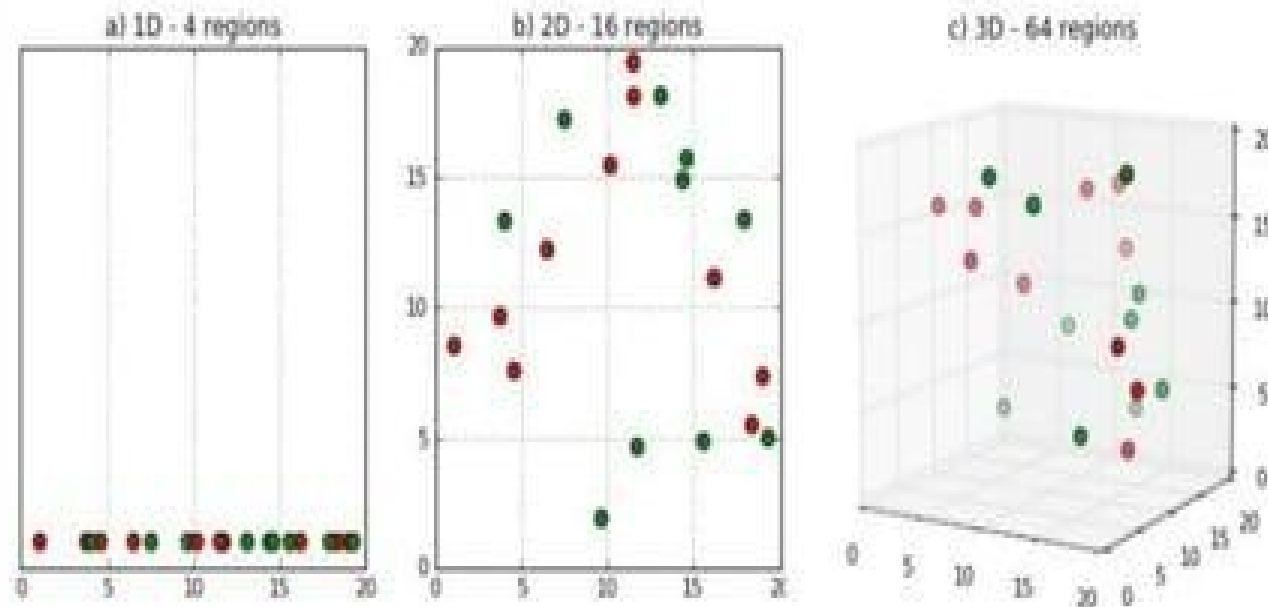


Principal Component Analysis and LDA



Curse of Dimensionality

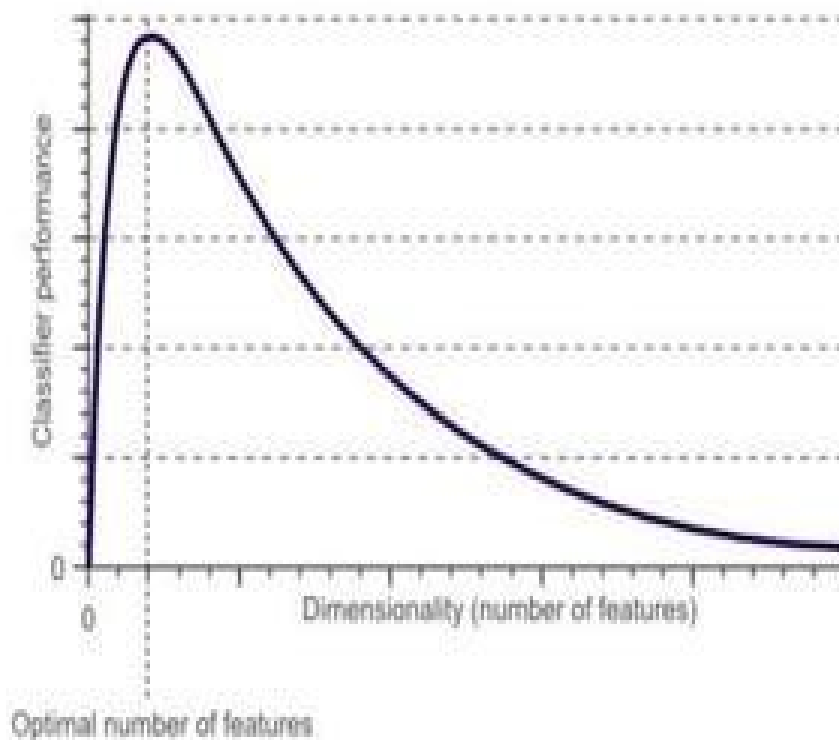


Growth is exponential and everytime we increase the number of dimensions, more data will have to be added to fill the empty spaces.

This exponential growth in data causes high sparsity in the data set and unnecessarily increases storage space and processing time for the particular modelling algorithm. Think of image recognition problem of high resolution images $1280 \times 720 = 921,600$ pixels i.e. 921600 dimensions. OMG. And that's why it's called Curse of Dimensionality.

Curse of Dimensionality

- Refers to the problem caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space.



OVERFITTING

Overfitting occurs when a model starts to memorize the aspects of the training set and in turn loses the ability to generalize



For some algorithms, the number of features controls over-fitting.
E.g. logistic regression

Principal Component Analysis

- A mathematical procedure with ***simple matrix operations*** from ***linear algebra and statistics*** to transform a number of *correlated variables into smaller number of uncorrelated variables* called principal components.
- Emphasize variation and bring out strong patterns in a dataset.
- To make data easy to ***explore and visualize***.
- Still contains most of the information in the large set.

Principal Component Analysis (contd...)

Figure 1A

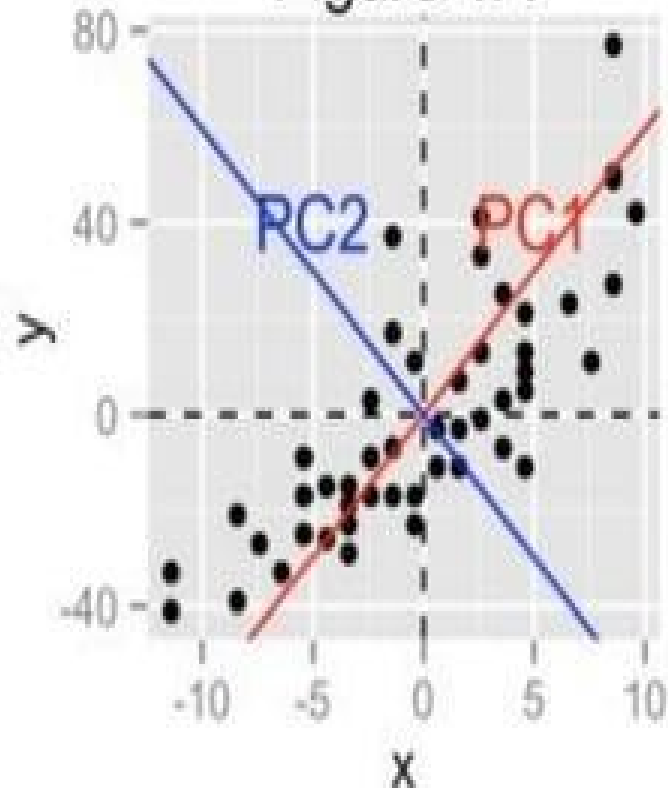


Figure 1B

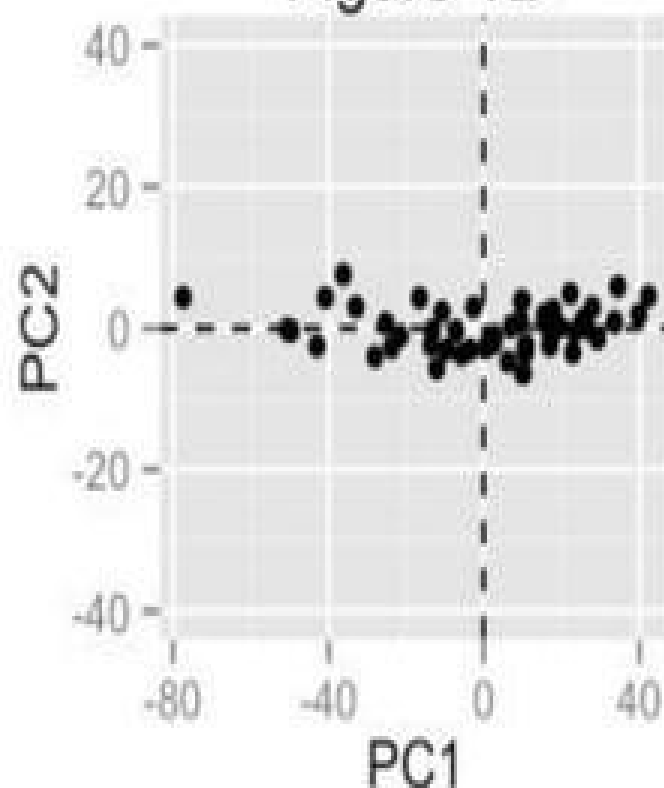


Figure 1A: The data are represented in the X-Y coordinate system

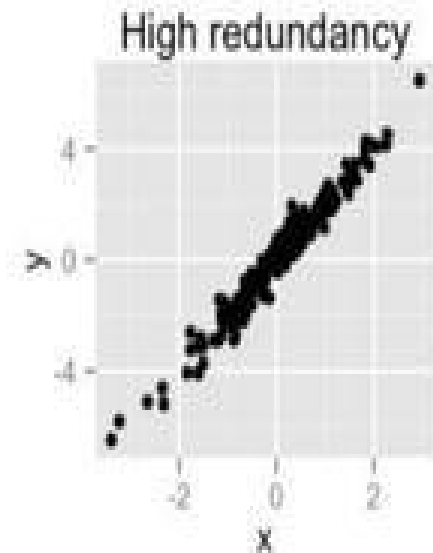
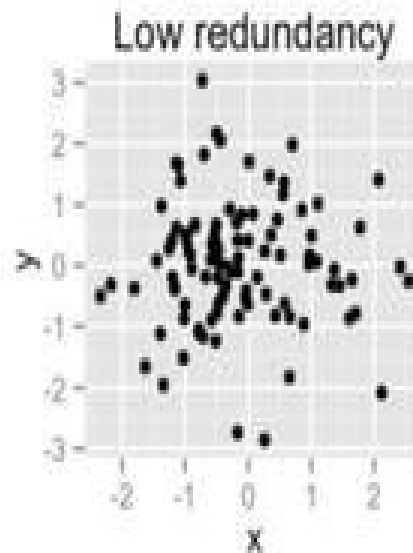
Figure 1B: The *PC1* axis is the **first principal direction** along which the samples show the largest variation

Goals of PCA

- To *identify hidden pattern* in a data set
- To *reduce the dimensionality* of the data by removing the noise and redundancy in the data
- To identify *correlated variables*

Principal Component Analysis (contd...)

- PCA method is particularly useful when the variables within the data set are highly correlated.
- **Correlation** indicates that there is **redundancy** in the data.
- PCA can be used to reduce the original variables into a smaller number of new variables (= **principal components**) explaining most of the variance in the original variables.



Steps to implement PCA in 2D dataset

Step 1: **Normalize** the data

Step 2: Calculate the **covariance matrix**

Step 3: Calculate the **eigenvalues and eigenvectors**

Step 4: **Choosing** principal components

Step 5: Forming a **feature vector**

Step 6: **Forming Principal Components**

Step 1: Normalization

This is done by subtracting the respective means from the numbers in the respective column. So if we have two dimensions X and Y, all X become x_- and all Y become y_- .

For all X; $x_- = X - \mu_x$

For all Y; $y_- = Y - \mu_y$

This produces a dataset whose mean is zero.

Step 2: Calculation of correlation

$$\text{Matrix (covariance)} = \begin{bmatrix} \text{var}(x) & \text{var}(x, y) \\ \text{var}(y, x) & \text{var}(y) \end{bmatrix}$$

Covariance Matrix for Iris Dataset

	<i>Sepal.Length</i>	<i>Sepal.Width</i>	<i>Petal.Length</i>	<i>Petal.Width</i>
<i>Sepal.Length</i>	0.69	-0.04	1.27	0.52
<i>Sepal.Width</i>	-0.04	0.19	-0.33	-0.12
<i>Petal.Length</i>	1.27	-0.33	3.12	1.30
<i>Petal.Width</i>	0.52	-0.12	1.30	0.58

Calculation of correlation(contd...)

If **x** and **y** be two variables with length **n**,

$$\sigma_{xx}^2 = \frac{\sum_i (x_i - m_x)(x_i - m_x)}{n - 1}$$

The variance of **x** , variance of **y** and variance of **x & y** is given by following equations.

$$\sigma_{yy}^2 = \frac{\sum_i (y_i - m_y)(y_i - m_y)}{n - 1}$$

m_x : mean of **x** variables

m_y : mean of **y** variables

$$\sigma_{xy}^2 = \frac{\sum_i (x_i - m_x)(y_i - m_y)}{n - 1}$$

Calculation of correlation (contd...)

Correlation is the index to measure how strongly two variable are related to each other. The value of the same ranges for **-1** to **+1**. (i.e. $-1 < r < 1$)

If $r < 0$, variables are ***negatively correlated*** (e.g. x increases when y increases)

If $r > 0$, variables are ***positively correlated*** (e.g. x increases when y decreases)

If $r = 0$, variables has ***no correlation***

Step 3: Calculation of Eigenvalue and Eigenvector

Calculate Eigenvalue and Eigenvector of the covariance matrix using power method:

$$|\lambda - A| = 0$$

Where, I is an identity matrix of same dimension as A
 λ is eigenvalue.

For each value of λ , corresponding eigenvector 'v' is obtained by solving:

$$(\lambda - A)v = 0$$

Step 4: Choosing Component

- Eigenvalues from largest to smallest so that it gives us the components in order of significance.
- If we have a dataset with n variables, then we have the corresponding n eigenvalues and eigenvectors.
- Eigenvector (v) corresponding to largest eigenvalue (λ) is called first principal component.
- To reduce the dimensions, we choose the first p eigenvalues and ignore the rest.

Step 5: Forming a feature vector

Form a feature vector consists of selected eigenvectors

Feature Vector = (eig1, eig2)

Step 5: Forming Principal Components

$$\text{NewData} = \text{FeatureVector}^T \times \text{ScaledData}^T$$

NewData is the Matrix consisting of the principal components,

FeatureVector is the matrix we formed using the eigenvectors we chose to keep,

and

ScaledData is the scaled version of original dataset

Sample Implementation with sklearn

Output

```
3 # Principal Component Analysis
4 from numpy import array
5 from sklearn.decomposition import PCA
6 # define a matrix
7
8 A = array([[1, 2], [3, 4], [5, 6]])
9 print(A)
10
11 # create the PCA instance
12 pca = PCA(2)
13 # fit on data
14 pca.fit(A)
15 # access values and vectors ie.covariance
16 matrix
17 print(pca.components_)
18 print(pca.explained_variance_)
19 # transform data
20 B = pca.transform(A)
21 print(B)
```

[[1 2][3 4][5 6]]

[[0.70710678 0.70710678]
[0.70710678 -0.70710678]]

[8.00000000e+00 2.25080839e-33]

[[-2.82842712e+00 2.22044605e-16]
[0.00000000e+00 0.00000000e+00]
[2.82842712e+00 -2.22044605e-16]]

Applications of PCA

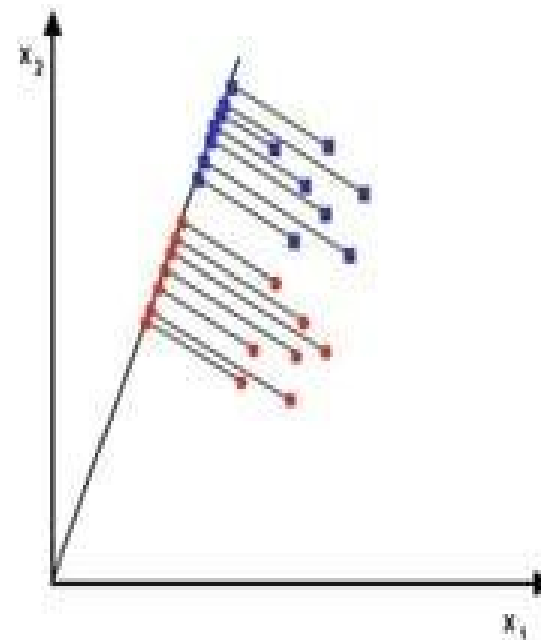
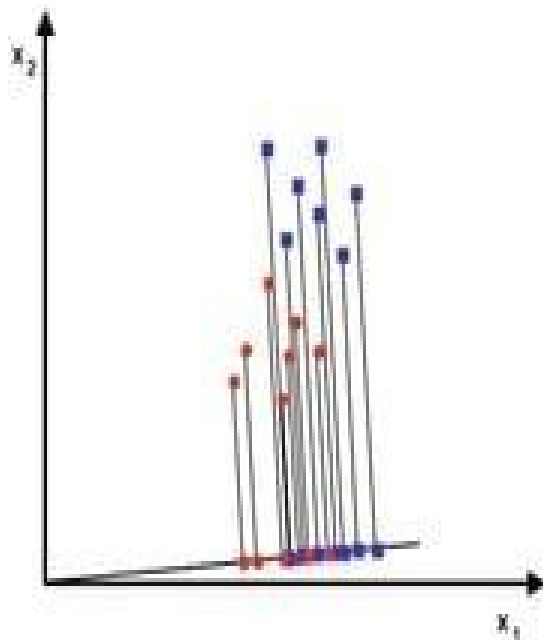
- Commonly used dimensionality reduction technique in domains like *facial recognition*, *computer vision* and *image compression*.
- Used in *finding patterns in data of high dimension* in the field of *finance*, *data mining*, *bioinformatics*, *psychology*.

Linear Discriminant Analysis (LDA)

- The objective of LDA is to perform dimensionality reduction
- However, we want to preserve as much of the class discriminatory information as possible
- LDA helps you find the boundaries around clusters of classes
- It projects your data points on a line so that your clusters are as separated as possible, with each cluster having a relative (close) distance to a centroid.
- Supervised algorithm as it takes the class label into consideration
- Assume we have a set of D -dimensional samples $x(1), x(2), \dots, x(N)$, N_1 of which belong to class ω_1 , and N_2 to class ω_2
 - We seek to obtain a scalar y by projecting the samples x onto a line $y = w^T x$
 - Of all the possible lines we would like to select the one that maximizes the separability of the scalars

Linear Discriminant Analysis (Cont...)

- It determines a new dimension which is nothing but an axis which should satisfy two criteria:
 - Maximize the distance between the centroid of each class.
 - Minimize the variation (which LDA calls scatter), within each category.



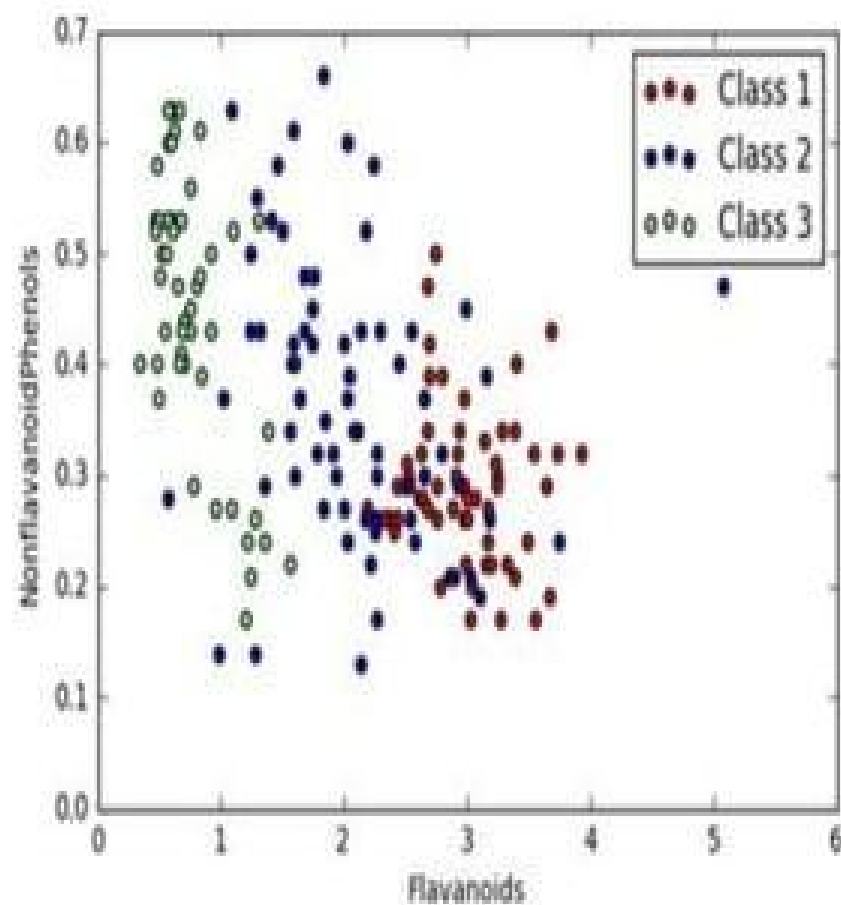


Figure 1.1 : Shows the plotting of data using two features



:)

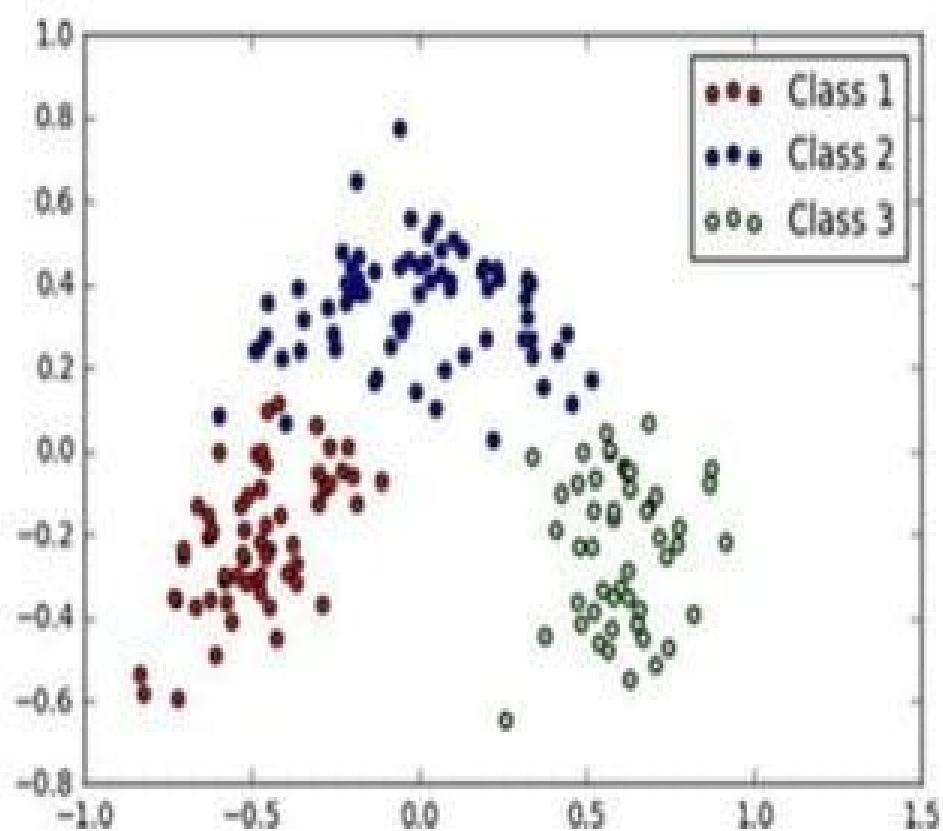
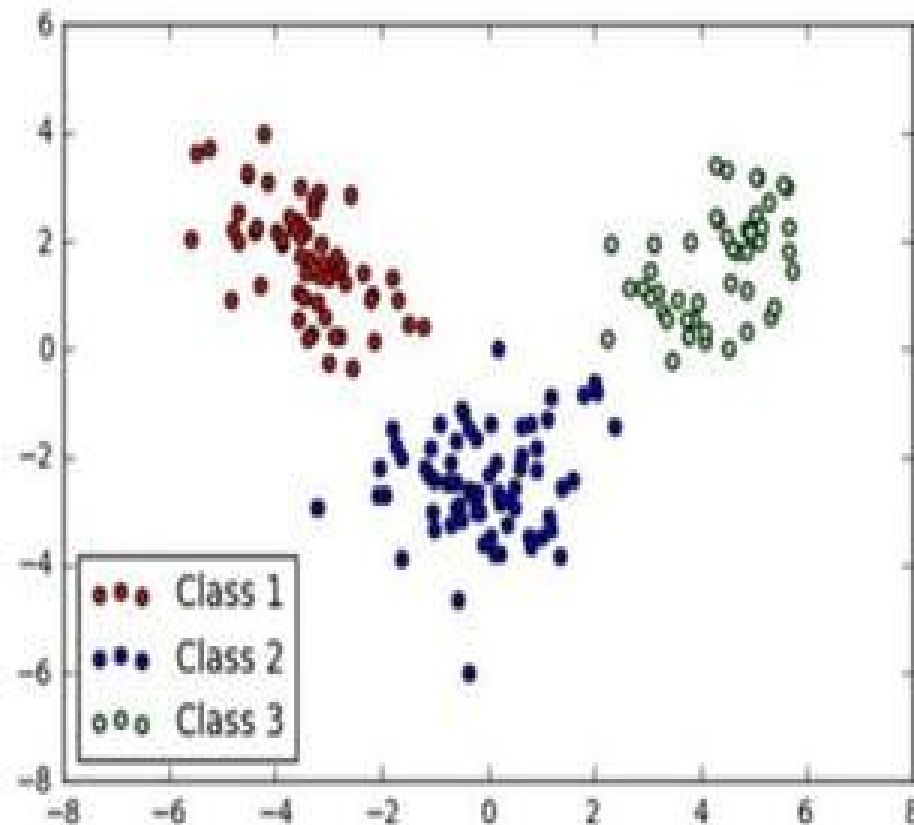


Figure 1.2 : Shows the plotting of data using PCA components



In the graph, you can visualize the whole dataset properly differentiate well among the classes. This is the major difference between PCA and LDA.

Different approaches to LDA

- Class-dependent transformation:
 - Involves maximizing the ratio of between class variance to within class variance
 - The main objective is to maximize this ratio so that adequate class separability is obtained
 - The class-specific type approach involves using two optimizing criteria for transforming the data sets independently.
- Class-independent transformation:
 - This approach involves maximizing the ratio of overall variance to within class variance
 - This approach uses only one optimizing criterion to transform the data sets and hence all data points irrespective of their class identity are transformed using this transform
 - In this type of LDA, each class is considered as a separate class against all other classes

Mathematical Operations

For ease of understanding, this concept is applied to a two-class problem. Each data set has 100 2-D data points.

1. Formulate the data sets and the test sets, which are to be classified in the original space. For ease of understanding let us represent the data sets as a matrix consisting of features in the form given below:

$$\mathbf{X} = \begin{bmatrix} x^{w1} & x^{w2} \\ \dots & \dots \\ \dots & \dots \\ x^{s1} & x^{s2} \\ x^{l1} & x^{l2} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y^{w1} & y^{w2} \\ \dots & \dots \\ \dots & \dots \\ y^{s1} & y^{s2} \\ y^{l1} & y^{l2} \end{bmatrix} \quad (1)$$

Mathematical Operations(Cont...)

2. Compute the mean of each data set and mean of entire data set. Let μ_1 and μ_2 be the mean of set 1 and set 2 respectively and μ_3 be mean of entire data, which is obtained by merging set 1 and set 2, is given by Equation 1.

$$\mu_3 = p_1 \times \mu_1 + p_2 \times \mu_2 \quad (2)$$

where p_1 and p_2 are the apriori probabilities of the classes. In the case of this simple two class problem, the probability factor is assumed to be 0.5.

Mathematical Operations(Cont...)

3. In LDA, within-class and between-class scatter are used to formulate criteria for class separability. Within-class scatter is the expected covariance of each of the classes

$$S_w = \sum_j p_j \times (cov_j) \quad (3)$$

Therefore, for the two-class problem,

$$S_w = 0.5 \times cov_1 + 0.5 \times cov_2 \quad (4)$$

Mathematical Operations(Cont...)

Let and be the covariance of set 1 and set 2 respectively. Covariance matrix is computed using the following equation:

$$cov_j = (x_j - \mu_j)(x_j - \mu_j)^T \quad (5)$$

The between-class scatter is computed using the following equation.

$$S_b = \sum_j (\mu_j - \mu_3) \times (\mu_j - \mu_3)^T \quad (6)$$

Mathematical Operations(Cont...)

The transformations are found as the eigen vector matrix of the different criteria defined in Equations 7 and 8

$$criterion_j = inv(cov_j) \times S_b \quad (7)$$

For the class independent transform, the optimizing criterion is computed as

$$criterion = inv(S_w) \times S_b \quad (8)$$

- An eigen vector of a transformation represents a 1-D invariant subspace of the vector space in which the transformation is applied.
- To obtain a non-redundant set of features all eigen vectors corresponding to non-zero eigen values only are considered and the ones corresponding to zero eigen values are neglected.

Mathematical Operations(Cont...)

5. Obtained the transformation matrices, we transform the data sets using the single LDA transform or the class specific transforms which ever the case may be.

For the class dependent LDA,

$$transformed_set_j = transform_j^T \times set_j \quad (9)$$

For the class independent LDA,

$$transformed_set = transform_spec^T \times data_set^T \quad (10)$$

Similarly the test vectors are transformed and are classified using the euclidean distance of the test vectors from each class mean.

Mathematical Operations(Cont...)

- Once the transformations are completed using the LDA transforms, Euclidean distance or RMS distance is used to classify data points.
- Euclidean distance is computed to get the mean of the transformed data set
- The smallest Euclidean distance among the n distances classifies the test vector as belonging to class n .

PCA vs. LDA

- LDA and PCA are linear transformation techniques: LDA is a supervised whereas PCA is unsupervised – PCA ignores class labels.
- PCA performs better in case where number of samples per class is less. Whereas LDA works better with large dataset having multiple classes; class separability is an important factor while reducing dimensionality.
- PCA does more of feature classification and LDA does data classification.
- In PCA, the shape and location of the original data sets changes when transformed to a different space whereas LDA doesn't change the location but only tries to provide more class separability and draw a decision region between the given classes

References

- **LDA and PCA for dimensionality reduction**
[<https://sebastianraschka.com/faq/docs/lda-vs-pca.html>]
- A. M. Martinez and A. C. Kak, "**PCA versus LDA**," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228-233, Feb. 2001.
- **Principal Component Analysis**
[<http://setosa.io/ev/principal-component-analysis/>]

References (contd...)

- **Introduction to Principal Components and Factor Analysis**
[<ftp://statgen.ncsu.edu/pub/thorne/molevoclass/AtchleyOct19.pdf>]
- **Introduction to PCA**
[<https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial>]
- **Linear Discriminant Analysis - A Brief Analysis**
[<http://www.sthda.com/english/wiki/print.php?id=206>]



Feature Engineering techniques
for
text, images, audio, and video.

Features extraction techniques for images

- Histogram of gradients (HOG)
- Maximally stable extremal regions (MSER)
- Scale-invariant feature transform (SIFT)
- Speeded Up Robust Features (SURF)
- Features from accelerated segment test (FAST)
- Local binary pattern (LBP)
- Local phase Quantization (LPQ)
- Edge detection techniques (Canny etc.)

1. Histogram of Oriented Gradients (HOG)


HOG is a feature descriptor that captures gradient orientation and magnitude distributions in localized image regions. It is widely used in object detection, especially pedestrian detection.

How it works:

1. **Gradient Computation:** Compute horizontal (G_x) and vertical (G_y) gradients using Sobel operators.
2. **Gradient Orientation and Magnitude:** Calculate the gradient magnitude and direction:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2}$$

$$\text{Orientation} = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

3. **Spatial Binning:** Divide the image into small cells (e.g., 8×8).
4. **Histogram Creation:** Form histograms of gradient orientations within each cell.
5. **Block Normalization:** Normalize histograms over overlapping blocks to improve invariance to illumination.
6. **Feature Vector Formation:** Concatenate histograms into a single feature vector.

Example Application:

- Used in pedestrian detection in self-driving cars.

2. Maximally Stable Extremal Regions (MSER)

MSER is a region-based feature detector that identifies stable regions under varying illumination and transformations.

How it works:

1. Convert the image to grayscale.
2. Identify connected components at different intensity thresholds.
3. Select regions that remain stable across multiple thresholds.
4. Store these as MSER regions.

Example Application:

- Used in text detection for Optical Character Recognition (OCR).



3. Scale-Invariant Feature Transform (SIFT)

SIFT detects keypoints and describes them using a robust descriptor, making it scale and rotation-invariant.

How it works:

1. **Scale-space Representation:** Apply Gaussian filters at multiple scales.
2. **Keypoint Detection:** Find extrema in the Difference of Gaussian (DoG) space.
3. **Keypoint Localization:** Use Hessian matrix to refine keypoints.
4. **Orientation Assignment:** Assign dominant gradient orientations to make descriptors rotation-invariant.
5. **Feature Descriptor Computation:** Extract histograms of gradients in local patches around keypoints.

Example Application:

- Used in image stitching for panorama creation.

4. Speeded-Up Robust Features (SURF)

SURF is a faster alternative to SIFT and uses integral images for rapid computation.

How it works:

1. **Scale-space Representation:** Uses Hessian matrix for fast keypoint detection.
2. **Orientation Assignment:** Uses Haar wavelet responses.
3. **Descriptor Generation:** Uses wavelet responses to create a 64 or 128-dimensional feature vector.

Example Application:

- Used in object recognition and tracking in real-time applications.

5. Features from Accelerated Segment Test (FAST)

FAST is a high-speed corner detection algorithm.

How it works:

1. Consider a circular region of 16 pixels around a candidate pixel.
2. If a contiguous set of pixels (e.g., 12 out of 16) are brighter or darker than the central pixel, mark it as a corner.
3. Apply non-maximum suppression to remove weak corners.

Example Application:

- Used in real-time applications like mobile augmented reality.

6. Local Binary Patterns (LBP)

LBP is a texture descriptor that encodes pixel neighborhood relationships.

How it works:

1. Divide the image into small regions.
2. For each pixel, compare its intensity with its 8-neighboring pixels.
3. Assign a binary value (1 if the neighbor is greater, 0 otherwise).
4. Convert the binary pattern into a decimal value.
5. Construct a histogram of LBP values.

Example Application:

- Used in face recognition systems.

7. Local Phase Quantization (LPQ)

LPQ is a blur-invariant texture descriptor.

How it works:

1. Apply a Short-Time Fourier Transform (STFT) to small image regions.
2. Quantize the phase information into a binary code.
3. Create a histogram of phase-coded values.

Example Application:

- Used in blur-robust biometric recognition.

8. Canny Edge Detector

Canny is an edge detection technique that finds strong edges in an image.

How it works:

1. Noise Reduction: Apply Gaussian smoothing.
2. Gradient Computation: Compute Sobel gradients.
3. Non-Maximum Suppression: Thin edges by keeping only local maxima.
4. Hysteresis Thresholding: Retain edges using high and low thresholds.

Example Application:

- Used in medical image segmentation.

Summary Table

Method	Type	Key Feature	Application
HOG	Feature Descriptor	Gradient orientation histograms	Pedestrian detection
MSER	Feature Detector	Stable connected regions	OCR text detection
SIFT	Feature Detector & Descriptor	Scale and rotation invariance	Image stitching
SURF	Feature Detector & Descriptor	Fast alternative to SIFT	Object recognition
FAST	Feature Detector	Rapid corner detection	Augmented reality
LBP	Texture Descriptor	Binary texture encoding	Face recognition
LPQ	Texture Descriptor	Blur-invariant texture features	Biometric recognition
Canny	Edge Detector	Multi-stage edge detection	Medical imaging

Textual Features

Number of words

Frequency

Parts of speech

Paragraph

Sentences

1. Number of Words

The number of words in a document helps analyze text length, readability, and complexity.

How it Works:

1. Tokenize the text: Split the text into words.
2. Count the total words, including or excluding stop words (common words like "the," "is," "and").

Example:

Text:

"Natural Language Processing is a branch of AI."

- Word Count (including stop words): 7
- Word Count (excluding stop words like "is", "a", "of"): 5

Applications:

- Used in readability analysis (e.g., Flesch-Kincaid readability score).
- Helps in spam detection (e.g., extremely short messages might be spam).

2. Word Frequency

Word frequency measures how often a word appears in a document or corpus. It is useful for text mining, keyword extraction, and sentiment analysis.

How it Works:

1. Tokenize the text into words.
2. Count the occurrences of each word.
3. Normalize (optional): Convert counts into percentages.

Example:

Text:

"Data science is fun. Data science involves statistics."

- Word Frequencies:
 - "data" → 2
 - "science" → 2
 - "is" → 1
 - "fun" → 1
 - "involves" → 1
 - "statistics" → 1

Applications:

- Used in search engines (higher frequency words help in ranking).
- Helps in sentiment analysis (e.g., frequency of "happy" vs. "sad").

4. Paragraph Detection

Paragraph detection involves segmenting text into meaningful sections. A paragraph is a collection of related sentences.

How it Works:

1. Split the text based on line breaks (`\n\n`) or indentation.
2. Count the number of paragraphs.

Example:

Text:

```
pgsql
```

[Copy](#)[Edit](#)

```
AI is transforming industries. It improves efficiency and reduces costs.
```

```
Machine learning, a subset of AI, enables computers to learn from data.
```

- Number of Paragraphs: 2

Applications:

- Used in document summarization (each paragraph may discuss a separate topic).
- Helps in document structuring (e.g., AI-based formatting).

5. Sentence Detection

Sentence detection breaks text into individual sentences for grammatical analysis.

How it Works:

1. Split text using punctuation (. ! ?) and spacing.
2. Identify sentence boundaries carefully (e.g., "Dr. Smith is here." is one sentence, not tw

Example:

Text:

"NLP is amazing! It helps machines understand language. What do you think?"

- Sentences:
 1. NLP is amazing!
 2. It helps machines understand language.
 3. What do you think?
- Sentence Count: 3

Applications:

- Used in chatbots and voice assistants (breaking responses into sentences).
- Helps in grammar checking (e.g., Grammarly detects run-on sentences).

Features extraction techniques for textual data

Bag of words

Term frequency-inverse document frequency

Word Embeddings

Feature Extraction Techniques for audio data

Features extraction techniques for audio data

- Mel frequency cepstral coefficients (MFCCs)
- Linear Prediction Coefficient (LPC)
- Linear Prediction Cepstral Coefficients (LPCC)
- Line Spectral Frequencies (LSF)
- Discrete Wavelet Transform (DWT)
- Perceptual Linear Prediction (PLP)