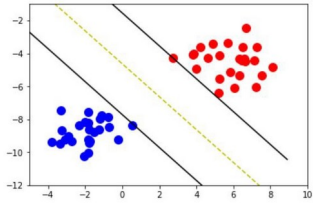
Support Vector Machine

Tushar B. Kute,
http://tusharkute.com



What is support vector?

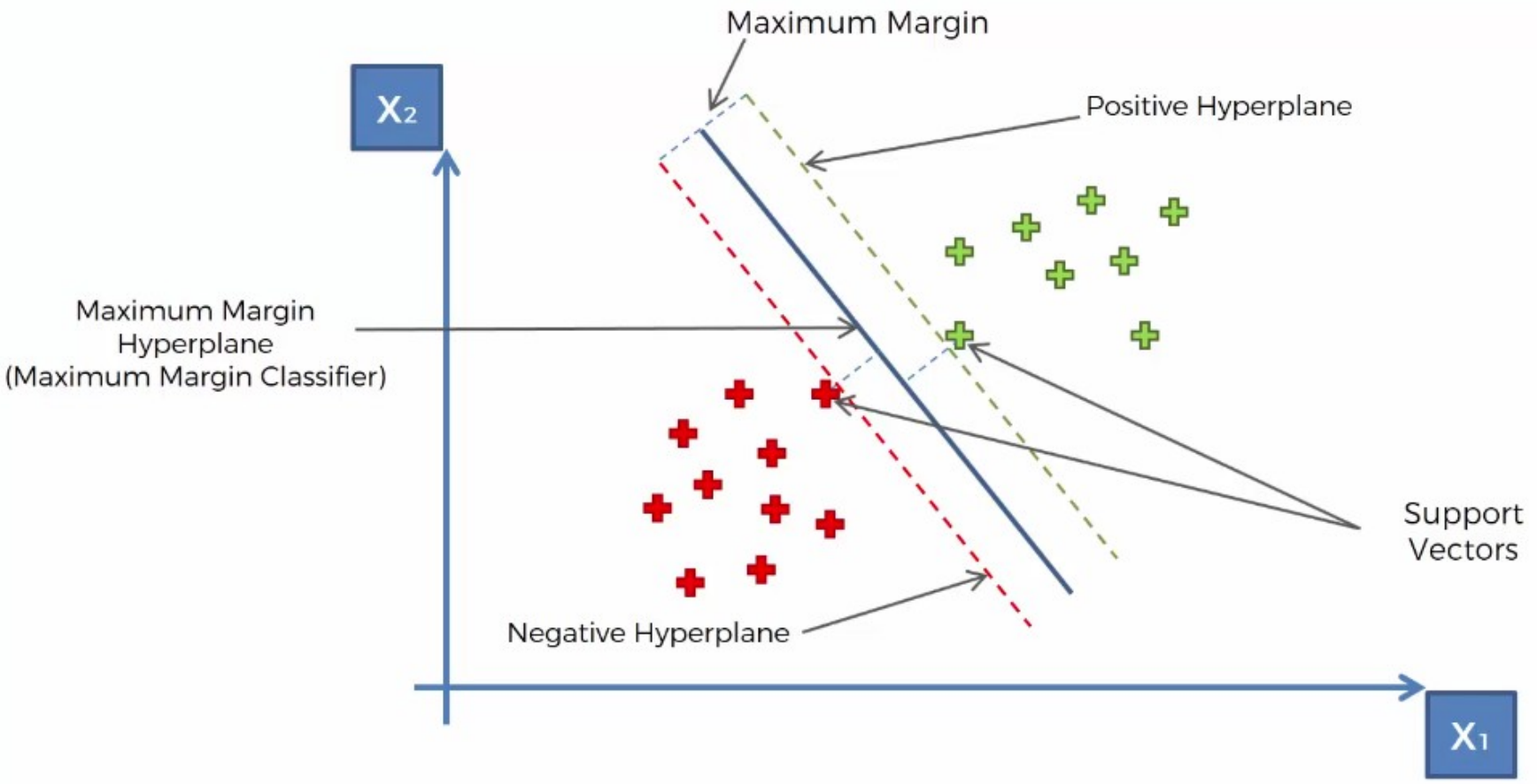
- “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.
- In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.
- Then, we perform classification by finding the hyperplane that differentiate the two classes very well.

Support Vector Machine

- Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems.
- It can easily handle multiple continuous and categorical variables.
- SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error.
- The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

miTu skillologies
tusharkute.com

Decision Vectors



Definitions

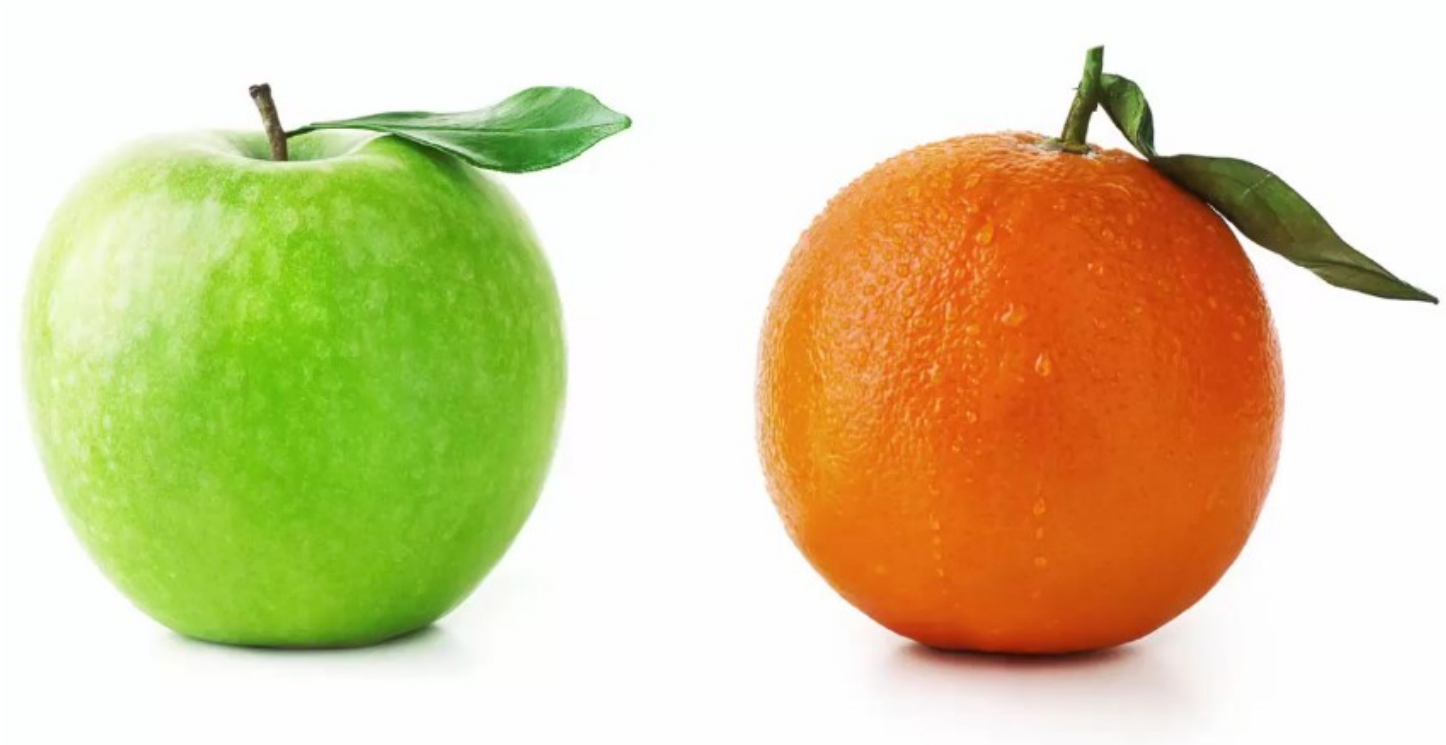
- Support Vectors
  - Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.
- Hyperplane
  - A hyperplane is a decision plane which separates between a set of objects having different class memberships.

miTu skillologies
tusharkute.com

Definitions

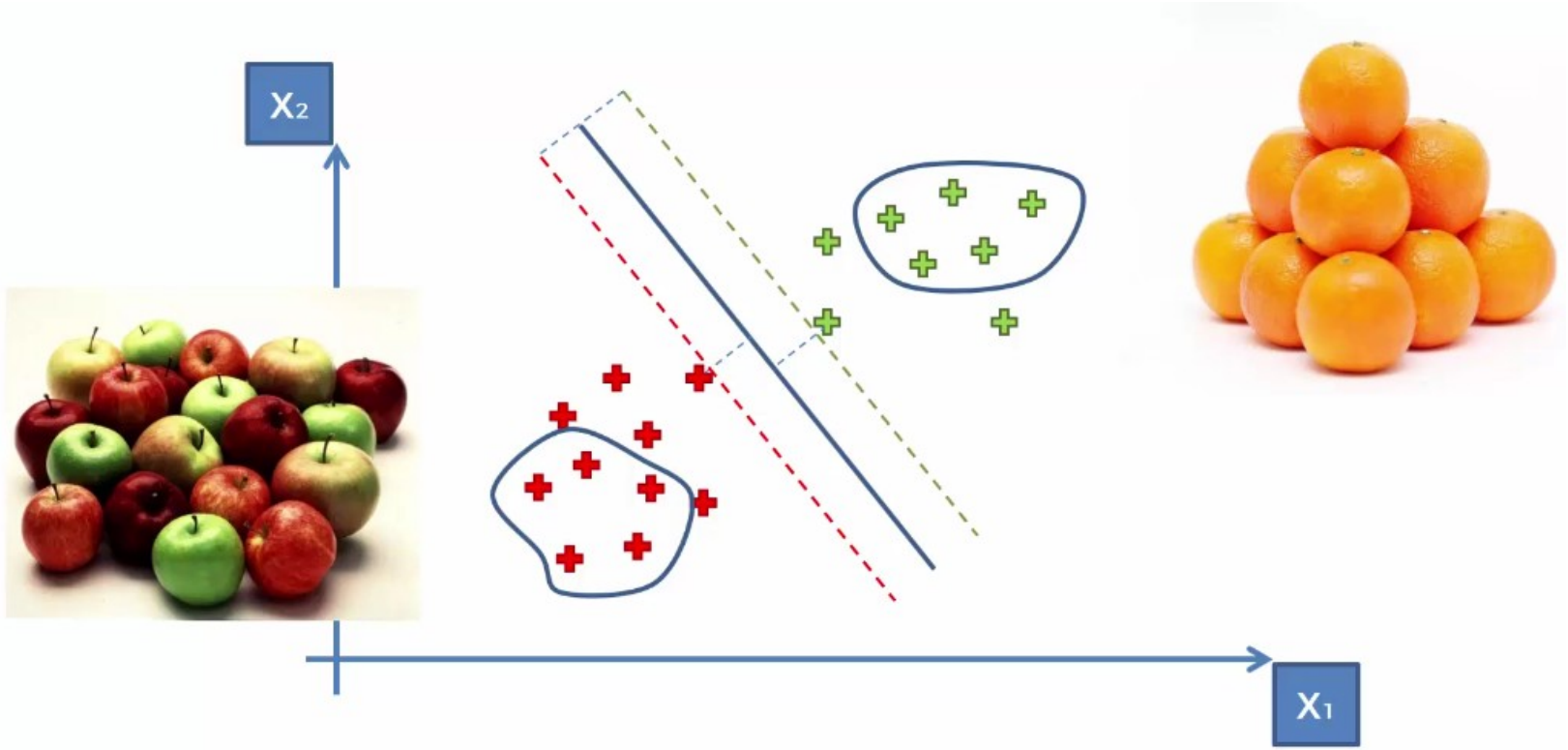
- Margin
  - A margin is a gap between the two lines on the closest class points.
  - This is calculated as the perpendicular distance from the line to support vectors or closest points.
  - If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

Why SVM is so special ?

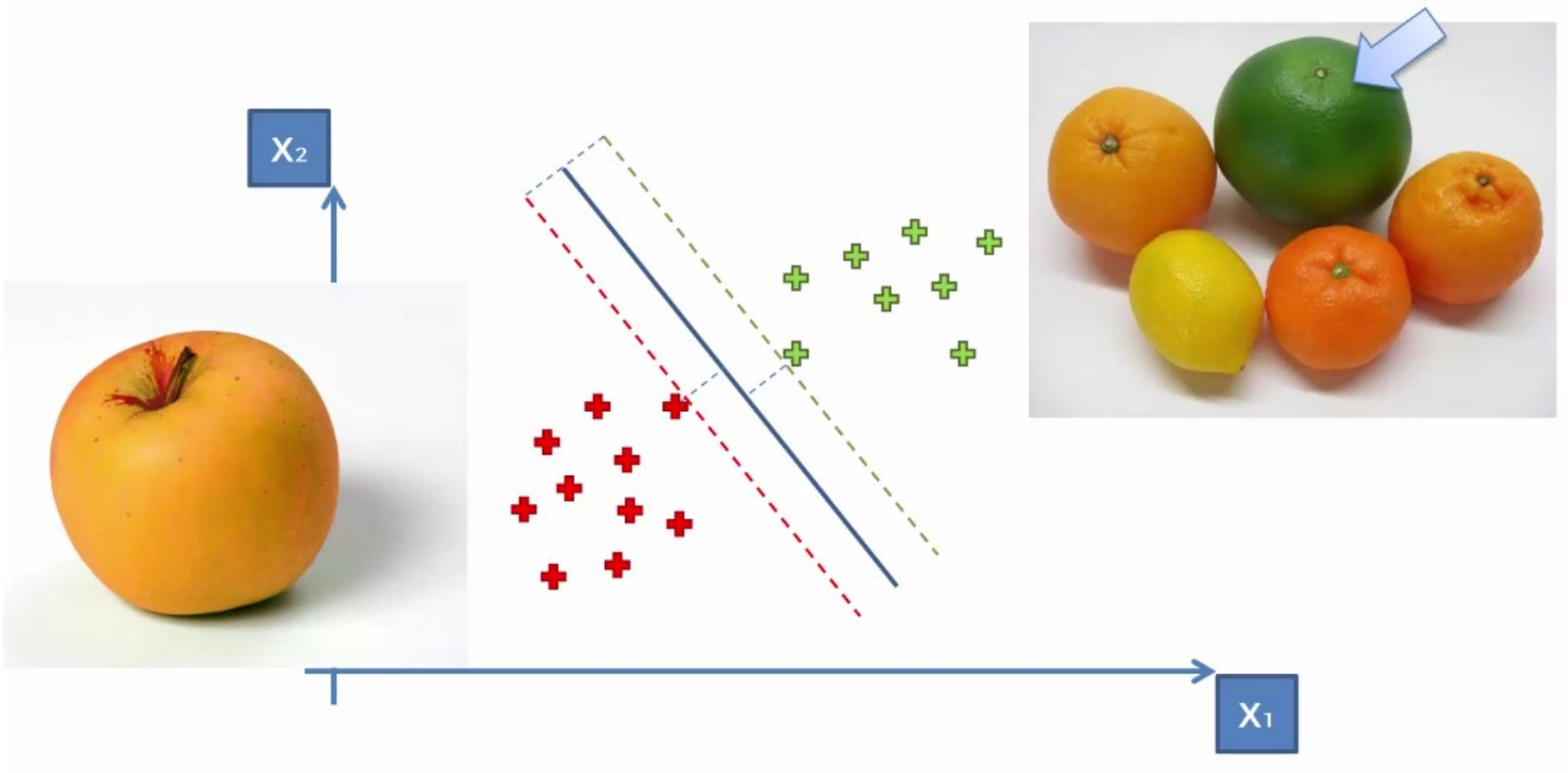


Example Reference: Super Data Science

How SVM works for this?



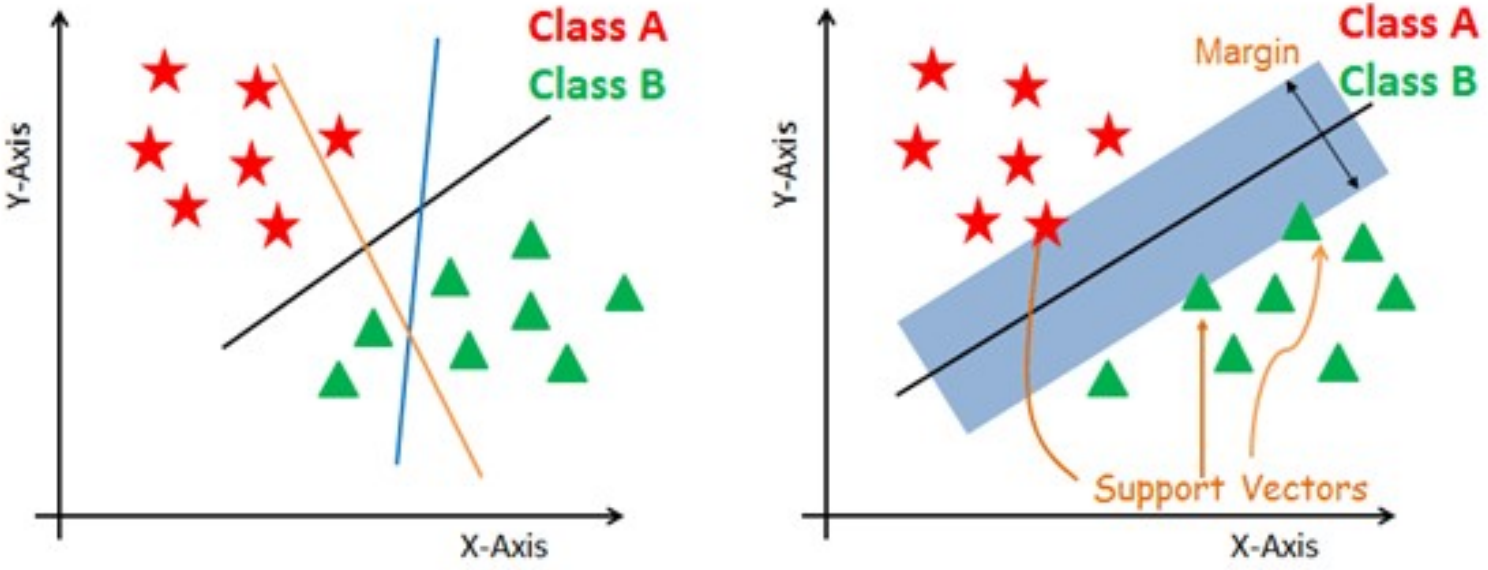
How classification will work ?



How SVM works ?

- The main objective is to segregate the given dataset in the best possible way.
- The distance between the either nearest points is known as the margin.
- The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:
  - Generate hyperplanes which segregates the classes in the best way.
  - Select the right hyperplane with the maximum segregation from the either nearest data points.

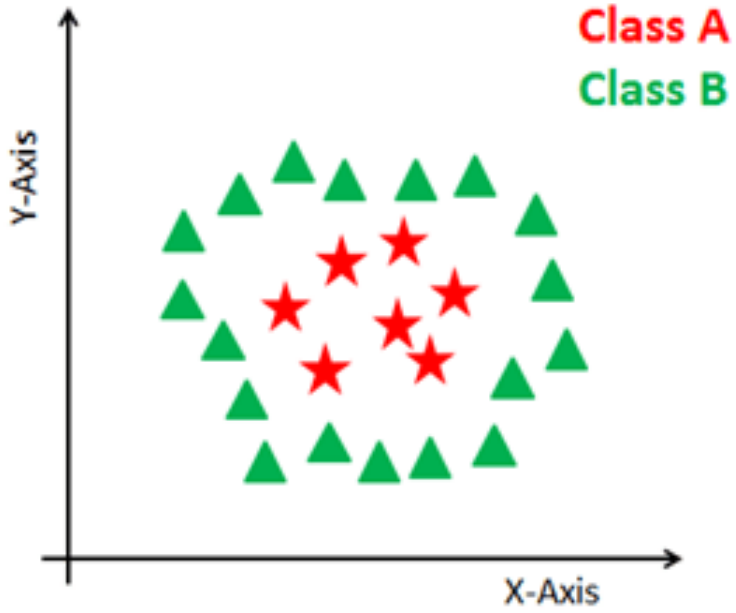
How SVM works ?

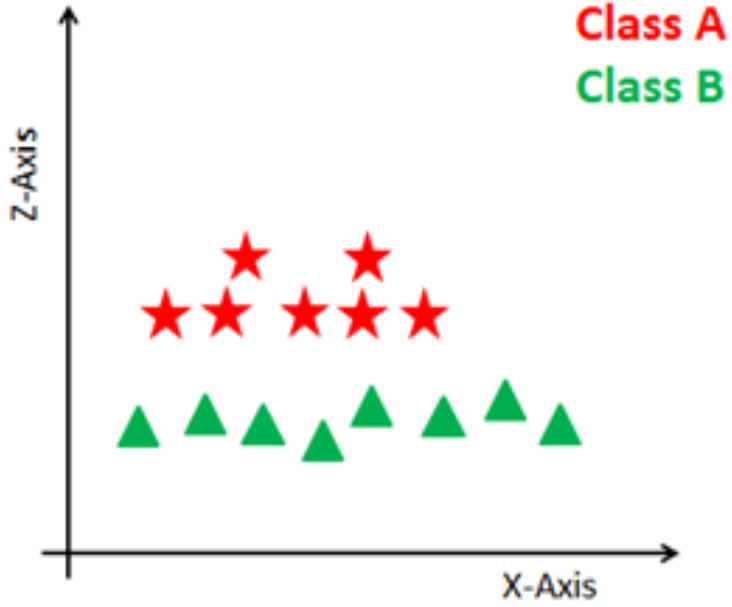


Non-linear and inseparable planes

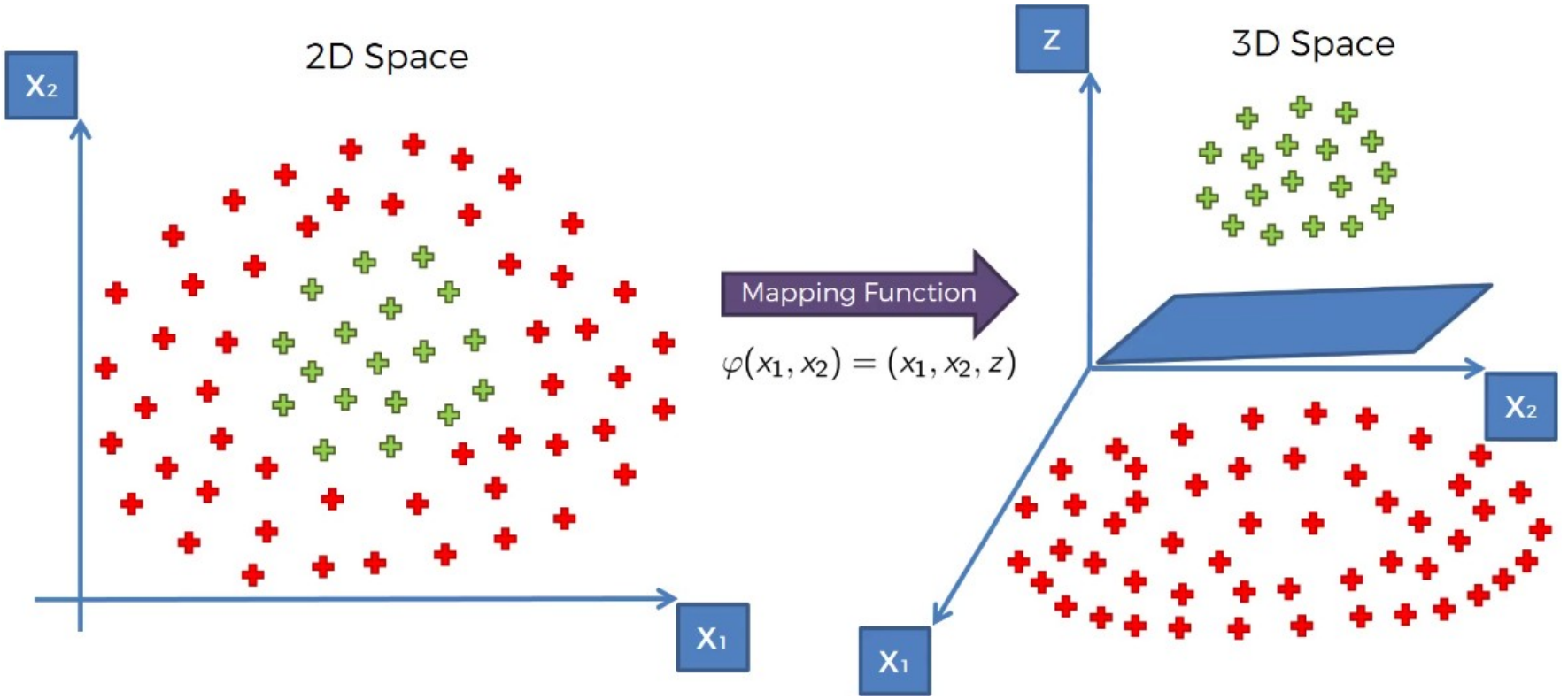
- Some problems can’t be solved using linear hyperplane.
- In such situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right.
- The data points are plotted on the x-axis and z-axis (Z is the squared sum of both x and y: z=x^2=y^2).
- Now you can easily segregate these points using linear separation.

Non-linear and inseparable planes

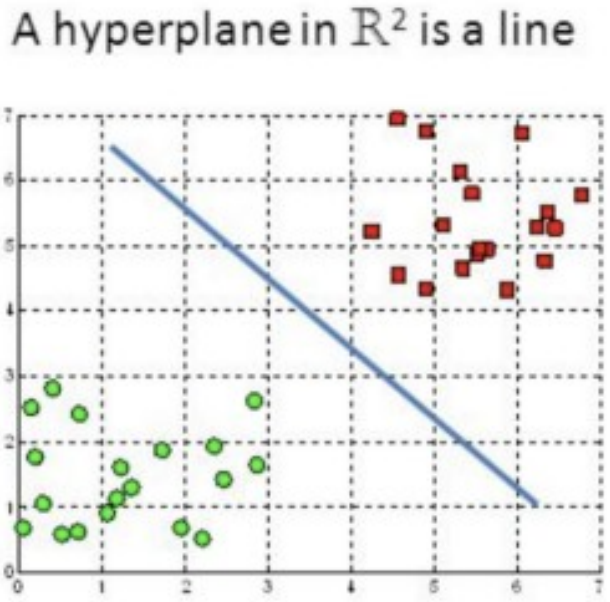


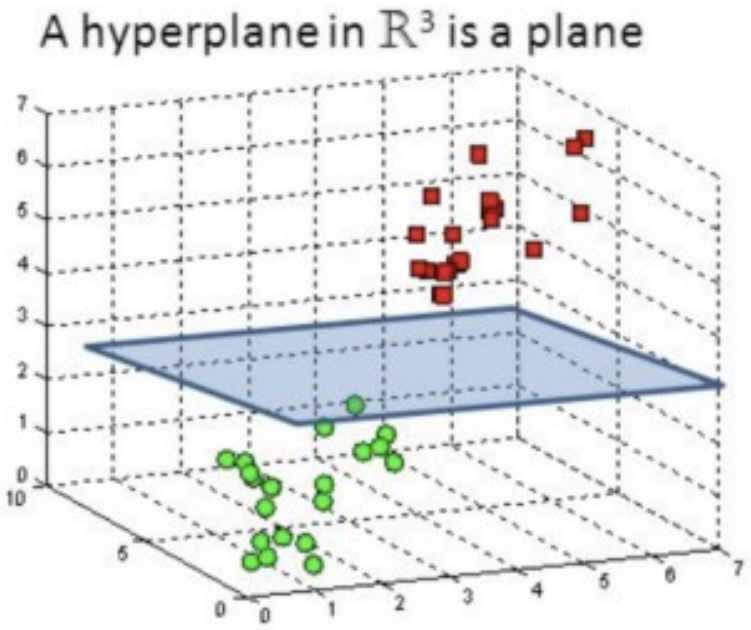


High Dimensional Space Mapping



High Dimensional Space Mapping





SVM Kernels

- The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form.
- SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space.
- In other words, you can say that it converts non-separable problem to separable problems by adding more dimension to it.
- It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

Kernel Types

- Linear Kernel
- Polynomial Kernel
- Radial Basis Function Kernel
- Sigmoid Kernel

Radial Basis Function Kernel

- The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

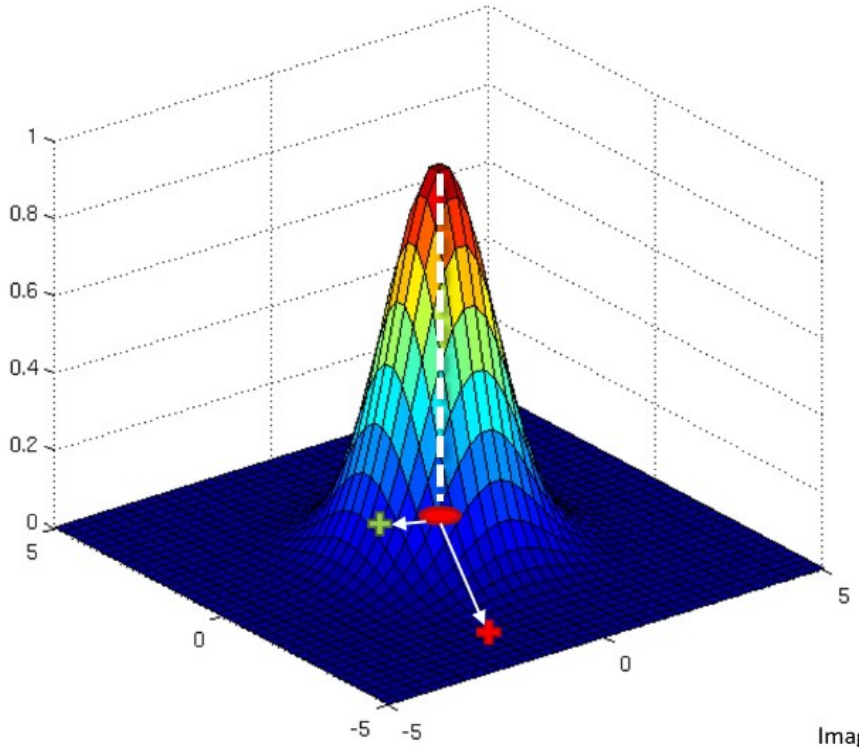
$$K(x,xi) = \exp(-gamma * sum((x - xi^2))$$

- Here gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. Gamma=0.1 is considered to be a good default value.
- The value of gamma needs to be manually specified in the learning algorithm.

Radial Basis Function Kernel

$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$

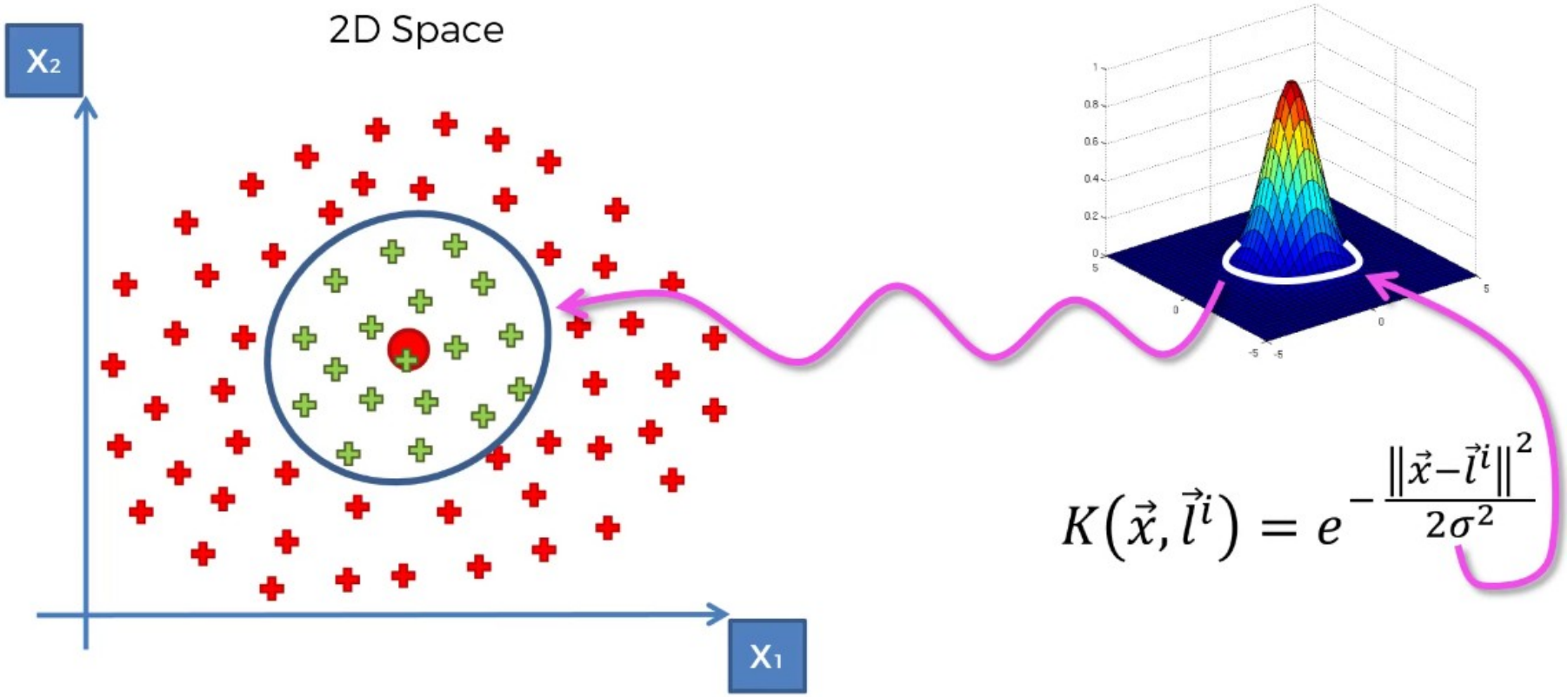
Radial Basis Function Kernel



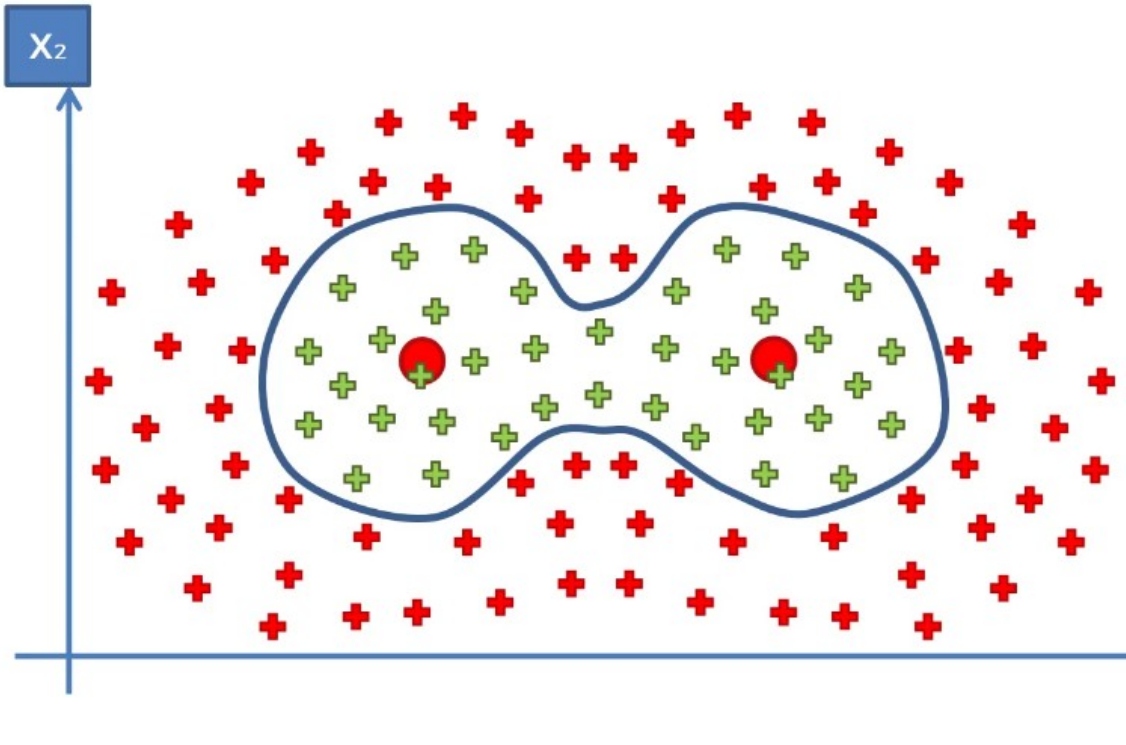
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

Radial Basis Function Kernel



Radial Basis Function Kernel



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

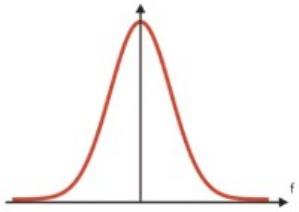
Green when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$

Red when:

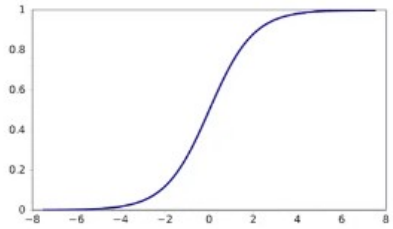
$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) = 0$$

Types of Kernels



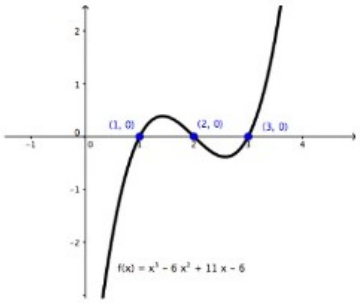
Gaussian RBF Kernel

$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$



Sigmoid Kernel

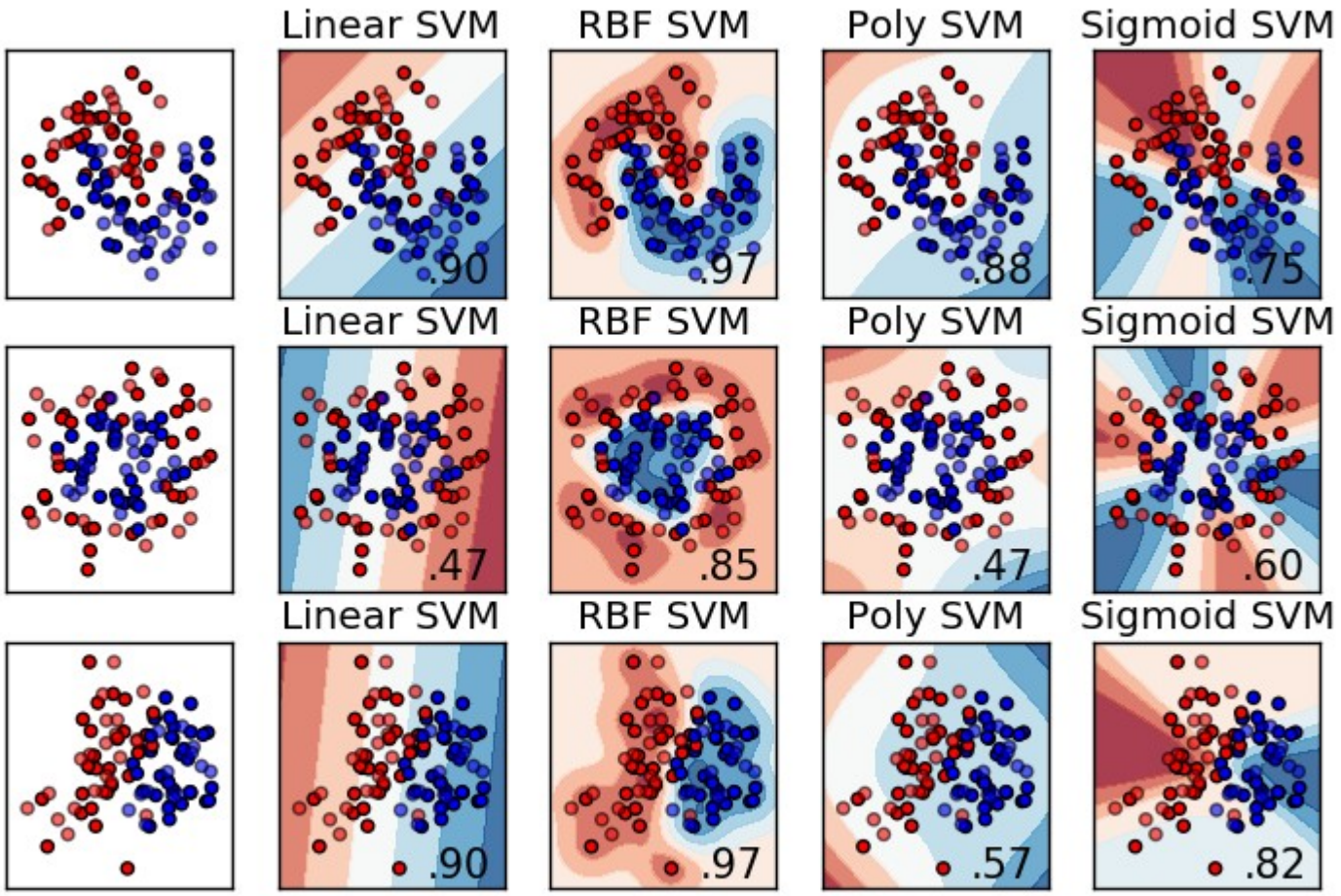
$$K(X, Y) = \tanh(\gamma \cdot X^{T}Y + r)$$



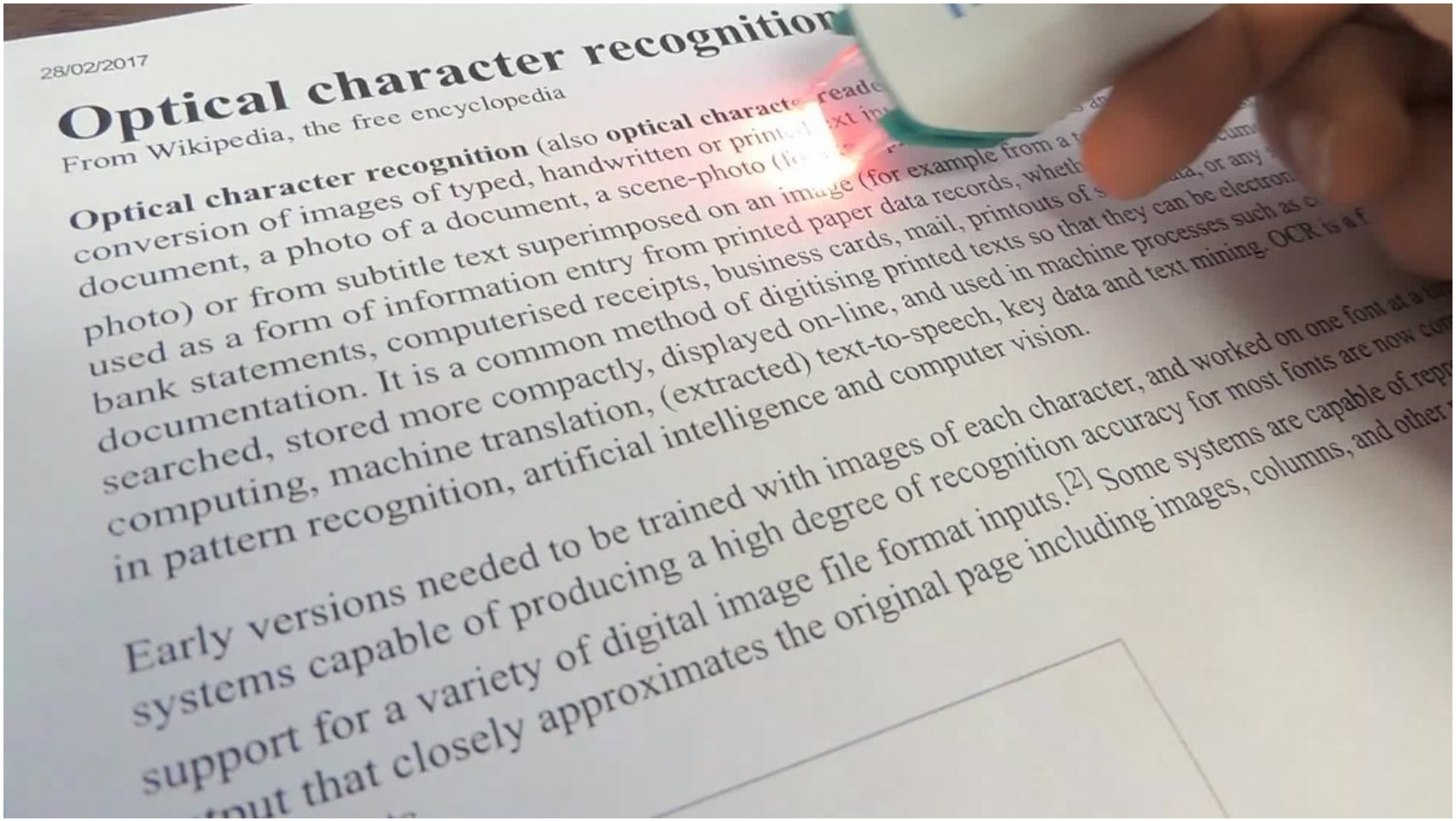
Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^{T}Y + r)^{d}, \gamma > 0$$

Types of Kernels



Example: OCR


28/02/2017
Optical character recognition
From Wikipedia, the free encyclopedia
Optical character recognition (also optical
conversion of images of typed, handwritten or
document, a photo of a document, a scene-photo
photo) or from subtitle text superimposed on an
used as a form of information entry from printed paper data records,
bank statements, computerised receipts, business cards, mail, printouts of
documentation. It is a common method of digitising printed texts so that they can be electron
searched, stored more compactly, displayed on-line, and used in machine processes such as
computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a f
in pattern recognition, artificial intelligence and computer vision.
Early versions needed to be trained with images of each character, and worked on one font at a
systems capable of producing a high degree of recognition accuracy for most fonts are now
support for a variety of digital image file format inputs.[2] Some systems are capable of
that closely approximates the original page including images, columns, and other

Example: Image Processing

- Image processing is a difficult task for many types of machine learning algorithms.
- The relationships linking patterns of pixels to higher concepts are extremely complex and hard to define.
- For instance, it's easy for a human being to recognize a face, a cat, or the letter "A", but defining these patterns in strict rules is difficult.
- Furthermore, image data is often noisy. There can be many slight variations in how the image was captured, depending on the lighting, orientation, and positioning of the subject.

Example: Data Collection

- When OCR software first processes a document, it divides the paper into a matrix such that each cell in the grid contains a single glyph, which is just a term referring to a letter, symbol, or number.
- Next, for each cell, the software will attempt to match the glyph to a set of all characters it recognizes.
- Finally, the individual characters would be combined back together into words, which optionally could be spell-checked against a dictionary in the document's language.

The Dataset

- We'll use a dataset donated to the UCI Machine Learning Data Repository (http://archive.ics.uci.edu/ml) by W. Frey and D. J. Slate.
- The dataset contains 20,000 examples of 26 English alphabet capital letters as printed using 20 different randomly reshaped and distorted black and white fonts.
- The following figure, published by Frey and Slate, provides an example of some of the printed glyphs.
- Distorted in this way, the letters are challenging for a computer to identify, yet are easily recognized by a human being:

Actual Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	"letter"	"xbox"	"ybox"	"width"	"height"	"onpix"	"xbar"	"ybar"	"x2bar"	"y2bar"	"xybar"	"x2ybar"	"xy2bar"	"xedge"	"xedgey"	"yedge"	"yedgex"
2	"T"	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8
3	"I"	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10
4	"D"	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9
5	"N"	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8
6	"G"	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10
7	"S"	4	11	5	8	3	8	8	6	9	5	6	6	0	8	9	7
8	"B"	4	2	5	4	4	8	7	6	6	7	6	6	2	8	7	10
9	"A"	1	1	3	2	1	8	2	2	2	8	2	8	1	6	2	7
10	"J"	2	2	4	4	2	10	6	2	6	12	4	8	1	6	1	7
11	"M"	11	15	13	9	7	13	2	6	2	12	1	9	8	1	1	8
12	"X"	3	9	5	7	4	8	7	3	8	5	6	8	2	8	6	7
13	"O"	6	13	4	7	4	6	7	6	3	10	7	9	5	9	5	8
14	"G"	4	9	6	7	6	7	8	6	2	6	5	11	4	8	7	8
15	"M"	6	9	8	6	9	7	8	6	5	7	5	8	8	9	8	6
16	"R"	5	9	5	7	6	6	11	7	3	7	3	9	2	7	5	11
17	"F"	6	9	5	4	3	10	6	3	5	10	5	7	3	9	6	9
18	"O"	3	4	4	3	2	8	7	7	5	7	6	8	2	8	3	8
19	"C"	7	10	5	5	2	6	8	6	8	11	7	11	2	8	5	9
20	"T"	6	11	6	8	5	6	11	5	6	11	9	4	3	12	2	4
21	"J"	2	2	3	3	1	10	6	3	6	12	4	9	0	7	1	7
22	"J"	1	3	2	2	1	8	8	2	5	14	5	8	0	7	0	7
23	"H"	4	5	5	4	4	7	7	6	6	7	6	8	3	8	3	8
24	"S"	3	2	3	3	2	8	8	7	5	7	5	7	2	8	9	8
25	"O"	6	11	7	8	5	7	6	9	6	7	5	9	4	8	5	5
26	"J"	3	6	4	4	2	6	6	4	4	14	8	12	1	6	1	6

Working with dataset

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

# Read the dataset
letter = pd.read_csv("letterdata.csv")
print letter.shape
# Print top entries
print letter.head()

# Input parameters
X = letter.drop('letter', axis=1)
# Output parameters
y = letter['letter']
```

Training and Prediction

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split
(X,y,test_size = 0.20)

# Support Vector Classification
from sklearn.svm import SVC
# SVM with Linear Kernel
svclassifier = SVC(kernel='linear')

# Train the classifier
svclassifier.fit(X_train, y_train)

# Prediction
y_pred = svclassifier.predict(X_test)
```

Characterization

```
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score

# Print confusion matrix
print confusion_matrix(y_test,y_pred)

# Print classification report
print classification_report(y_test,y_pred)

# Accuracy score of the algorithm
print accuracy_score(y_test,y_pred) * 100
```

Output:

	precision	recall	f1-score	support
A	0.88	0.91	0.90	152
B	0.80	0.88	0.83	138
C	0.90	0.87	0.88	141
D	0.80	0.90	0.85	162
E	0.78	0.81	0.80	147
F	0.87	0.90	0.89	155
G	0.67	0.78	0.72	149
H	0.76	0.69	0.72	156
I	0.87	0.92	0.89	125
J	0.86	0.87	0.86	144
K	0.82	0.77	0.80	164
L	0.89	0.92	0.91	153
M	0.96	0.95	0.95	160
N	0.91	0.87	0.89	182
O	0.84	0.78	0.81	138
P	0.94	0.82	0.88	171
Q	0.82	0.81	0.82	164
R	0.76	0.79	0.77	140
S	0.68	0.69	0.68	140
T	0.92	0.90	0.91	183
U	0.90	0.96	0.93	158
V	0.96	0.94	0.95	150
W	0.95	0.92	0.93	170
X	0.90	0.90	0.90	167
Y	0.91	0.92	0.92	145
Z	0.89	0.75	0.81	146

```
85.6
```

References:

- https://mitu.co.in
- https://superdatascience.com
- https://stackabuse.com/
- https://jakevdp.github.io
- https://towardsdatascience.com
- https://www.datacamp.com
- http://scikit-learn.org/

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com