Advance Machine Learning

01CO1301
4 Credits

Marwadi University

Department of Computer Engineering

Ravikumar R Natarajan

Course Outcomes

- At the end of the course, students will be able to:
- To understand key concepts, tools and approaches for pattern recognition on complex data sets.
- To learn Kernel methods for handling high dimensional and non-linear patterns.
- To implement state-of-the-art algorithms such as Support Vector Machines and Bayesian networks.
- To Solve real-world machine learning tasks: from data to inference.
- To apply theoretical concepts and the motivations behind different learning frameworks.

Marwadi University

Department of Computer Engineering

Kernel Methods

Unit #2

Content

- Kernel Methods for Non-linear Data
- Support Vector Machines
- Kernel Ridge Regression
- Structure Kernels
- Kernel PCA
- Latent Semantic Analysis

What are Kernel Methods?

- Kernels or kernel methods (also called Kernel functions) are sets of different types of algorithms that are being used for pattern analysis.
- **They are used to solve a non-linear problem by using a linear classifier.**
- The concept is to use a mapping function to project nonlinear combinations of the original features onto a higher-dimensional space, where the data becomes linearly separable.

Need of Kernel?

- Let’s say we have a 2-dimensional dataset with two classes of observations, as illustrated in fig-1 below, we need to design a function to separate them. As it can be seen below that the given data in a 2-dimensional space is not linearly separable.

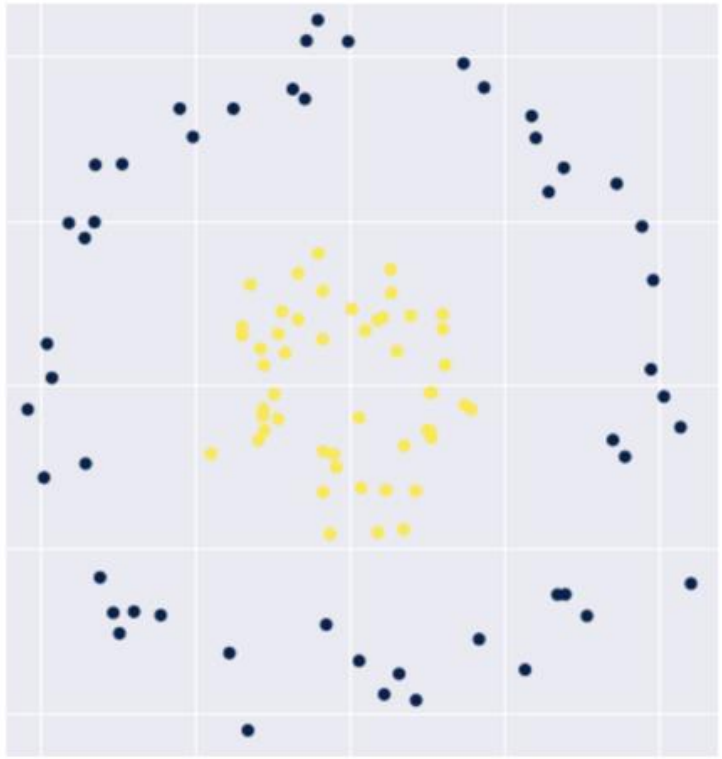


Fig-1

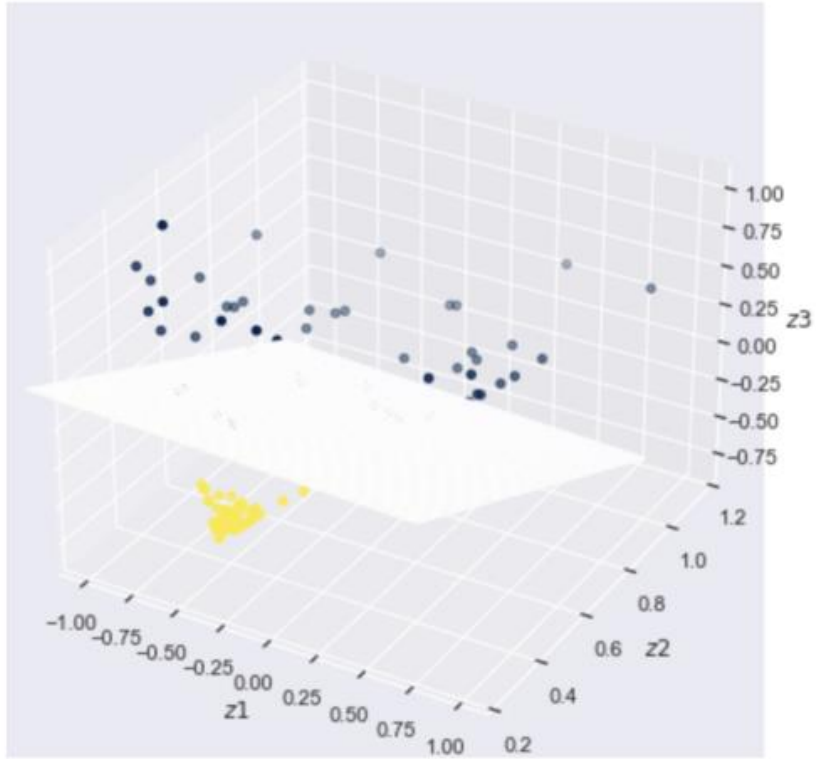


Fig-2

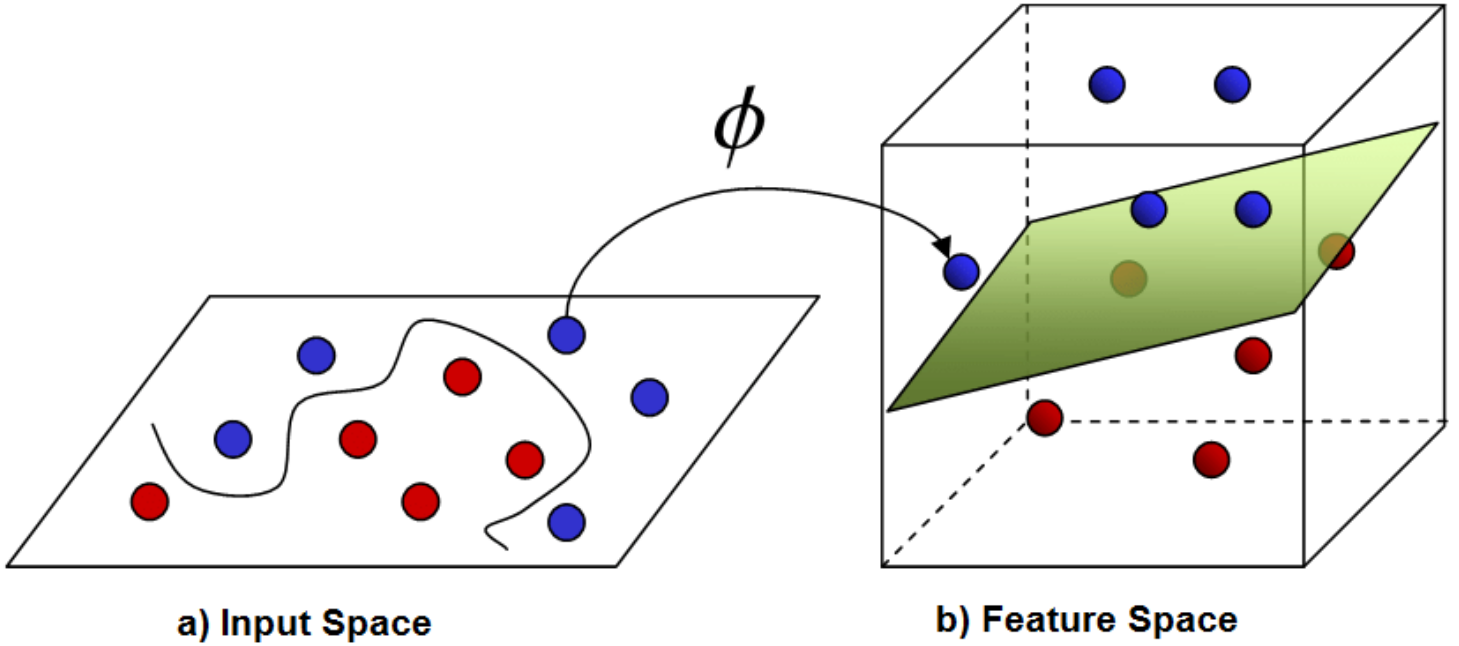
Need of Kernel?

- **What if we could translate this data into a higher-dimensional (3D) space where a linear classifier could separate the data?**

- This is when the kernel technique comes in handy. The goal is to use a function that projects / transform the data in such a way that it can be separated linearly.
- By using the maximum margin in the SVM method, data becomes linearly separable.

Kernel Trick

The kernel function takes in lower-dimensional inputs and outputs the dot product of converted vectors in higher-dimensional space.



Kernel Methods for Non-linear Data

In order to get a mathematical understanding of kernel, let us understand the Lili Jiang's equation of kernel which is:

K(x, y)=<f(x), f(y)> where,

- K is the kernel function,
- X and Y are the dimensional inputs,
- f is the map from n-dimensional to m-dimensional space and,
- < x, y > is the dot product.

Kernel Methods for Non-linear Data

Let us say that we have two points, x= (2, 3, 4) and y= (3, 4, 5)

As we have seen, K(x, y) = < f(x), f(y) >.
Let us first calculate < f(x), f(y) >
f(x)=(x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)
f(y)=(y1y1, y1y2, y1y3, y2y1, y2y2, y2y3, y3y1, y3y2, y3y3)
so,
f(2, 3, 4)=(4, 6, 8, 6, 9, 12, 8, 12, 16)and

f(3 ,4, 5)=(9, 12, 15, 12, 16, 20, 15, 20, 25)
so the dot product,
f (x). f (y) = f(2,3,4) . f(3,4,5)=
(36 + 72 + 120 + 72 +144 + 240 + 120 + 240 + 400)=1444
And,
K(x, y) = (2*3 + 3*4 + 4*5) ^2=(6 + 12 + 20)^2=38*38=1444.

Types of Kernel Functions

1. Liner Kernel

Let us say that we have two vectors with name x1 and Y1, then the linear kernel is defined by the dot product of these two vectors:

$K(x1, x2) = x1 . x2$

2. Polynomial Kernel

A polynomial kernel is defined by the following equation:

$K(x1, x2) = (x1 . x2 + 1)d$

Where,

d is the degree of the polynomial and x1 and x2 are vectors.

Types of Kernel Functions

3. Gaussian Kernel

This kernel is an example of a radial basis function kernel. Below is the equation for this:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

The given sigma plays a very important role in the performance of the Gaussian kernel and should neither be overestimated and nor be underestimated, it should be carefully tuned according to the problem.

Types of Kernel Functions

4. Exponential Kernel

This is in close relation with the previous kernel i.e. the Gaussian kernel with the only difference is – the square of the norm is removed.

The function of the exponential function is:

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{2\sigma^2}\right)$$

This is also a radial basis kernel function.

Types of Kernel Functions

5. Laplacian Kernel

This type of kernel is less prone for changes and is totally equal to previously discussed exponential function kernel, the equation of Laplacian kernel is given as:

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{\sigma}\right)$$

Types of Kernel Functions

6. Hyperbolic or the Sigmoid Kernel

This kernel is used in neural network areas of machine learning. The activation function for the sigmoid kernel is the bipolar sigmoid function. The equation for the hyperbolic kernel function is:

$$k(x, y) = \tanh(\alpha x^T y + c)$$

This kernel is very much used and popular among support vector machines.

Types of Kernel Functions

7. Anova radial basis kernel

This kernel is known to perform very well in **multidimensional regression problems** just like the Gaussian and Laplacian kernels. This also comes under the category of radial basis kernel.

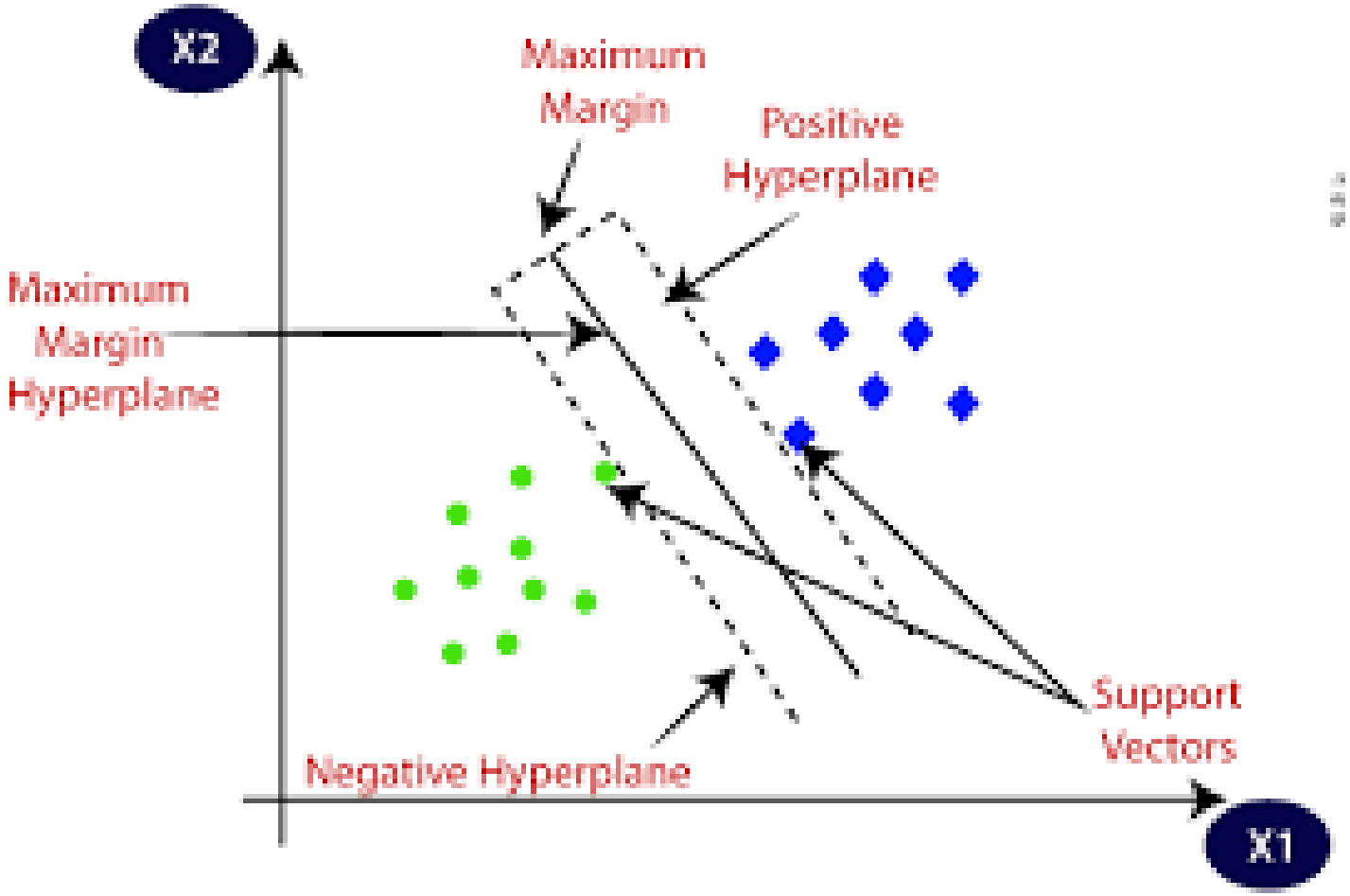
The equation for Anova kernel is :

$$k(x, y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

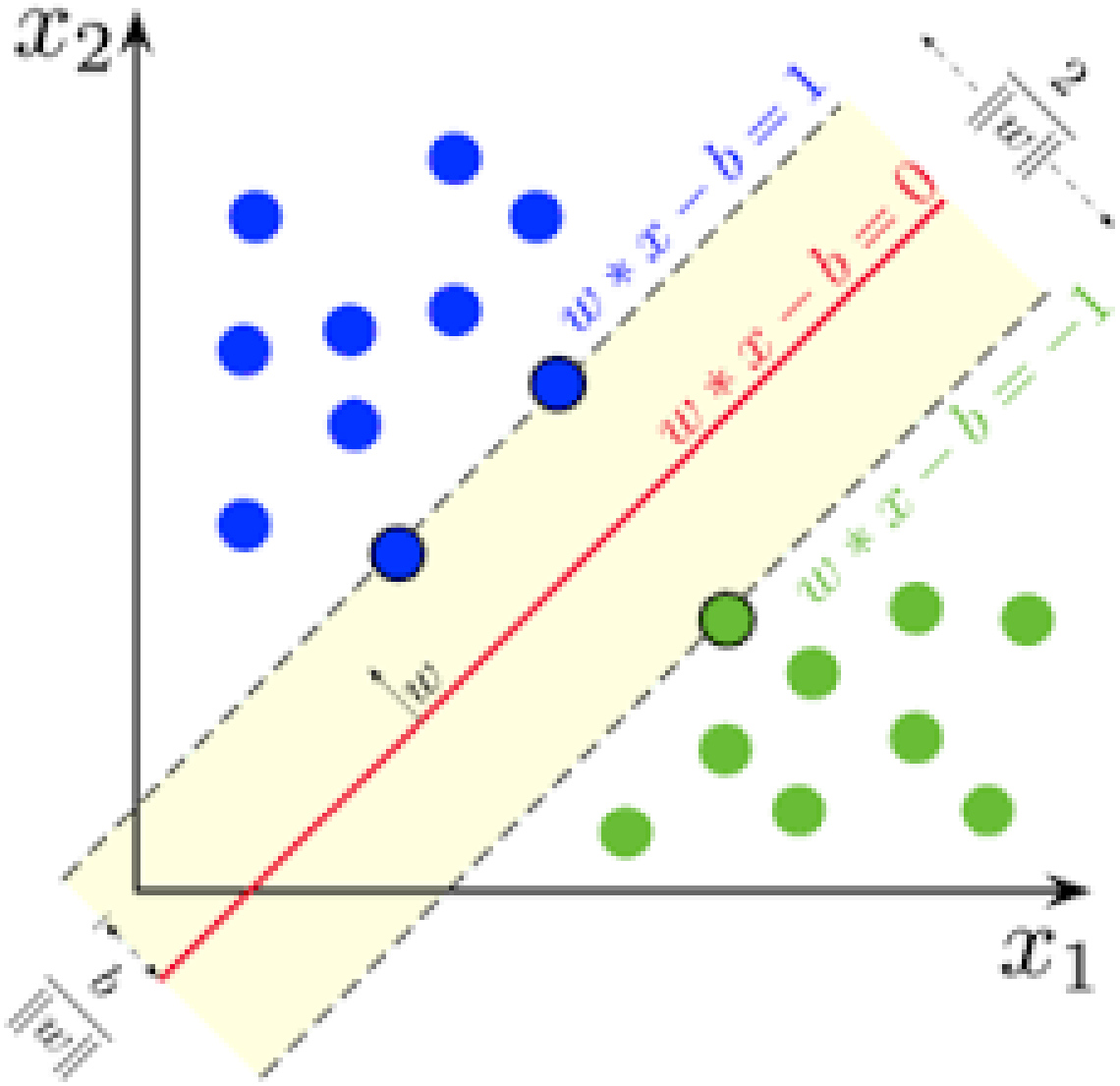
Support Vector Machine

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
- However, primarily, it is used for Classification problems in Machine Learning.
- 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 hyper plane.

Support Vector Machine



Support Vector Machine

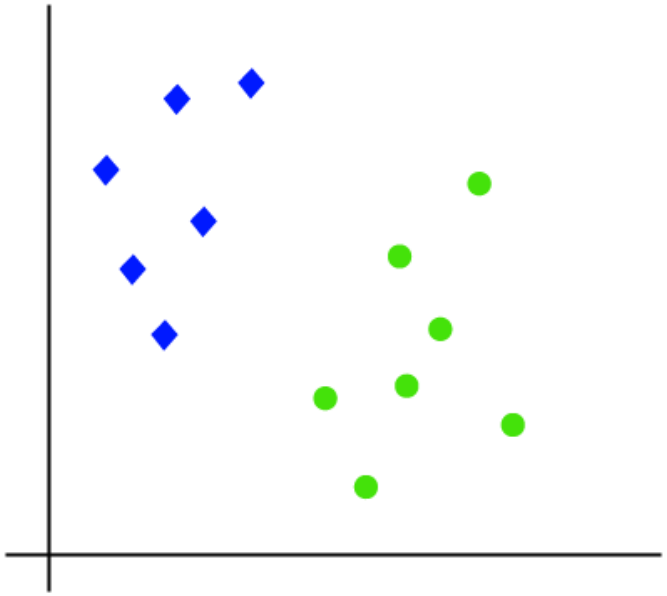


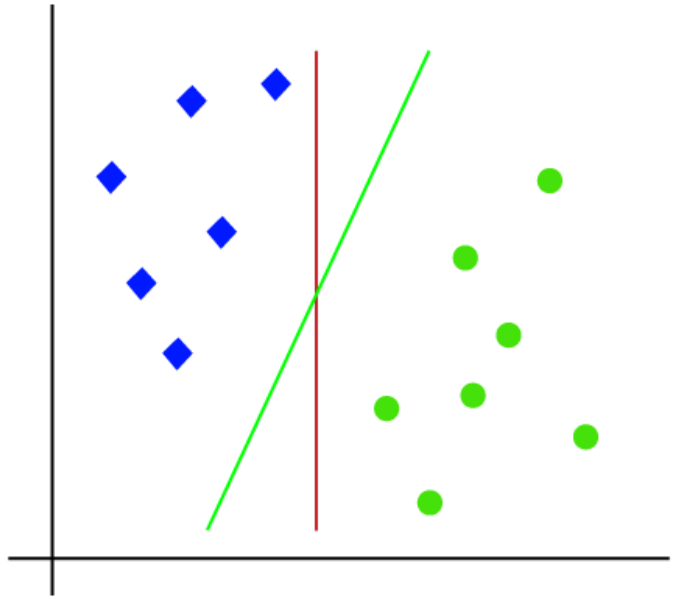
Types of Support Vector Machine

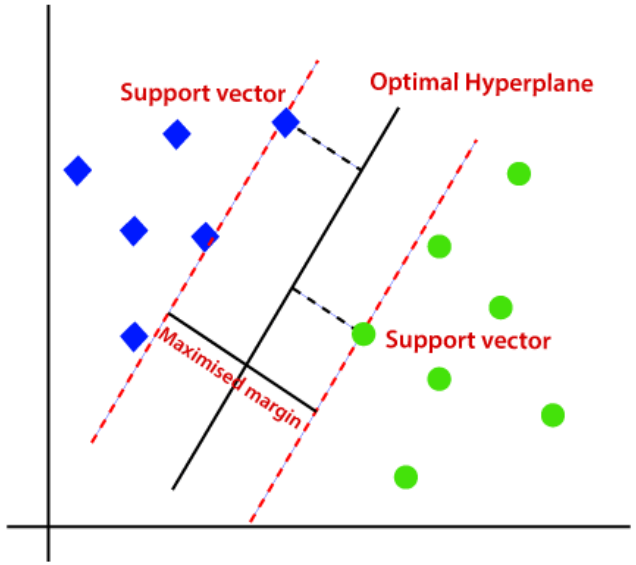
- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Linear SVM

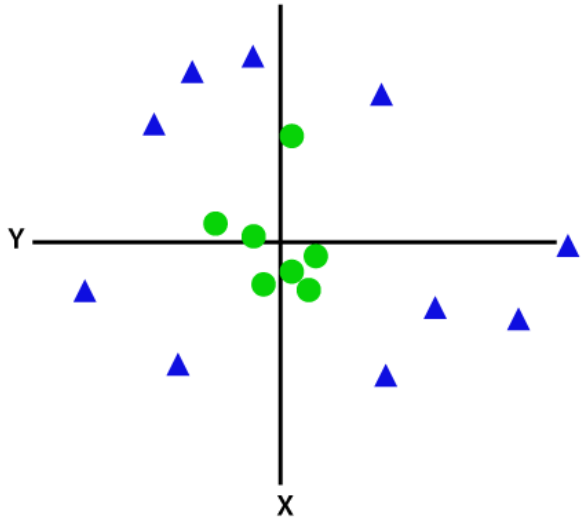





Support vector
Optimal Hyperplane
Maximised margin
Support vector

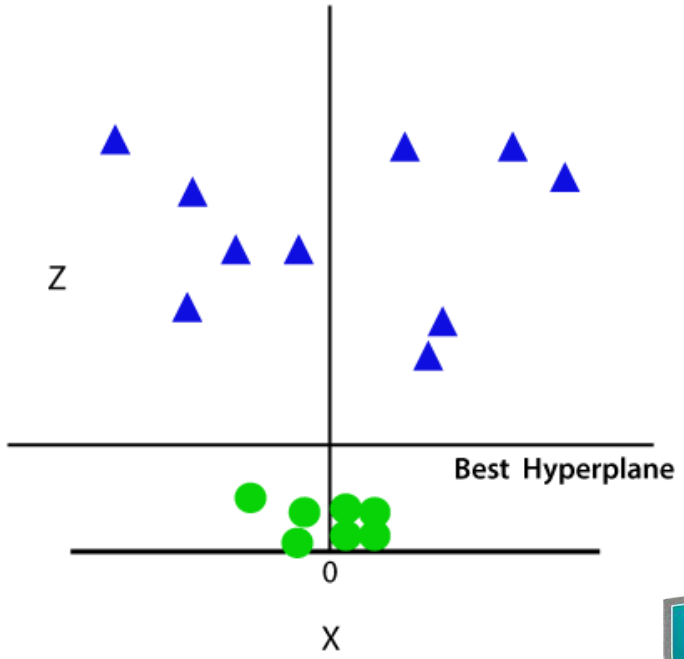
Non-linear SVM

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.



So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

z=x2 +y2



SVM vs SVC

If the hyperplane classifies the dataset linearly then the algorithm we call it as SVC and the algorithm that separates the dataset by non-linear approach then we call it as SVM.

Implementation of Linear and Non-linear SVM

Pulsar Dataset

https://colab.research.google.com/drive/1FnOLEq0G-T-0e5OYpWh9yMp5FxJgyUVG?usp=sharing

Kernel Ridge Regression

Kernel Ridge Regression

- Kernel ridge regression models are nonparametric regression models that are capable of modeling linear and nonlinear relationships between predictor variables and outcomes.
- Results can be highly sensitive to choices of model hyperparameters.
- Kernel Ridge Regression facilitates choice of hyperparameter values through k-fold cross-validation on specified grids of values using the **sklearn.model_selection.GridSearchCV** class.

$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

When Linear Regression Fails

- Linear regression is ubiquitous and it should be a first go-to when trying to fit data.

- For the next example, we have generated a larger database, with 21 points, in which y is calculated as:

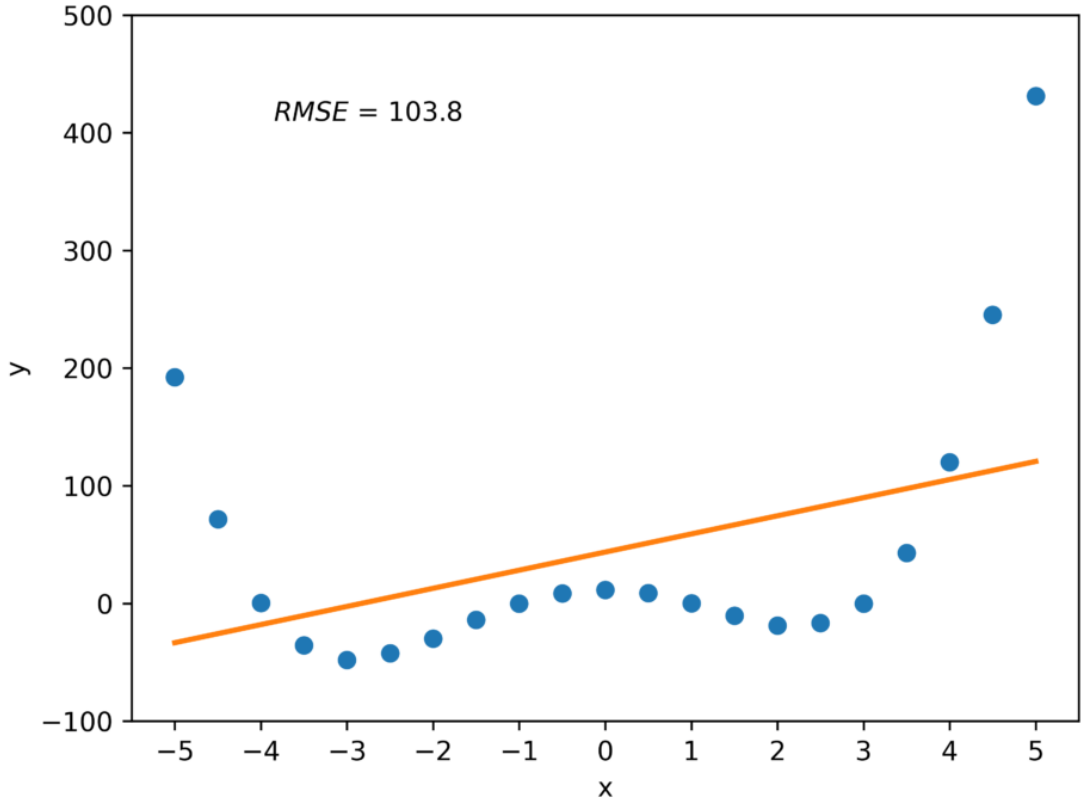
- $y=(x+4)\cdot(x+1)\cdot(x-1)\cdot(x-3)+rnd(-1,1)$
- This means that y is calculated as a 4th order polynomial plus a random variation in the interval [-1,1]. This dataset, along with the resulting linear regression is shown in the Figure below.

xn (arbitrary units)	yn (arbitrary units)
-5.00	-192.239
-4.50	71.537
-4.00	0.537
-3.50	-35.671
-3.00	-48.052
-2.50	-42.445
-2.00	-29.914
-1.50	-13.828
-1.00	-0.084
-0.50	8.668
0.00	11.419
0.50	8.778
1.00	0.209
1.50	-10.370
2.00	-18.790
2.50	-16.722
3.00	-0.018
3.50	42.934
4.00	120.075
4.50	245.389
5.00	431.082

When Linear Regression Fails

```
import math
import random
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Initialize lists and set random seed
list_x = []
list_y = []
list_x_pred = []
random.seed(2020)
# Create database with 21 points following quasi-lienar relation in interval x:[-5,5]
for i in range(-10,11):
   x = i/2
   rnd_number= random.uniform(-1,1)
   y = (x+4)*(x+1)*(x-1)*(x-3) + rnd_number
   list_x.append(x)
   list_y.append(y)
   print(x,y)
# Create list with 1060 points in interval x:[-5,5]
for i in range(-50,51):
   x = 0.1*i
   list_x_pred.append(x)
# Transform lists to np arrays
list_x = np.array(list_x).reshape(-1, 1)
list_x_pred = np.array(list_x_pred).reshape(-1, 1)
# Do linear regression using database with 21 points
regr = linear_model.LinearRegression()
regr.fit(list_x,list_y)
# Calculate value of linear regressor at 1060 points in interval x:[-5,5]
list_y_pred = regr.predict(list_x_pred)
# Calculate value of linear regressor at 21 points in interval x:[-5,5]
new_y = regr.predict(list_x)
# Print rmse value
rmse = math.sqrt(mean_squared_error(new_y, list_y))
print('##############################')
print('Root Mean Squared Error: %.1f' % rmse)
# Set axes and labels
fig = plt.figure()
ax = fig.add_subplot()
ax.set_xlim(-5.5,5.5)
ax.set_ylim(-100,500)
ax.xaxis.set_ticks(range(-5,6))
ax.yaxis.set_ticks(range(-100,600,100))
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.annotate(u'$RMSE$ = %.1f' % rmse, xy=(0.15,0.85), xycoords='axes fraction')
# Plot as orange line the regression line at interval
plt.plot(list_x_pred,list_y_pred,color='C1',linestyle='solid',linewidth=2)
# Plot as blue points the original database
plt.scatter(list_x, list_y,color='C0')
# Print plot to file
file_name='Figure_2.png'
plt.savefig(file_name,format='png',dpi=600)
plt.close()
```

When Linear Regression Fails

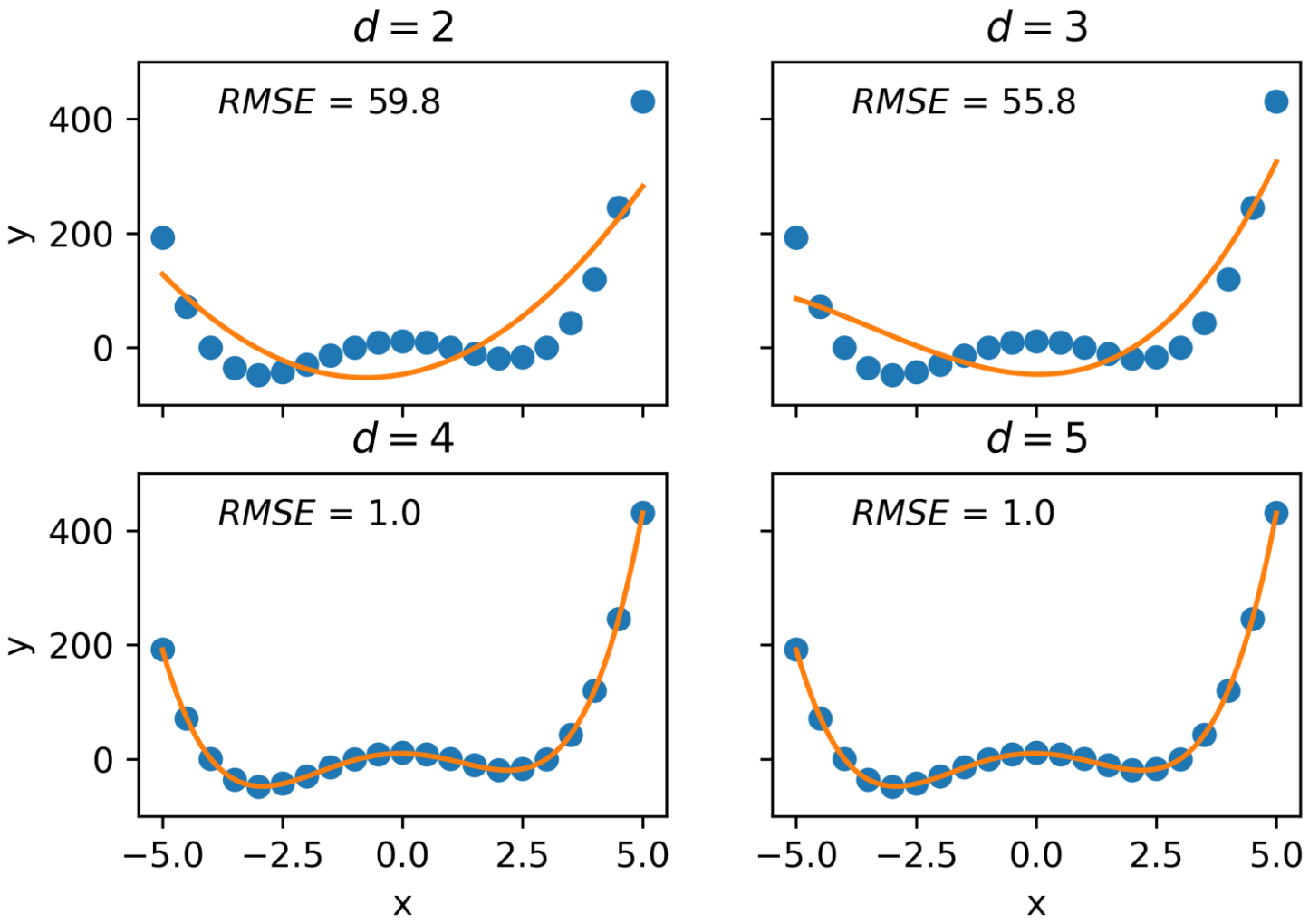


xn (arbitrary units)	yn (arbitrary units)
-5.00	-192.239
-4.50	71.537
-4.00	0.537
-3.50	-35.671
-3.00	-48.052
-2.50	-42.445
-2.00	-29.914
-1.50	-13.828
-1.00	-0.084
-0.50	8.668
0.00	11.419
0.50	8.778
1.00	0.209
1.50	-10.370
2.00	-18.790
2.50	-16.722
3.00	-0.018
3.50	42.934
4.00	120.075
4.50	245.389
5.00	431.082

Kernel Ridge Regression

- https://colab.research.google.com/drive/16ZVcmznj8Sxfs8AqnLv4biYEC4goV30q?usp=sharing

Kernel Ridge Regression after Regularization



Summary of Kernel Ridge Regression

- Ridge Regression is a technique used when the data suffers from multicollinearity (independent variables are highly correlated).
- The assumptions of this regression is same as least squared regression except normality is not to be assumed.
- Ridge regression shrinks the value of coefficients but doesn't reaches zero, which suggests no feature selection feature.
- This is a regularization method and uses L2 regularization.
- L2 regularization disperse the error terms in all the weights that leads to more accurate customized final models.

Lasso Regression

- Similar to Ridge Regression, Lasso (Least Absolute Shrinkage and Selection Operator) also penalizes the absolute size of the regression coefficients. In addition, it is capable of reducing the variability and improving the accuracy of linear regression models.
- **Lasso regression differs from ridge regression in a way that it uses absolute values in the penalty function, instead of squares.** This leads to penalizing (or equivalently constraining the sum of the absolute values of the estimates) values which causes some of **the parameter estimates to turn out exactly zero.** Larger the penalty applied, further the estimates get shrunk towards absolute zero.

$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}$$

Lasso Regression

- The assumptions of lasso regression is same as least squared regression except normality is not to be assumed.

- Lasso Regression shrinks coefficients to zero (exactly zero), which certainly helps in feature selection.

- Lasso is a regularization method and uses L1 regularization.
- If group of predictors are highly correlated, lasso picks only one of them and shrinks the others to zero.

- L1 regularization gives output in binary weights from 0 to 1 for the model's features and is adopted for decreasing the number of features in a huge dimensional dataset.
- Ex: F1: x + 2y + 10z + 5h + 30g = 100
- F2: 10z + 30g = 120

Linear, Ridge and Lasso Regression Comparison

- https://colab.research.google.com/drive/18CKUXARGsBtu9ZOt3-sa4RgOfiEzM5tW?usp=sharing

Structure Kernels

Kernel PCA

Principal Component Analysis - PCA

- It is a tool which is used to reduce the dimension of the data.
- It allows us to reduce the dimension of the data without much loss of information.
- PCA reduces the dimension by finding a few orthogonal linear combinations (principal components) of the original variables with the largest variance.
- sklearn.decomposition.KernelPCA
- PCA is used to reduce the curse of dimensionality. Or, in a more neutral tone, PCA is described to be an unsupervised decomposition algorithm.

Curse of Dimensionality

- Dimensionality in a dataset becomes a severe impediment to achieve a reasonable efficiency for most algorithms. Increasing the number of features does not always improve accuracy.

- When data does not have enough features, the model is likely to underfit, and when data has too many features, it is likely to overfit. **Hence it is called the curse of dimensionality**.

- The curse of dimensionality is an astonishing paradox for data scientists, based on the exploding amount of n-dimensional spaces — as the number of dimensions, n, increases.

Sparseness

- The sparseness of data is the property of being scanty or scattered. It lacks denseness, and its high percentage of the variable's cells do not contain actual data. Fundamentally full of "empty" or "N/A" values.

- Points in an n-dimensional space frequently become sparse as the number of dimensions grows. The distance between points will extend to grow as the number of dimensions increases.

c1	c2	c3	c4	c5
0	0	0	5	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
3	0	0	0	0
0	0	0	0	0

Dimensionality Reduction

- Dimensionality reduction eliminates some features of the dataset and creates a restricted set of features that contains all of the information needed to predict the target variables more efficiently and accurately.
- Reducing the number of features normally also reduces the output variability and complexity of the learning process.
- The covariance matrix is an important step in the dimensionality reduction process. It is a critical process to check the correlation between different features.

Dimensionality Reduction

- There are two ways of dimensionality reduction:
- Feature Selection
- Feature Extraction

- Dimensionality Reduction can ignore the components of lesser significance.

- **Feature Selection**
- In feature selection, usually, a **subset of original features is selected**.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_K} \end{bmatrix}$$

K<<N

Dimensionality Reduction

- **Feature Extraction**
- In feature extraction, **a set of new features are found**. That is found through some mapping from the existing features. Moreover, mapping can be either linear or non-linear.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix} \xrightarrow{f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

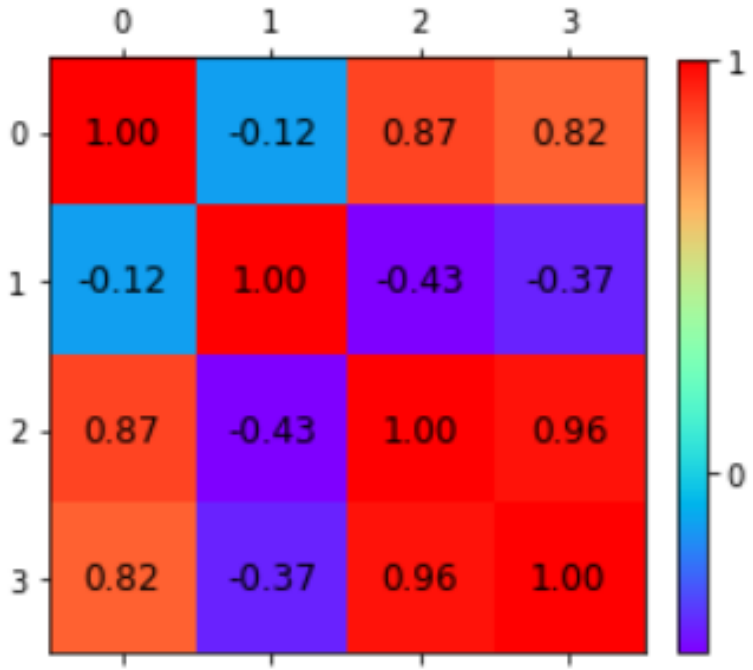
K<<N

Correlation Heat map

- Correlation varies from -1 to +1

To be precise,

- Values that are close to +1 indicate a positive correlation.
- Values close to -1 indicate a negative correlation.
- Values close to 0 indicate no correlation at all.



Kernel Principal Component Analysis - KPCA

- PCA is a linear method. That is, it can only be applied to datasets which are linearly separable. It does an excellent job for datasets, which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction.
- Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines.
- There are various kernel methods like linear, polynomial, and gaussian.

Steps in KPCA

1. First we will choose a kernel functions k(x_i, x_j) and let T be any transformation to a higher dimension.

2. And like PCA, we will find the covariance matrix of our data. But here, we will use kernel function to calculate this matrix. So will compute kernel matrix, which is the matrix that results from applying kernel function to all pairs of data.
   K=T(X)T(X)^T

3. Center our kernel matrix (this equivalent to subtract the mean of the transformed data and dividing by standard deviations) :
   K_new = K - 2(I)K + (I)K(I)
   where I is a matrix that its all elements are equal to i/d.

Steps in KPCA

4. Then, we will find eigenvectors and eigenvalues of this matrix.

5. Sort our eigenvectors based on their corresponding eigenvalues in a decreasing order.

6. We will choose what number of dimensions that we want our reduced dataset to be, let's call it m. Then we will choose our first m eigenvectors and concatenate them in one matrix.

7. Finally, Calculate the product of that matrix with your data. The result will be your new reduced dataset.

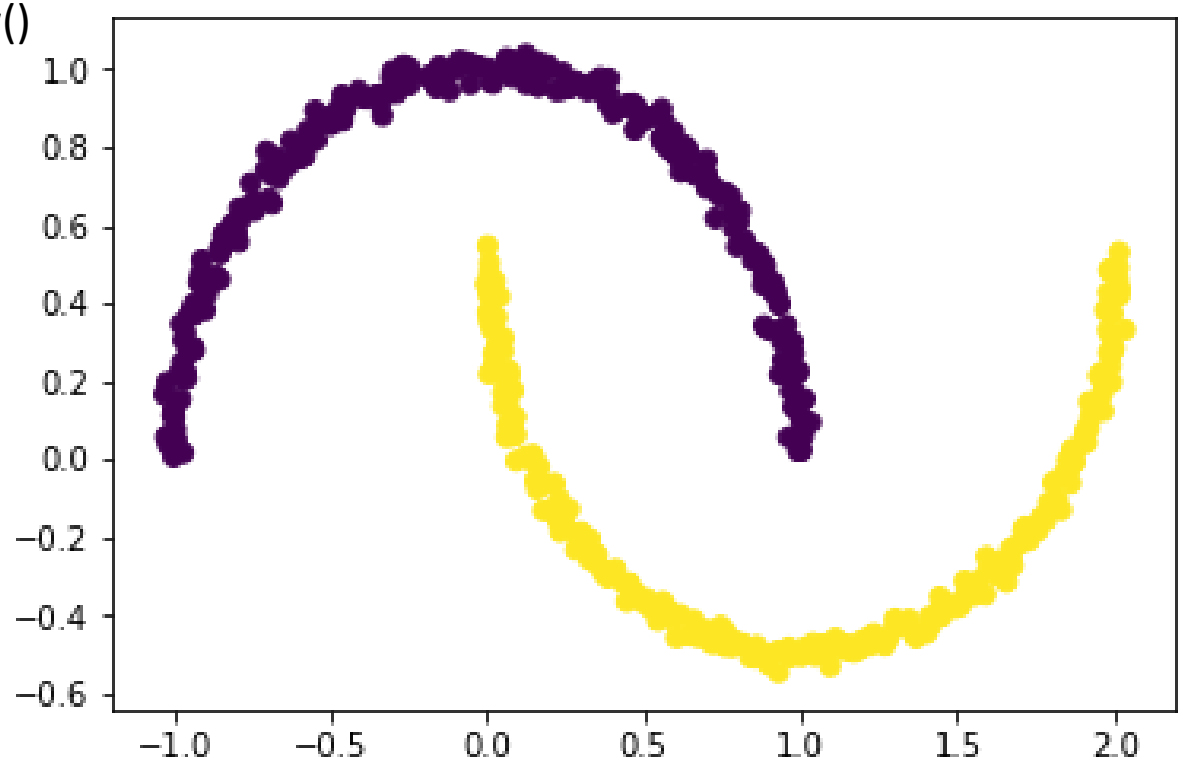
Applications of PCA

- These are the typical applications of PCA:

- Data Visualization.
- Data Compression.
- Noise Reduction.
- Data Classification.
- Image Compression.
- Face Recognition.

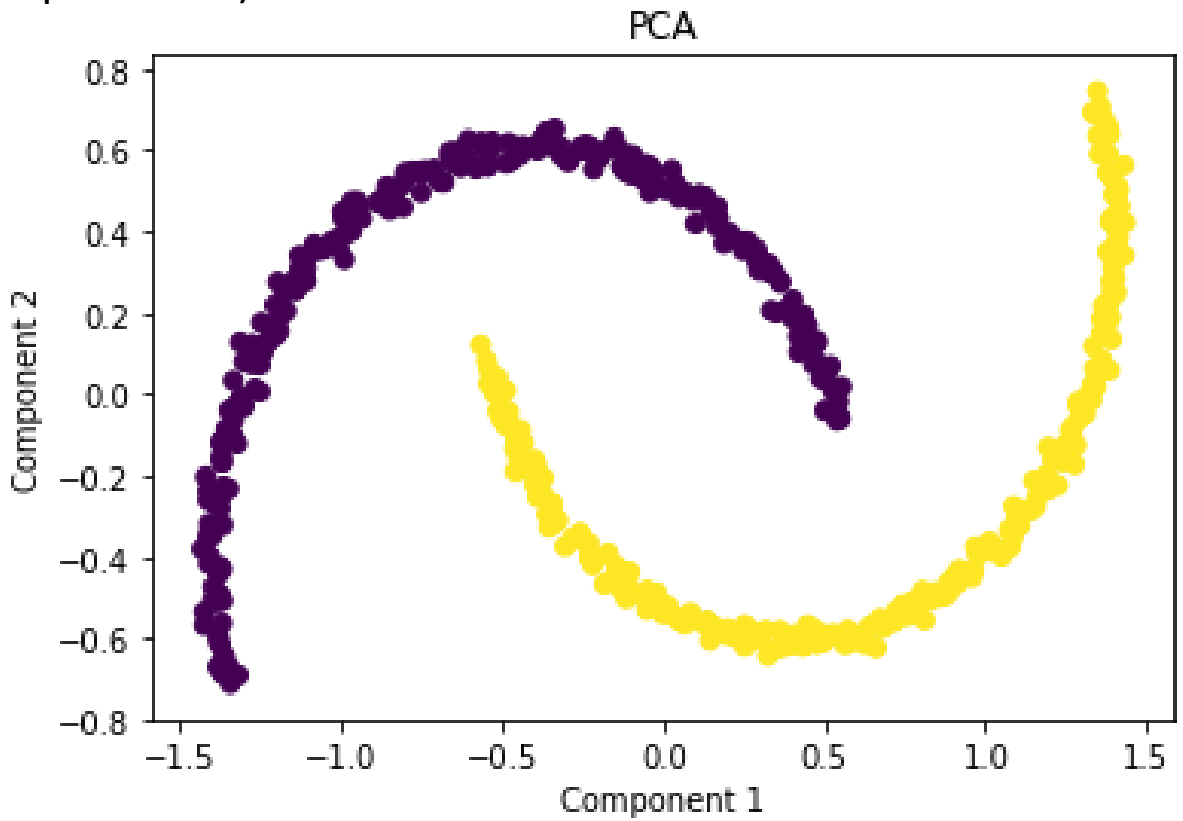
Implementation of PCA, KPCA

```
#Create non-linear dataset
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons

X, y = make_moons(n_samples = 500, noise = 0.02,
random_state = 417)

plt.scatter(X[:, 0], X[:, 1], c = y)
plt.show()
```



Implementation of PCA, KPCA

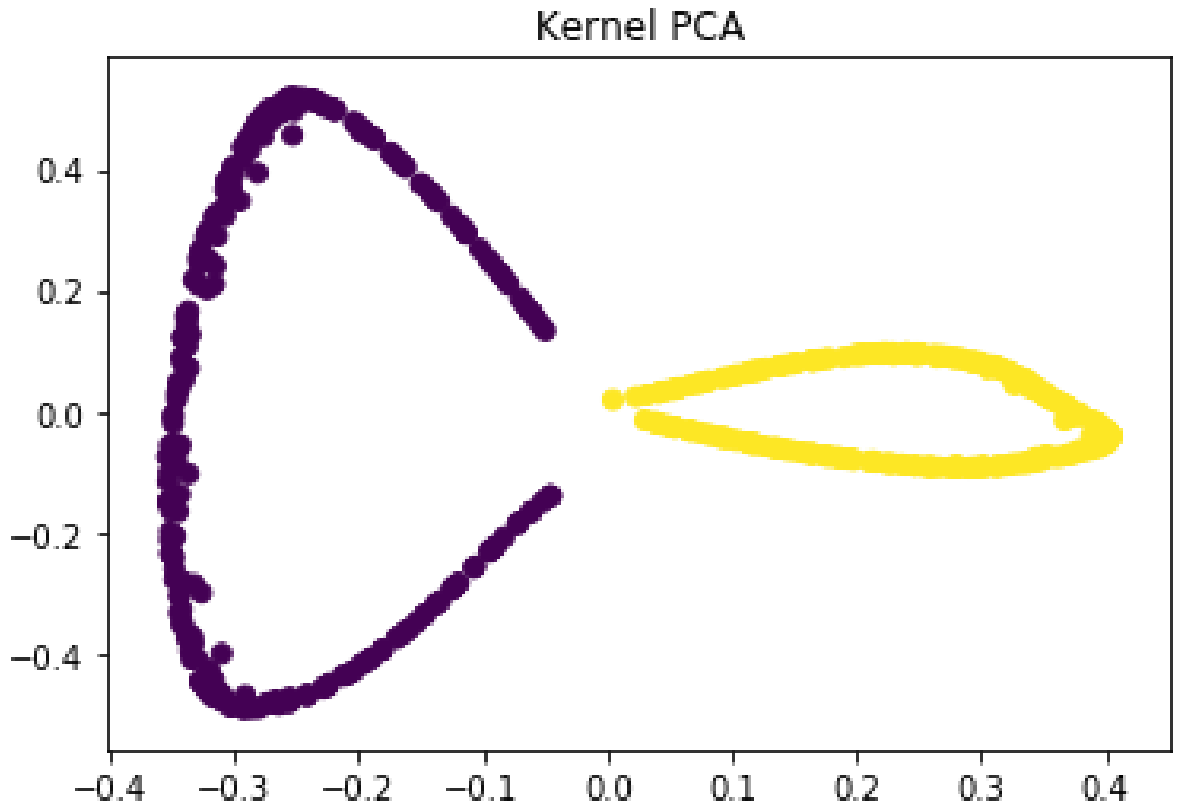
```
#Apply PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_pca = pca.fit_transform(X)

plt.title("PCA")
plt.scatter(X_pca[:, 0], X_pca[:, 1], c = y)
plt.xlabel("Component 1")
plt.ylabel("Component 2")
plt.show()
```



Implementation of PCA, KPCA

#Applying kernel PCA on this dataset with RBF kernel with a gamma value of 15.

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(kernel ='rbf', gamma = 15)
X_kpca = kpca.fit_transform(X)

plt.title("Kernel PCA")
plt.scatter(X_kpca[:, 0], X_kpca[:, 1], c = y)
plt.show()
```



Implementation of PCA, KPCA

https://colab.research.google.com/drive/1yJRvr1gjPVrqpSVocOeH7BFm_h31ja81?usp=sharing

https://colab.research.google.com/drive/1ydtkbihsYhOxjUCobR4JF_fotf81BAtV?usp=sharing

Latent Semantic Analysis

Introduction

- Analyzing texts is far more complicated than analyzing typical tabulated data (e.g. retail data) because texts fall under unstructured data.

- Textual data, even though very important, vary considerably in lexical and morphological standpoints.

- **Natural Language Processing:** A family of techniques used to derive meaning from text data.
- **Document**: A collection of words — the instances or "rows" of our dataset.
- **Body:** A collection of documents—our entire data set
- **Dictionary:** The set of all words that appear in at least one document in our body.
- **Topic:** A collection of words that co-occur
- **Latent:** Features that are "**hidden**" in the data which can not be directly measured. These features are essential to the data, but are not original features of the data set.

LSA

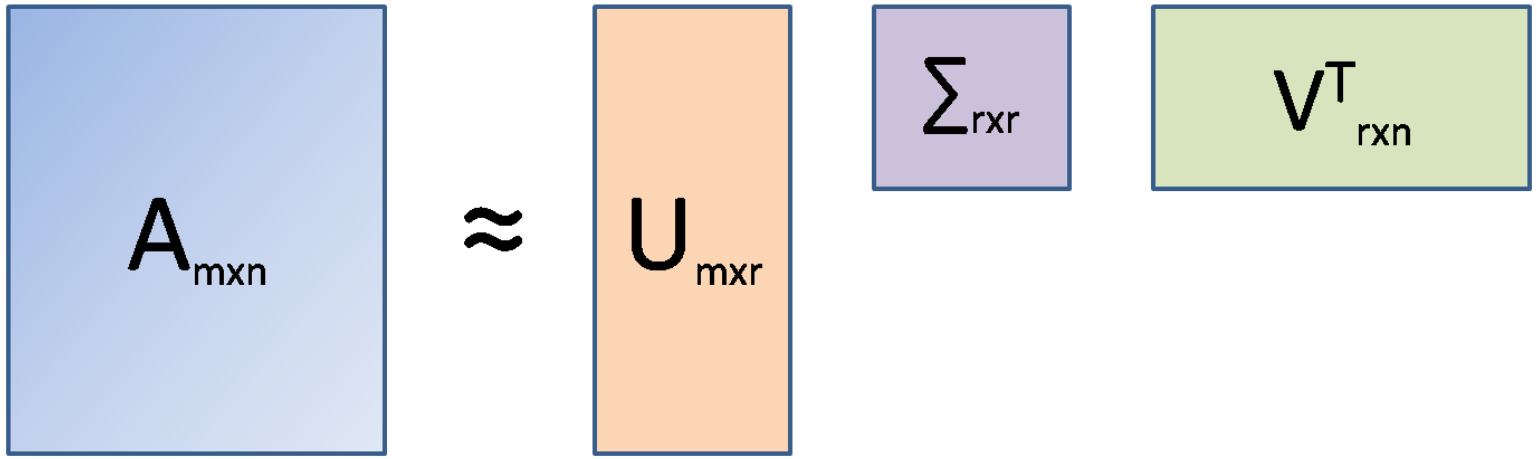
- Latent Semantic Analysis is a natural language processing method that uses the statistical approach to identify the association among the words in a document.
- **Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text.**

- LSA is an information retrieval technique which analyzes and identifies the pattern in **unstructured collection of text** and the relationship between them.

- LSA itself is an **unsupervised** way of uncovering synonyms in a collection of documents.
- In other words, word frequencies in different documents play a key role in extracting the latent topics. LSA tries to extract the dimensions using a machine learning algorithm called Singular Value Decomposition or SVD.

SVD

- Singular Value Decomposition or SVD is essentially a matrix factorization technique. In this method, any matrix can be decomposed into three parts as shown below.



Here, A is the document-term matrix
(documents in the **rows(m)**,
unique words in the **columns(n)**, and frequencies at the intersections of documents and words).
It is to be kept in mind that in LSA, the original document-term matrix is approximated by way of multiplying three other matrices, i.e., U, ∑ and VT. Here, **r** is the number of aspects or topics.
Once we fix r (r<<n) and run SVD, the outcome that comes out is called Truncated SVD and LSA is essentially a truncated SVD only.

SVD

- SVD is used in such situations because, unlike PCA, SVD does not require a correlation matrix or a covariance matrix to decompose. In that sense, SVD is free from any normality assumption of data (covariance calculation assumes a normal distribution of data).
- The U matrix is the document-aspect matrix, V is the word-aspect matrix, and ∑ is the diagonal matrix of the singular values. Similar to PCA, SVD also combines columns of the original matrix linearly to arrive at the U matrix.
- To arrive at the V matrix, SVD combines the rows of the original matrix linearly.
- Thus, from a sparse document-term matrix, it is possible to get a dense document-aspect matrix that can be used for either document clustering or document classification using available ML tools.
- The V matrix, on the other hand, is the word embedding matrix (i.e. each and every word is expressed by r floating-point numbers) and this matrix can be used in other sequential modeling tasks.
- However, for such tasks, **Word2Vec** and **Glove vectors** are available which are more popular.

TFIDF

- TF-IDF is an information retrieval technique that weighs a **Term's Frequency (TF) and its Inverse Document Frequency (IDF)**. Each word has its respective TF and IDF score. The product of the TF and IDF scores of a word is called the TFIDF weight of that word.
- Put simply, the higher the TFIDF score (weight), the rarer the word and vice versa.

Peanuts = ***0.3799546206033913 6***

Jumbo = *0.530965343023095*

Error = ***0.2302711360436964***

Among the three words, "peanut", "jumbo" and "error", tf-idf gives the highest weight to "jumbo". Why? This indicates that "jumbo" is a much rarer word than "peanut" and "error". This is how to use the tf-idf to indicate the importance of words or terms inside a collection of documents.

LSA Demo

- https://colab.research.google.com/drive/1p8gzPONscrgZwJeqQr4g_bY7Tez3pQL-?usp=sharing

Research Paper References

- **A Review of Kernel Methods in ML**
- https://drive.google.com/file/d/1vyPm7fHqdqZU9xG6ODUDJdo_h2BZzoV4/view?usp=sharing
- **A comprehensive survey on support vector machine classification_Applications_Challenges_Trends**
- https://drive.google.com/file/d/18xSI9uYFuwS-HKpZLMAGMtX4IV2UevoC/view?usp=sharing
- **Support Vector Machines_Theory and Applications**
- https://drive.google.com/file/d/1Xs_6kxsGizMpFbpfNHTQdn33PoGGrgZ-/view?usp=sharing
- **Extending latent semantic analysis to manage its syntactic blindness**
- https://drive.google.com/file/d/1GGzuPaZJLlfjIOP7PDdeSRfVM1dT7UwF/view?usp=sharing
- **Using Latent Semantic Analysis to Identify Research Trends in OpenStreetMap**
- https://drive.google.com/file/d/1vBczyhyNdszeEz1SjaxLGYg-VMJce-6o/view?usp=sharing

Up Next

- Bayesian Methods for using Prior Knowledge and Data
- Bayesian Inference
- Bayesian Belief Networks and Graphical Models
- Probabilistic Latent Semantic Analysis
- The Expectation-Maximization(EM) Algorithm
- Gaussian Processes

End of Unit 2

Thank You.