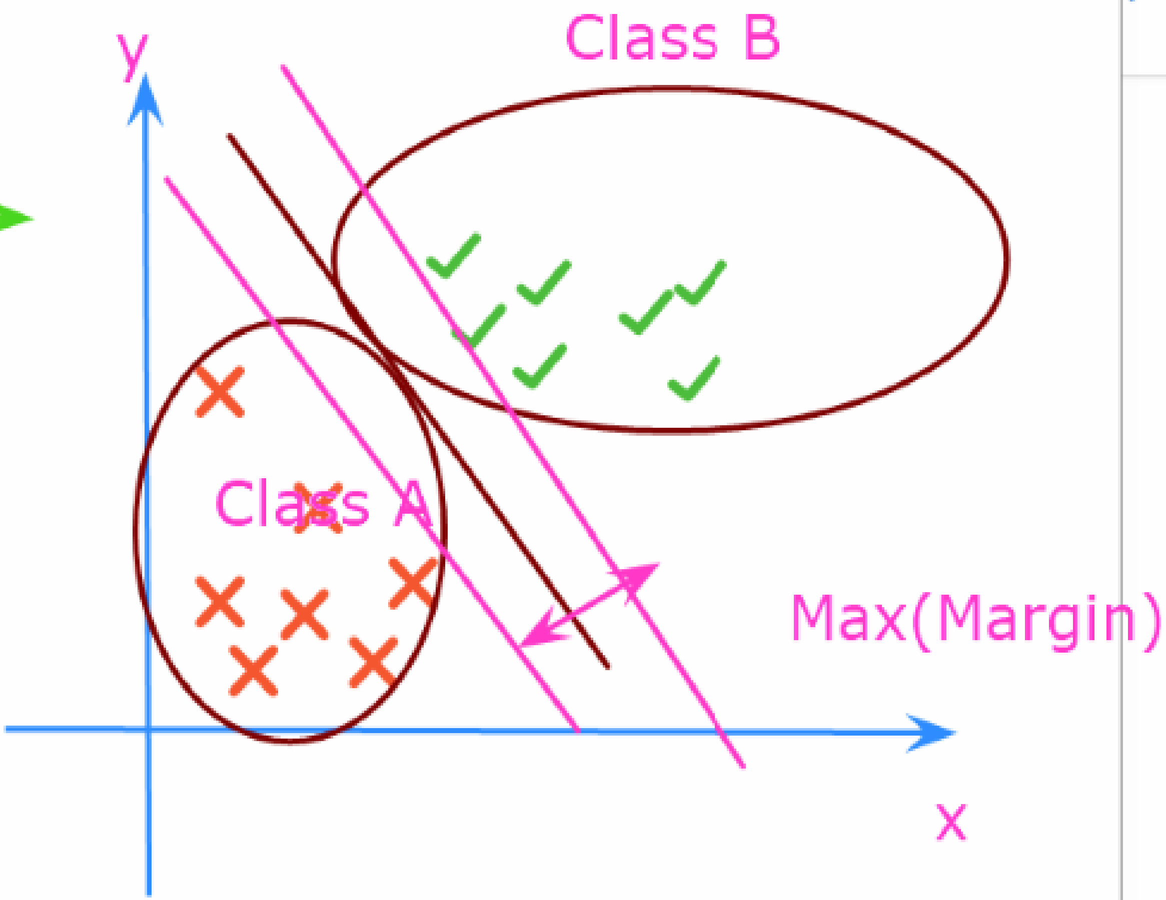
Agenda

- SVM
- SVM-Kernel





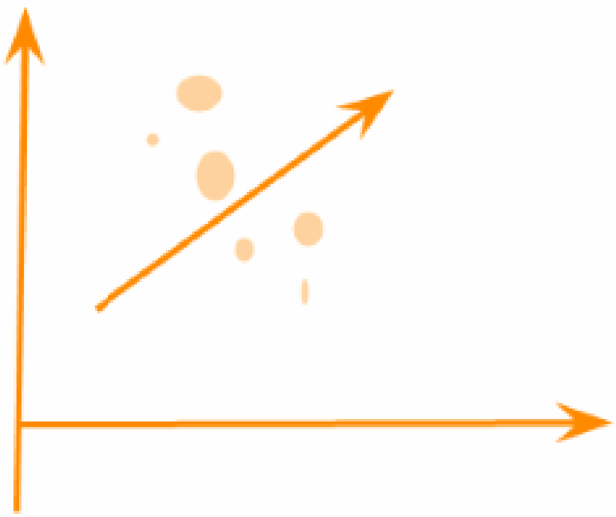


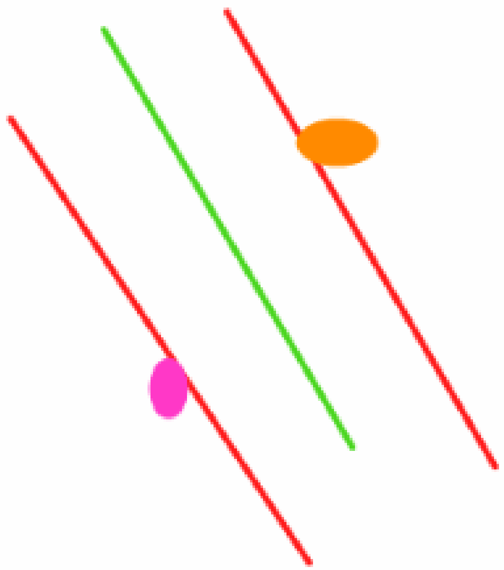


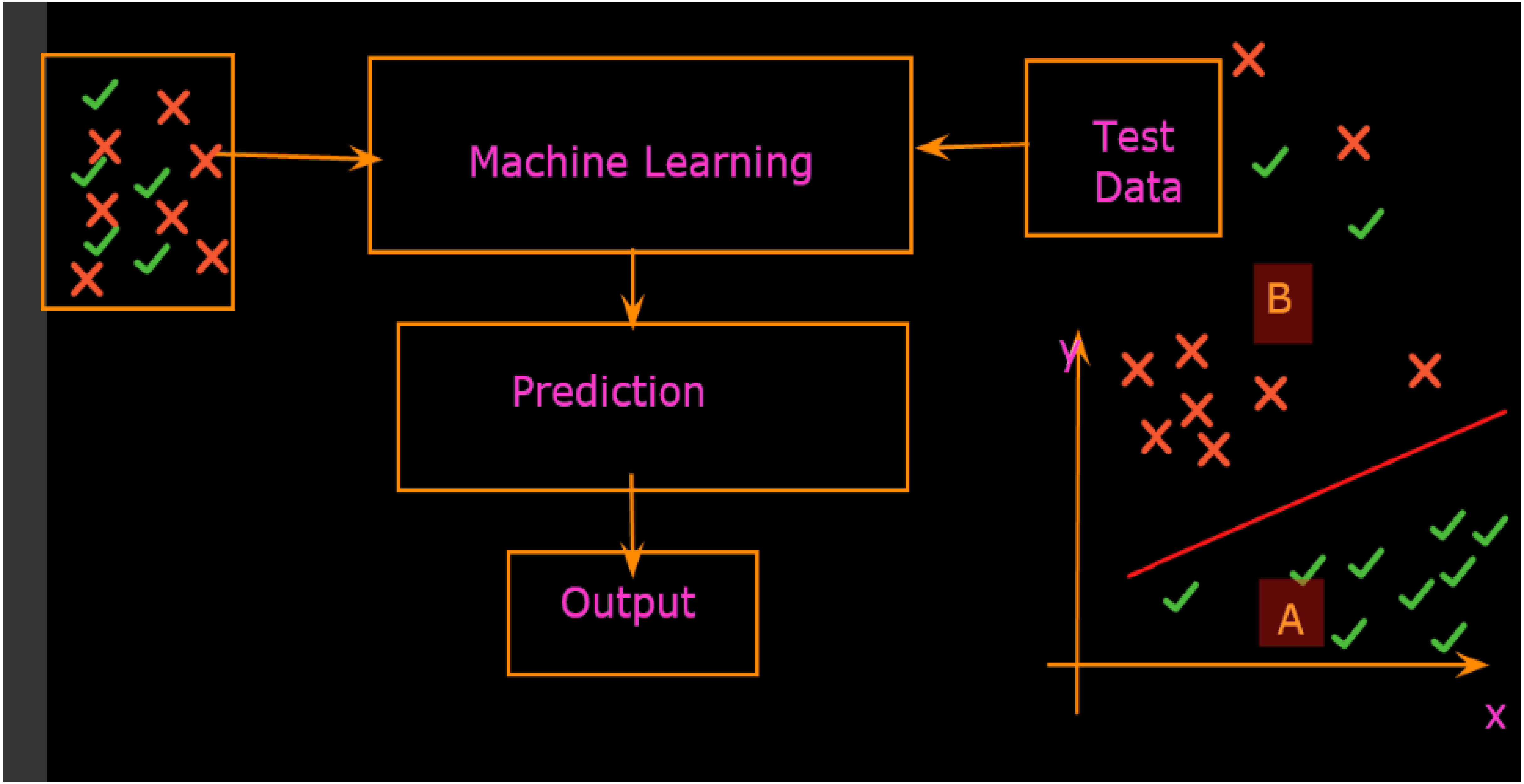
Agenda

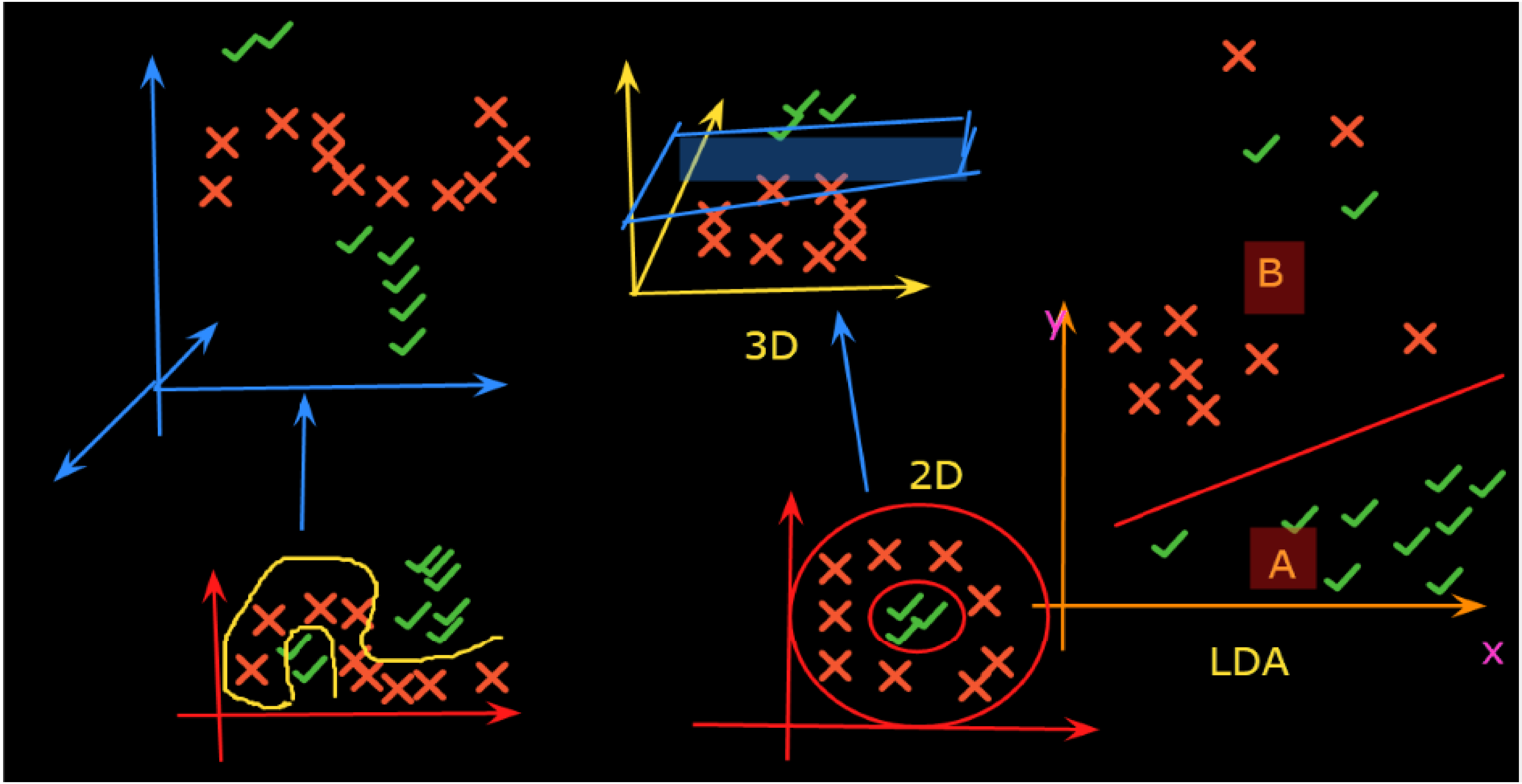
- SVM
- SVM-Kernel

kNN: similarity
DT: node bounderies
Naive Bayes: cond prob






Machine Learning
Test
Data
Prediction
Output
B
y
A
x


3D
2D
B
A
y
x
LDA

Support Vector Machine Algorithm

- **Goal :**
  - The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
  - This best decision boundary is called **a hyperplane**
  - SVM chooses the extreme points/vectors that help in creating the hyperplane
  - These extreme cases are called as **support vectors**

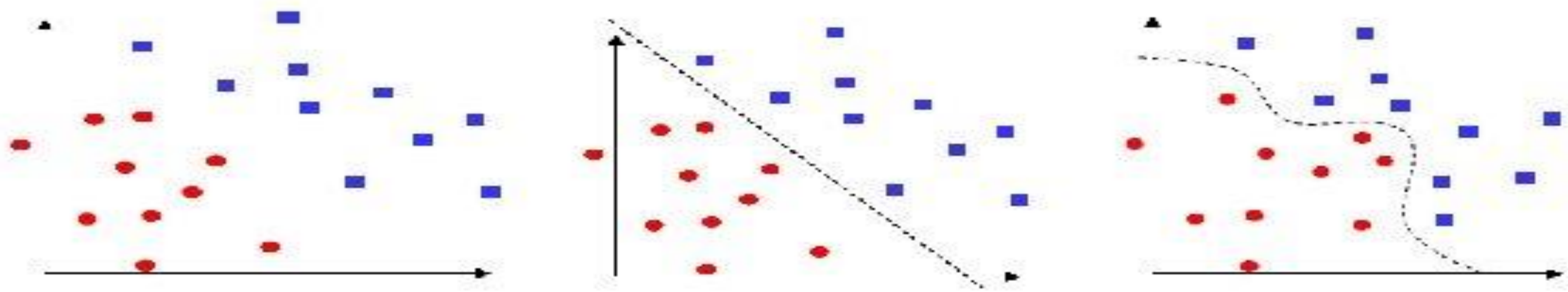
    and hence algorithm is termed as Support Vector Machine
- . Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

Support Vector Machine (SVM)

-- classifier, forward neural network, supervised learning

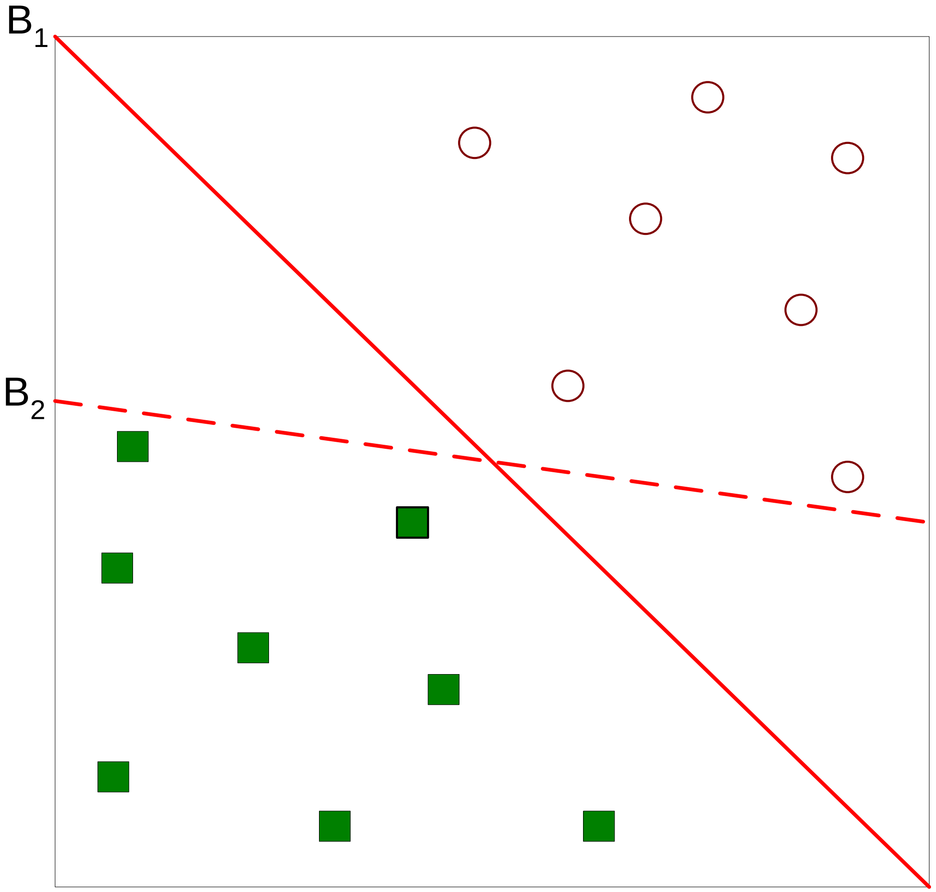
Difficulties with SVM:

i) binary classifier, ii) linearly separable patterns



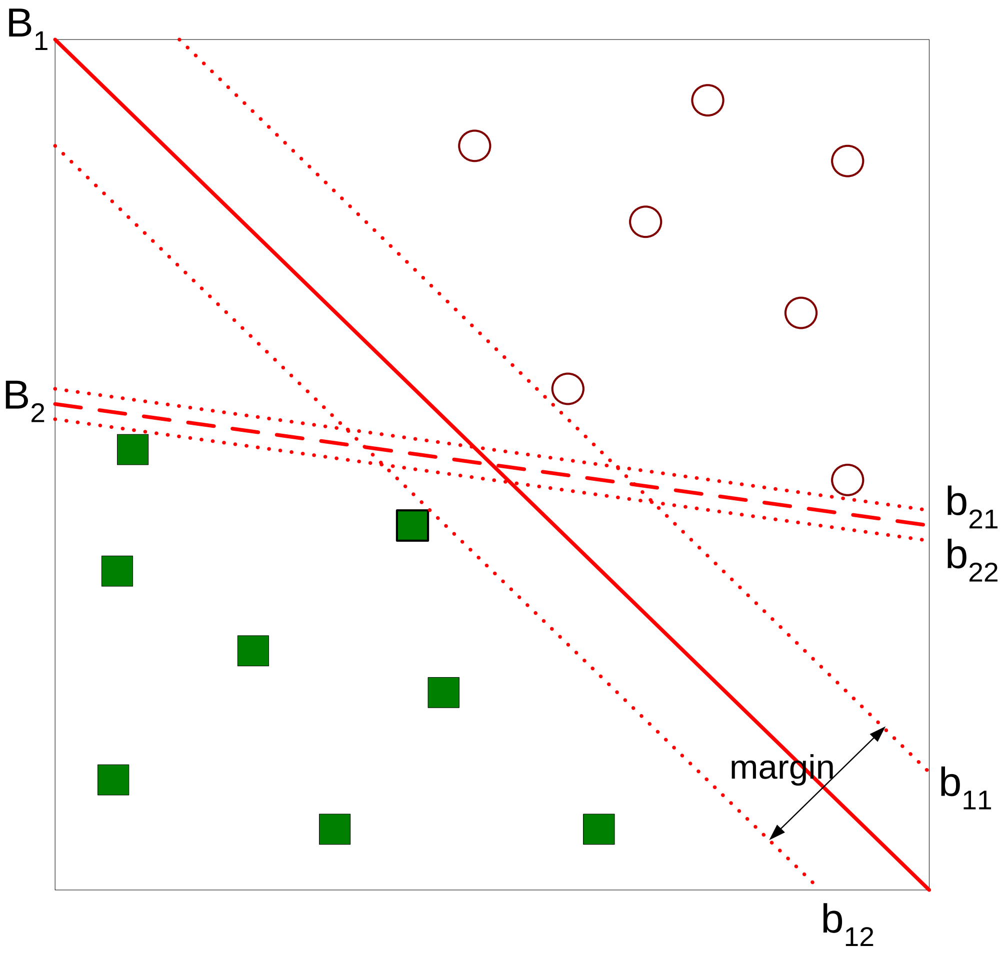
1

Support Vector Machines



- Which one is better? B1 or B2?
- How do you define better?

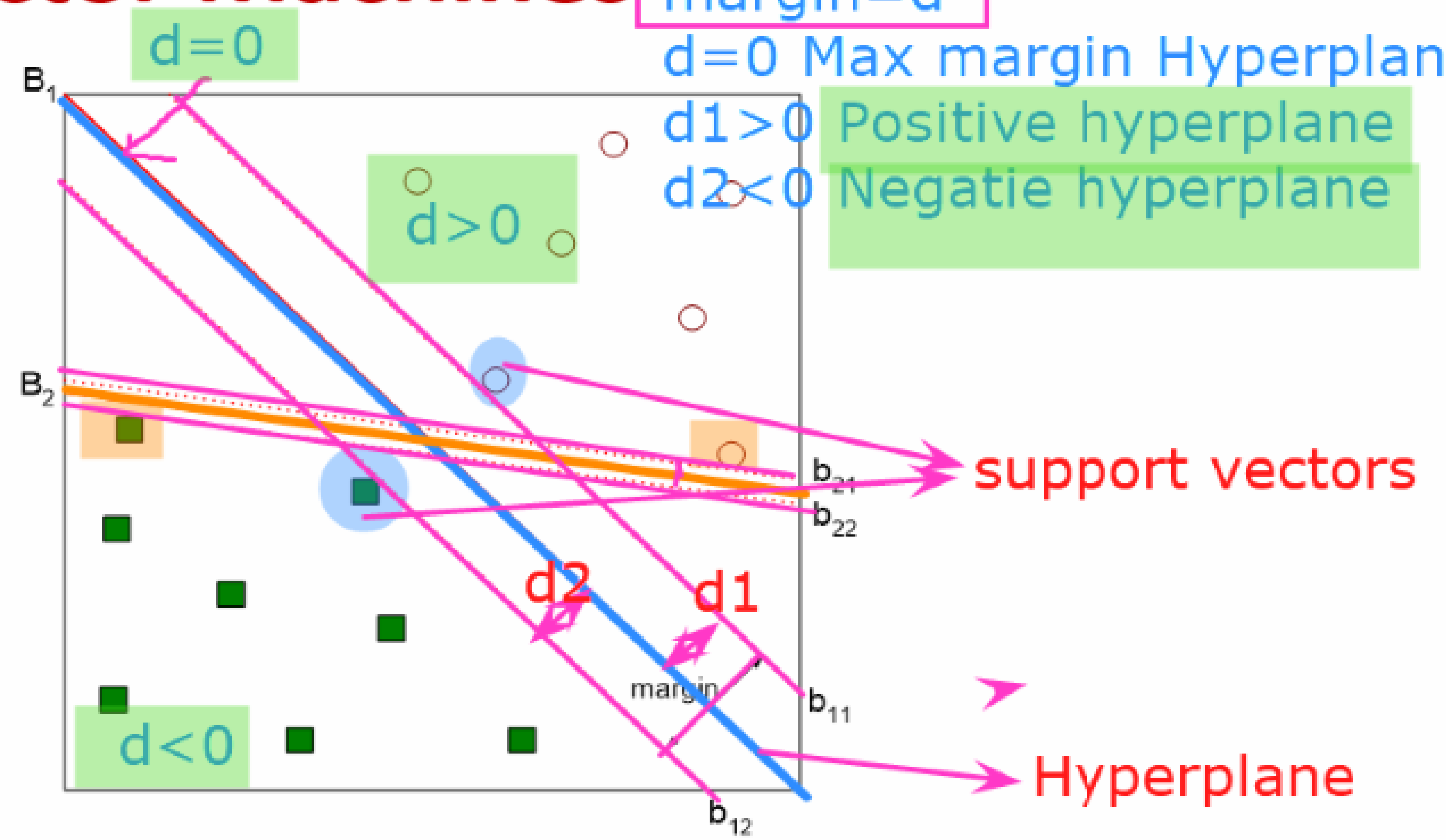
Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines

d=d1+d2
$\boxed{\text{margin=d}}$
d=0 Max margin Hyperplane
d1>0 Positive hyperplane
d2<0 Negatie hyperplane



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines

d=d1+d2

$\boxed{\text{margin=d}}$

d=0 Max margin Hyperplane

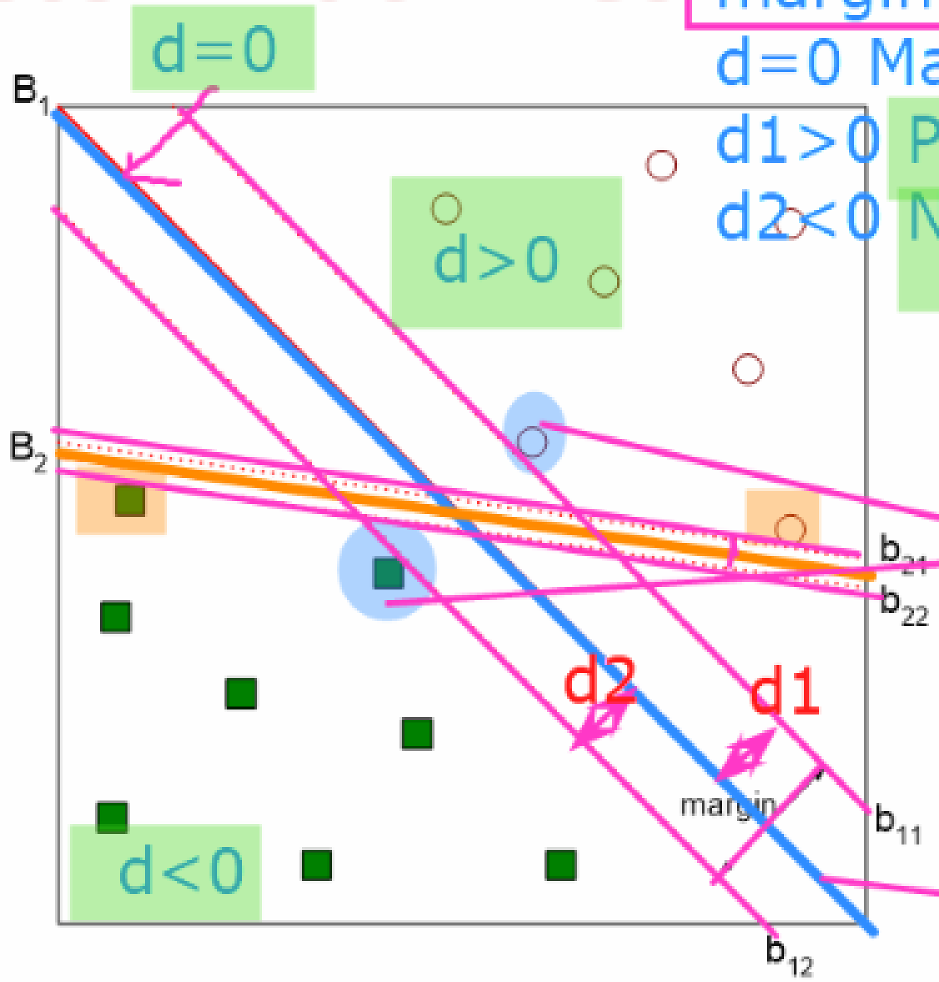
d1>0 Positive hyperplane

d2<0 Negatie hyperplane

wx+b=0 : d=0

wx+b=-1: d<0

wx+b=1 : d>0



support vectors

Hyperplane

- Find hyperplane maximizes the margin => B1 is better than B2

d=0

y=mx+c

y=wx+b   d=0

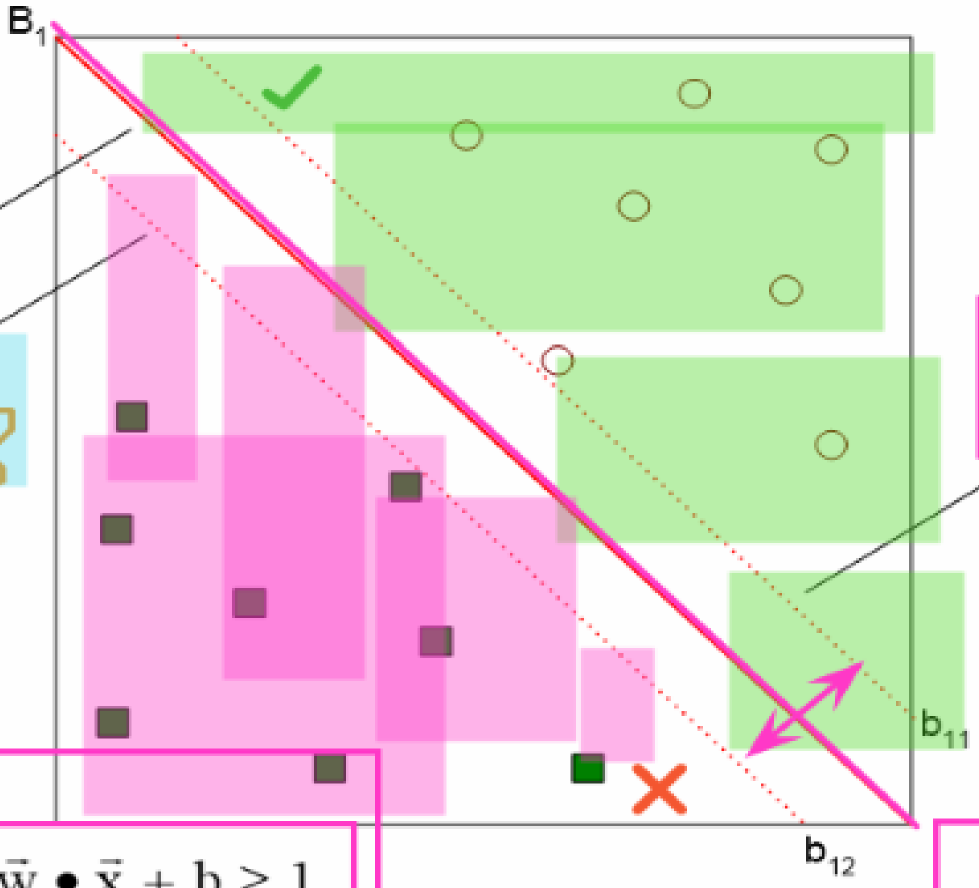
Support Vector Machines

slack variable ξ

Negative <0

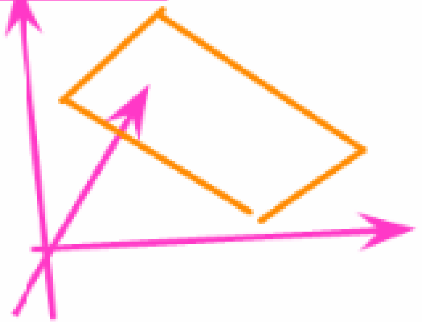
$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1 + \xi$$



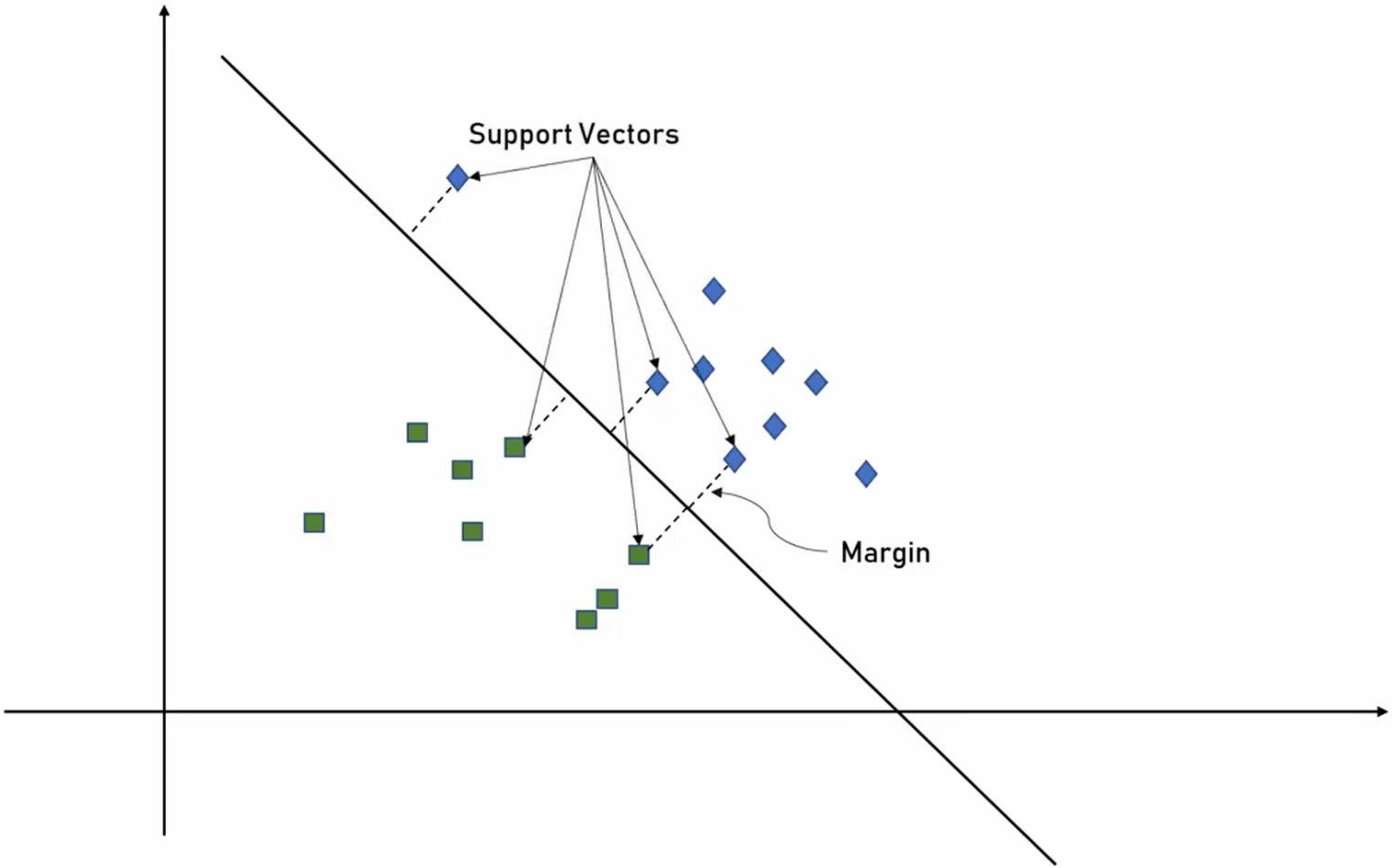
Positives >0

$$\vec{w} \bullet \vec{x} + b = +1 - \xi$$



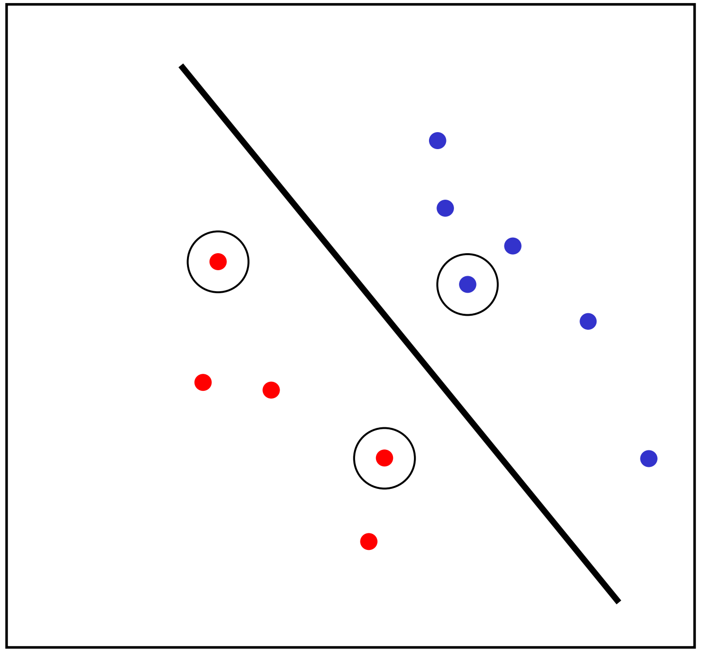
$$f(\vec{x}) = \begin{cases} \boxed{1} & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ \boxed{-1} & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|} + \lambda \sum \xi$$


Support Vectors
Margin

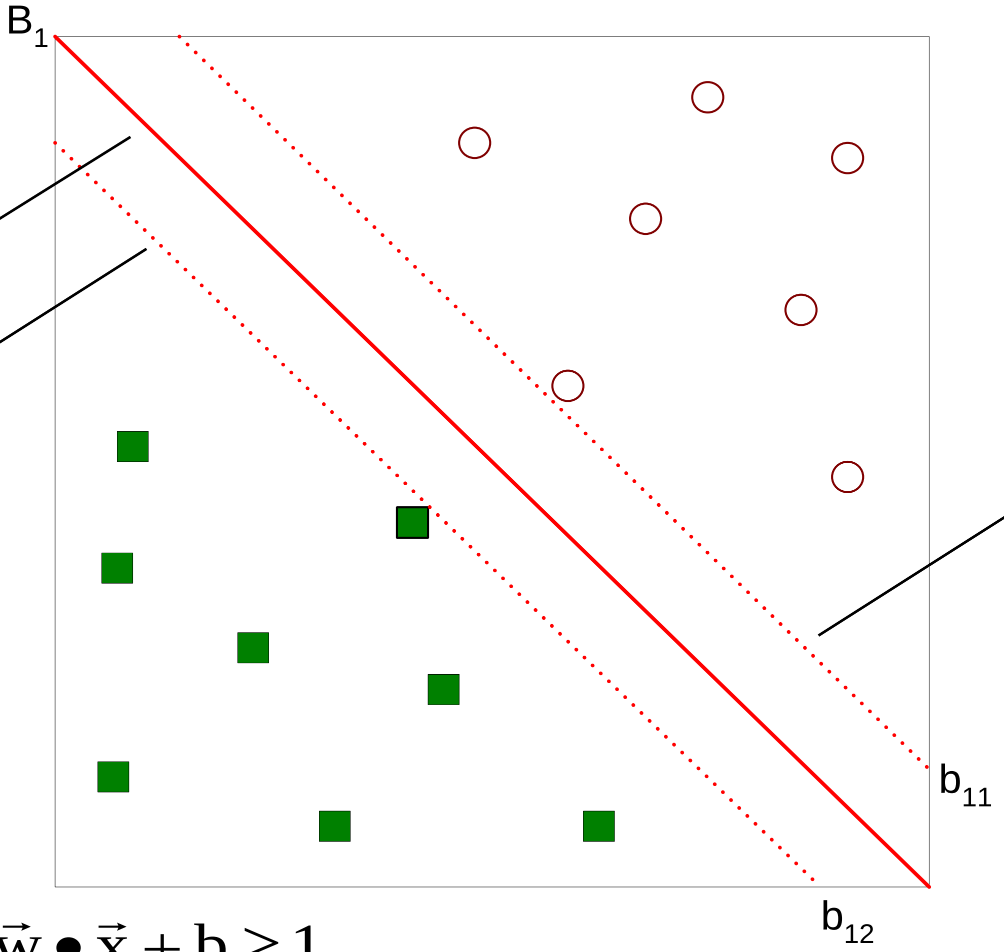
Support Vector Machines

- The line that maximizes the minimum margin is a good bet.
  - The model class of “hyper-planes with a margin of m” has a low VC dimension if m is big.
- This maximum-margin separator is determined by a subset of the datapoints.
  - Datapoints in this subset are called “support vectors”.
  - It will be useful computationally if only a small fraction of the datapoints are support vectors, because we use the support vectors to decide which side of the separator a test case is on.



The support vectors are indicated by the circles around them.

Support Vector Machines



$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$

$$\vec{w} \bullet \vec{x} + b = +1$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

Training a linear SVM

- To find the maximum margin separator, we have to solve the following optimization problem:

$$\mathbf{w}.\mathbf{x}^c + b > +1 \quad \textit{for positive cases}$$
$$\mathbf{w}.\mathbf{x}^c + b < -1 \quad \textit{for negative cases}$$
$$\textit{and} \quad ||\,\mathbf{w}\,||^2 \quad \textit{is as small as possible}$$

- This is tricky but it's a convex problem. There is only one optimum and we can find it without fiddling with learning rates or weight decay or early stopping.
  - Don't worry about the optimization problem. It has been solved. Its called quadratic programming.
  - It takes time proportional to N^2 which is really bad for very big datasets
    - so for big datasets we end up doing approximate optimization!

Testing a linear SVM

- The separator is defined as the set of points for which:

$$\mathbf{w}.\mathbf{x} + b = 0$$

$$so\ if\quad \mathbf{w}.\mathbf{x}^c + b > 0\quad say\ its\ a\ positive\ case$$

$$and\ if\quad \mathbf{w}.\mathbf{x}^c + b < 0\quad say\ its\ a\ negative\ case$$

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

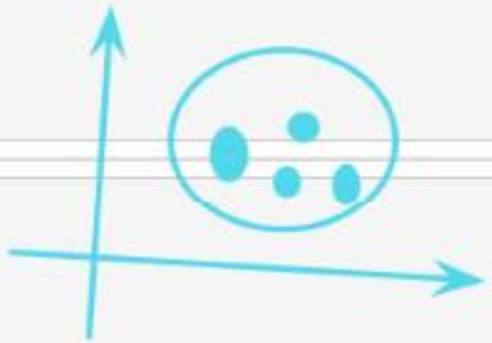
Run Code Commit

In [3]:
```python
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

In [4]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=1
```

In [5]:
```python
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

2d-->higher dimension



In [ ]:
```python
#Model Building
from sklearn.svm import SVC
SVC(kernel='linear')
```

Some commonly used kernels

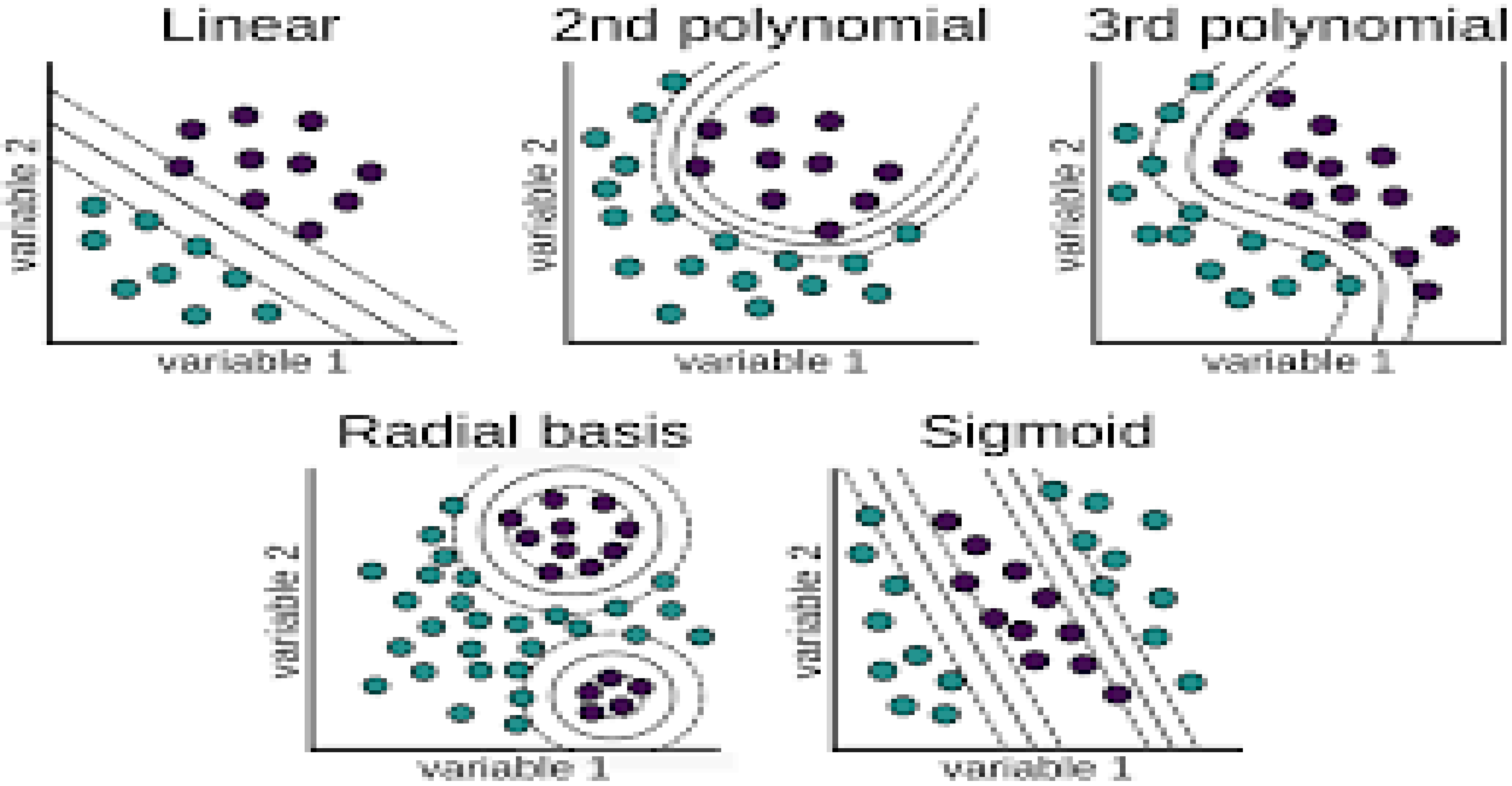
Polynomial: $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y} + 1)^p$

Gaussian radial basis function $K(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2 / 2\sigma^2}$

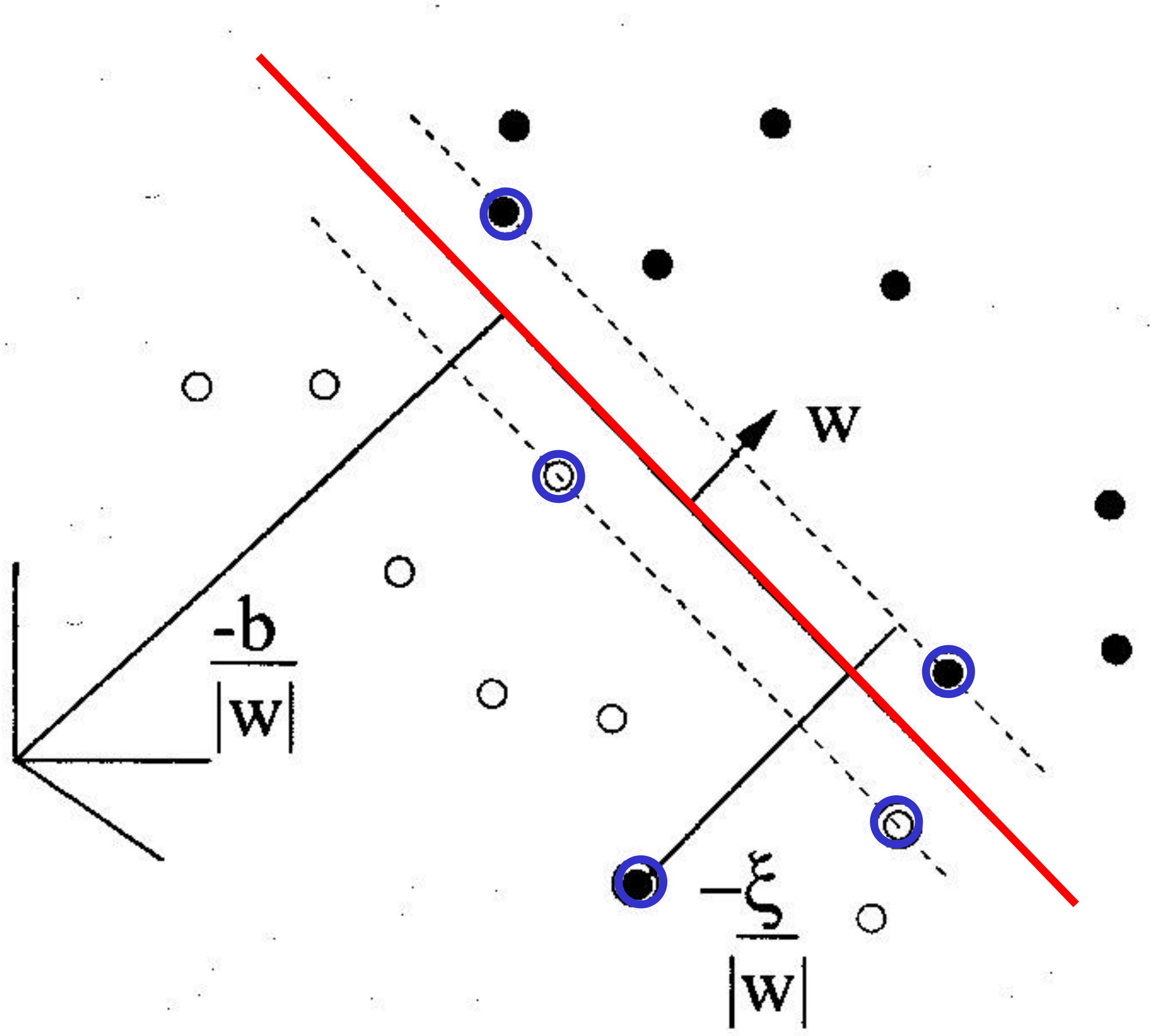
Parameters that the user must choose

Neural net: $K(\mathbf{x},\mathbf{y}) = \tanh(k\,\mathbf{x}.\mathbf{y} - \delta)$

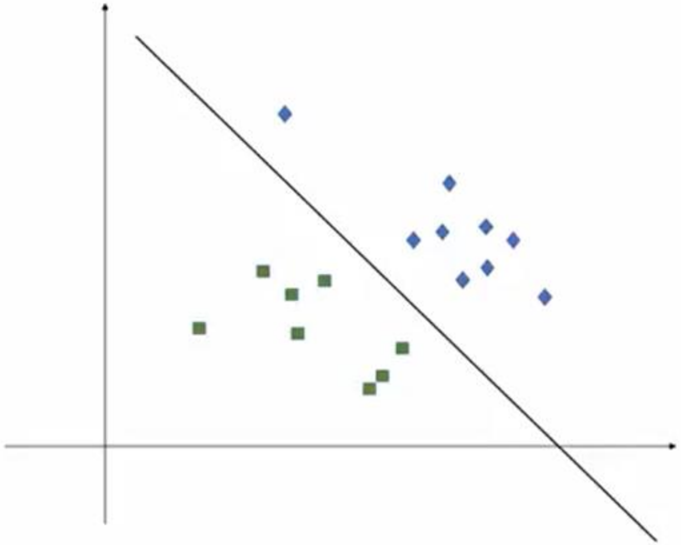
For the neural network kernel, there is one "hidden unit" per support vector, so the process of fitting the maximum margin hyperplane decides how many hidden units to use. Also, it may violate Mercer's condition.


Linear
variable 2
variable 1
2nd polynomial
variable 2
variable 1
3rd polynomial
variable 2
variable 1
Radial basis
variable 2
variable 1
Sigmoid
variable 2
variable 1

A picture of the best plane with a slack variable



2D



3D

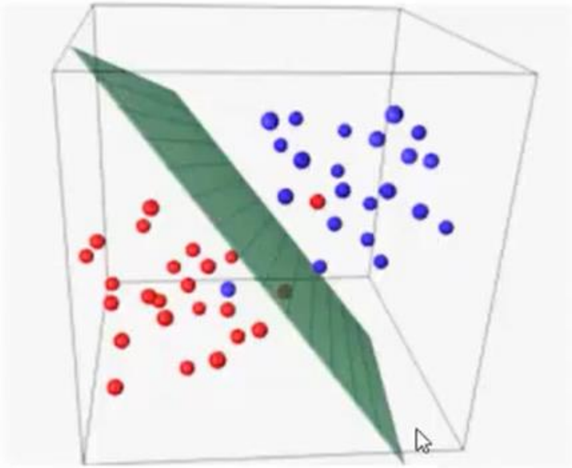
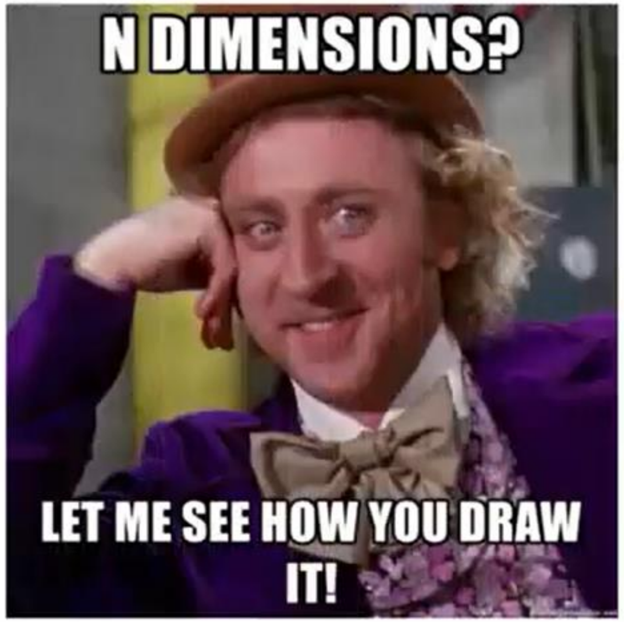
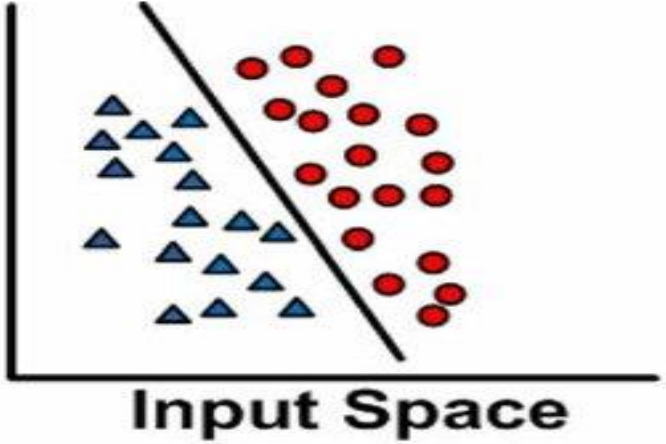


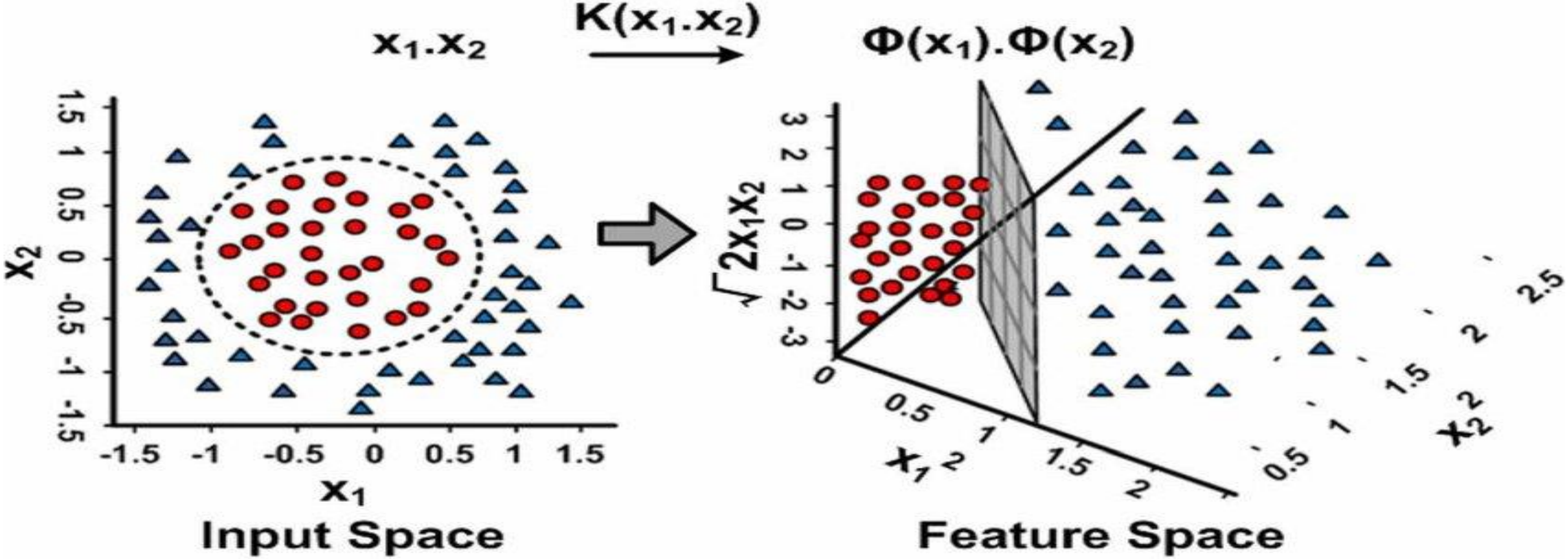
Image Credit: https://appliedmachinelearning.blog

nD




Input Space

(a)


$x_1.x_2$
$K(x_1.x_2)$
$\Phi(x_1).\Phi(x_2)$
1.5 1 0.5 0 -0.5 -1 -1.5
x_2
-1.5 -1 -0.5 0 0.5 1 1.5
x_1
Input Space
3 2 1 0 -1 -2 -3
$\sqrt{2}x_1x_2$
0 0.5 1 1.5 2
x_1^2
0.5 1 1.5 2 2.5
x_2^2
Feature Space

(b)