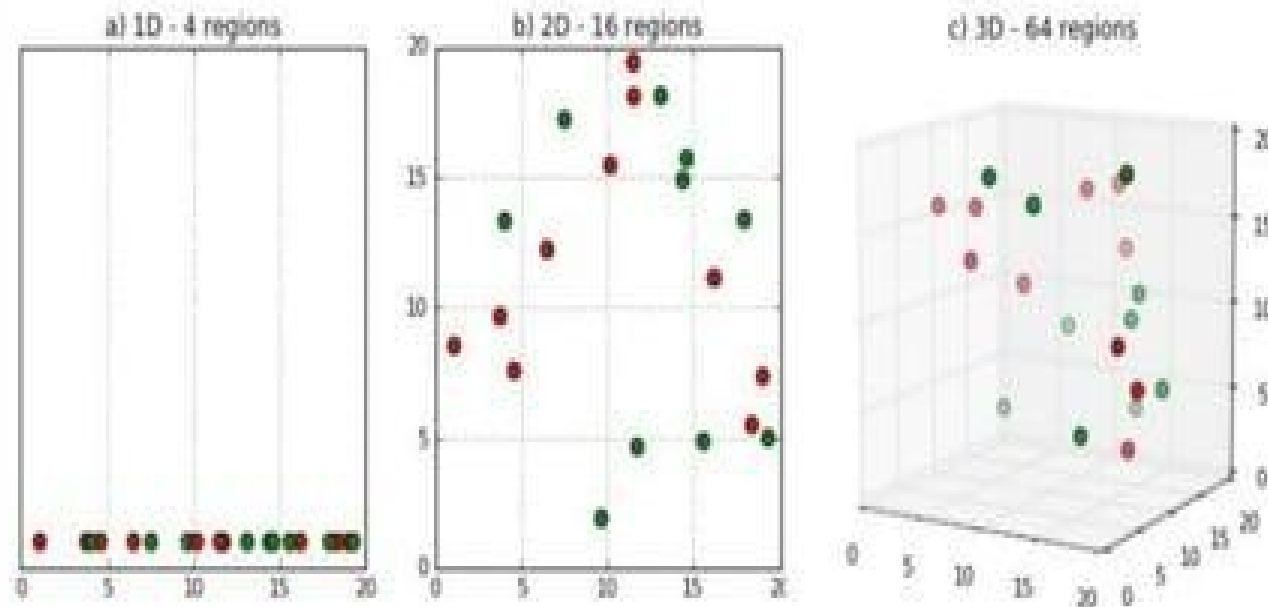# Principal Component Analysis and LDA

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 × 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.

Classifier performance vs Dimensionality (number of features), with Optimal number of features marked.
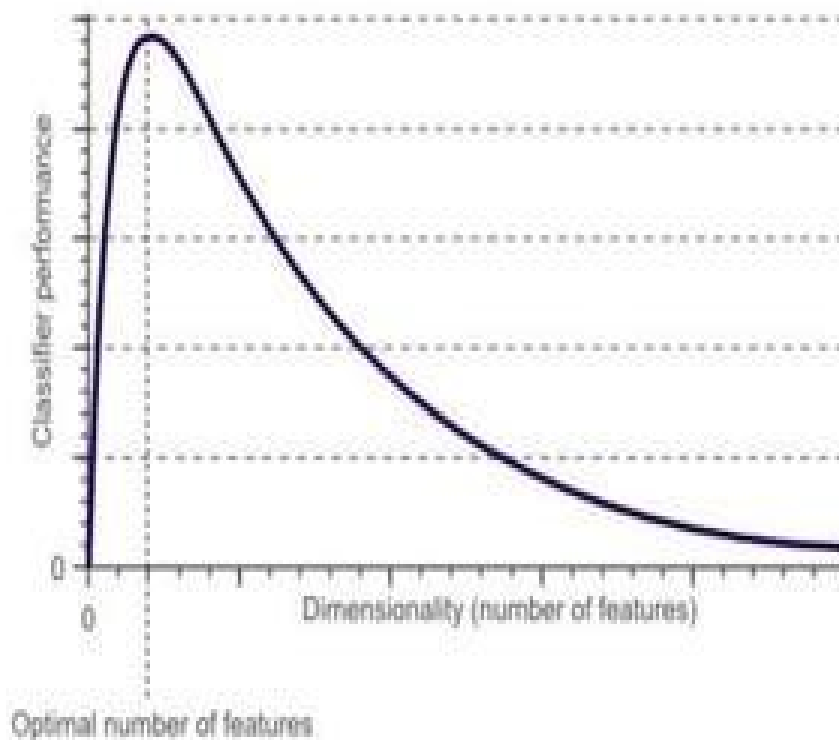
OVERFITTING

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

overfit model

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...)
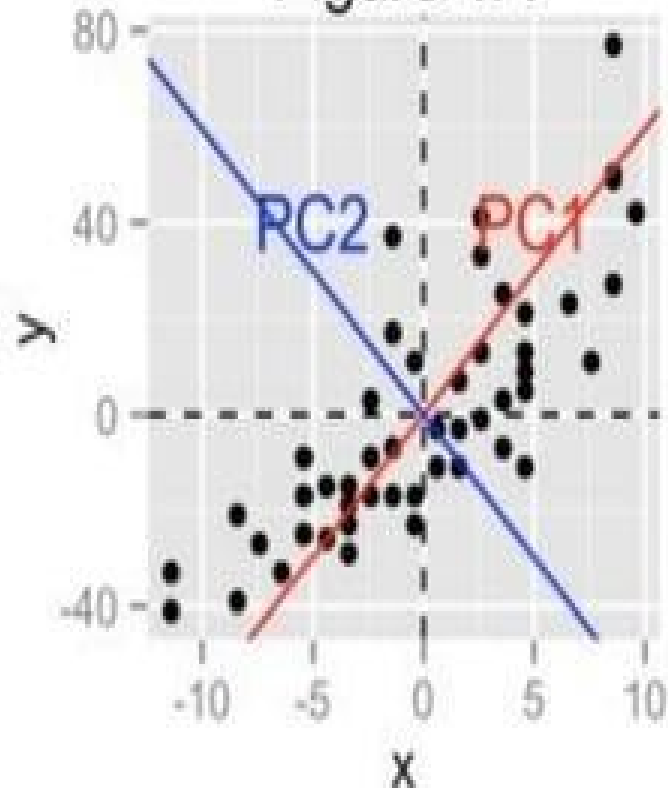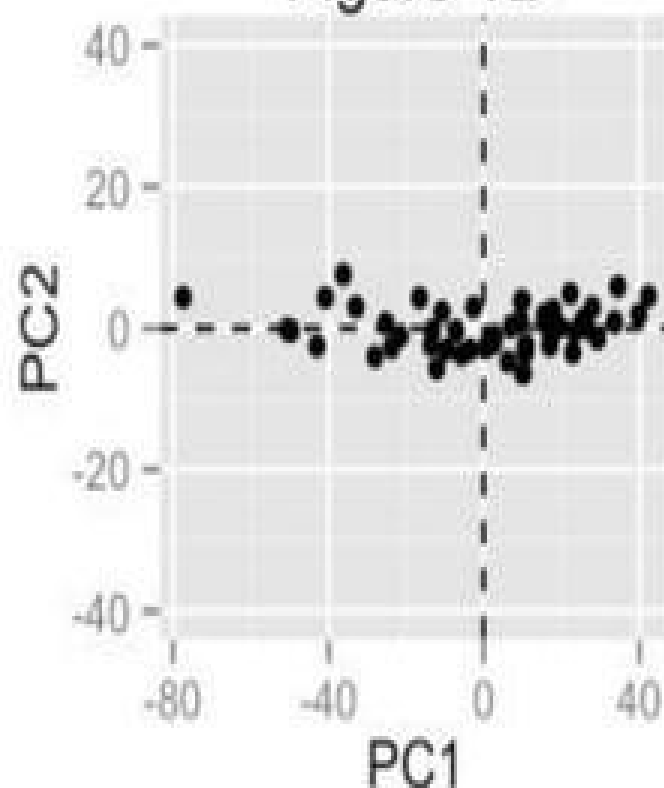

Figure 1A: The data are represented in the X-Y coordinate system

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.

Low redundancy

High redundancy
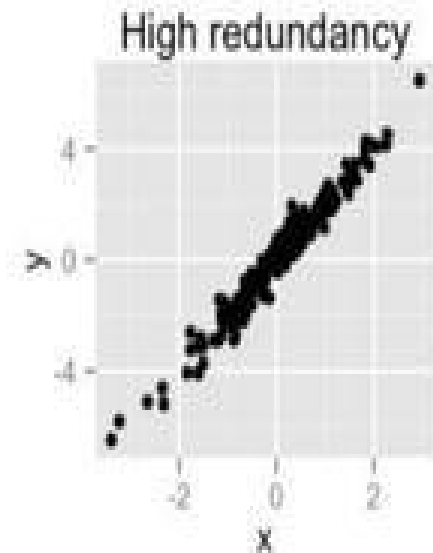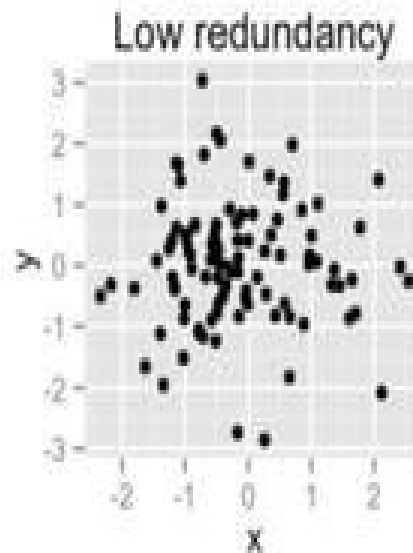
Steps to implement PCA in 2D dataset

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- μ_x

For all Y; y- = Y- μ_y

This produces a dataset whose mean is zero.

Step 2: Calculation of correlation

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

Covariance Matrix for Iris Dataset

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

Calculation of correlation(contd...)

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

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

m_x : mean of **x** variables

m_y : mean of **y** variables

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

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

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

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:

|λ - AI| = 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:

(λ - AI)v = 0

Step 4: Choosing Component

- Eigenvalues from largest to smallest so that it gives us the components in order or 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

$$NewData = FeatureVector^T \times ScaledData^T$$

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

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

Output

```
[[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 $x\,(1)\,,\,x\,(2)\,,\,\dots\,x\,(N)$, $N1$ of which belong to class $\omega1$, and $N2$ to class $\omega2$

  - 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
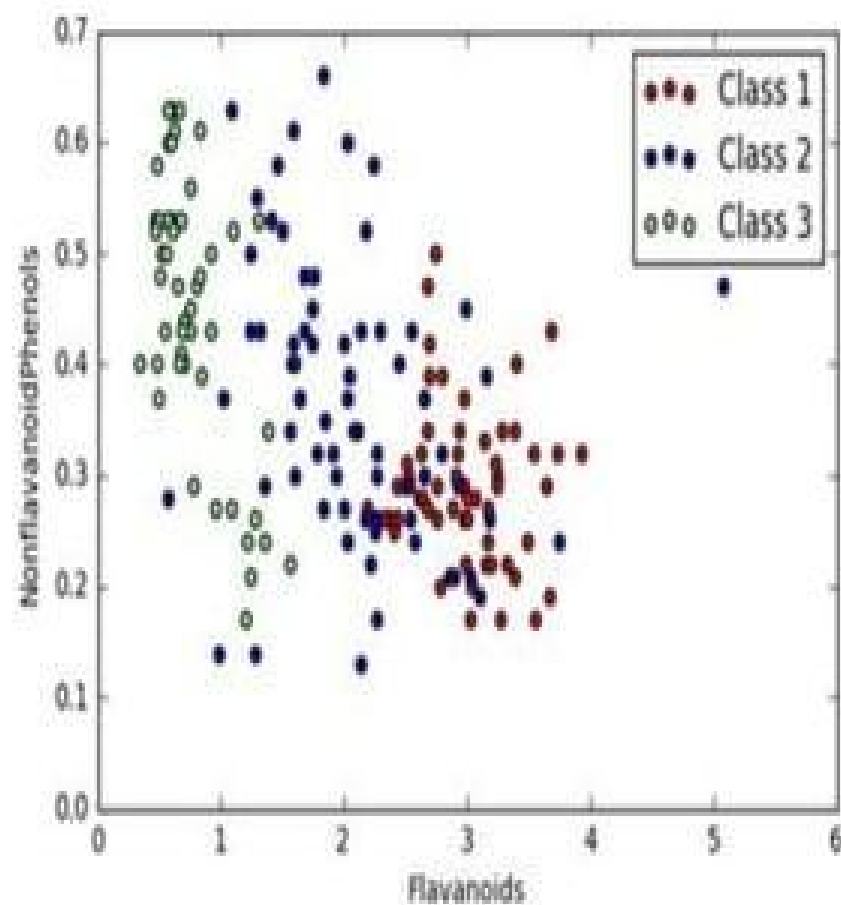
Linear Discriminant Analysis (Cont...)

- 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.
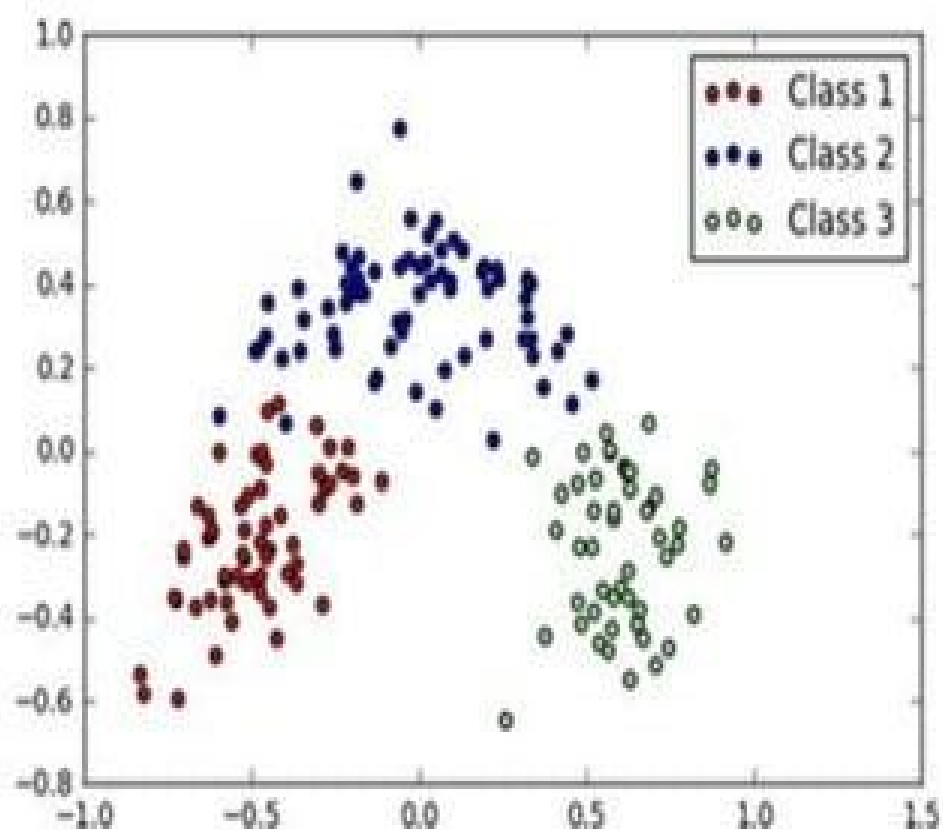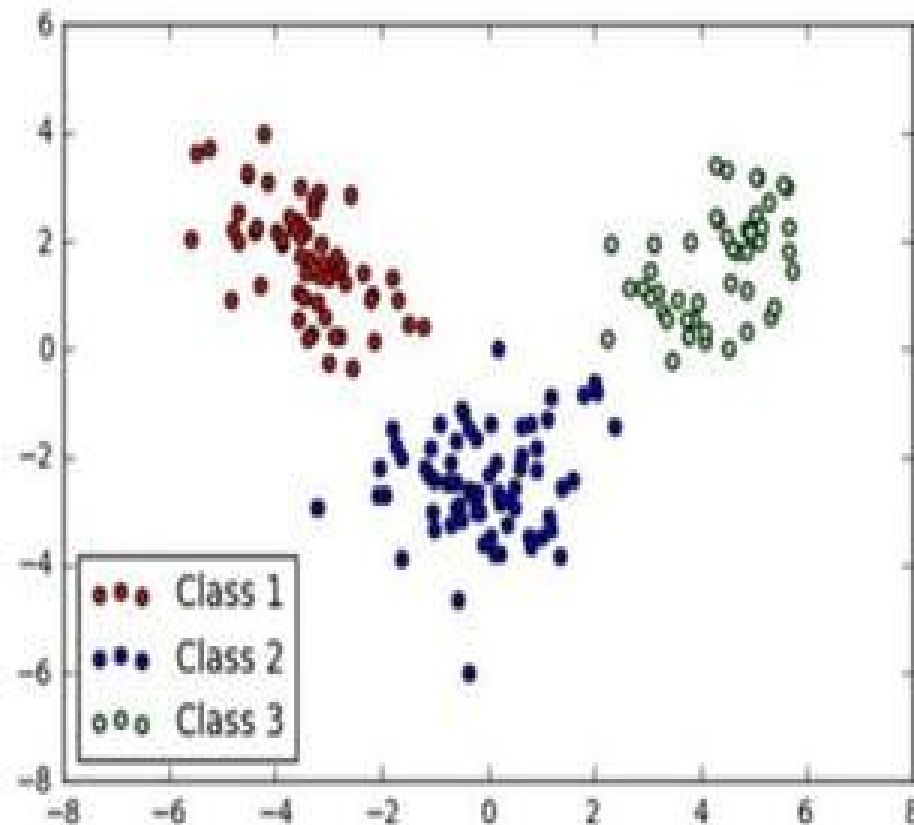 - Minimize the variation (which LDA calls scatter), within each category.
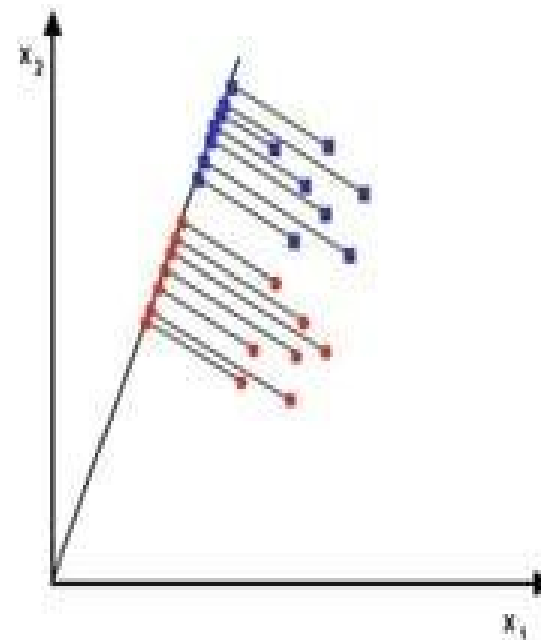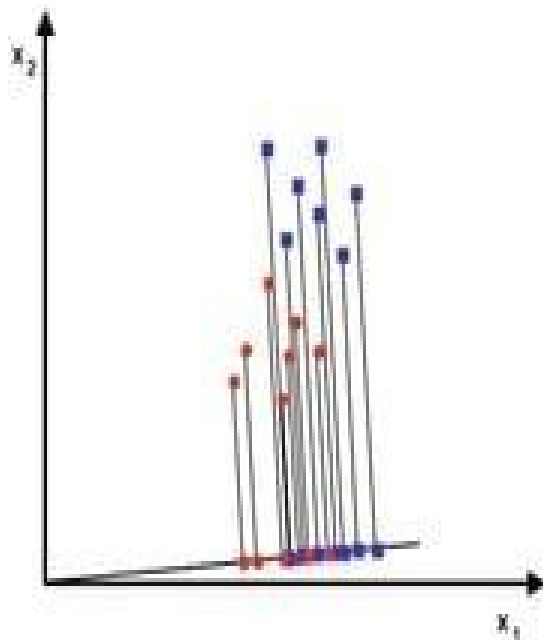
Figure 1.1 : Shows the plotting of data using two features

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:

$$Trset = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \cdots & \cdots \\ \cdots & \cdots \\ a_{m1} & a_{m2} \end{bmatrix} \qquad Tsset = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ \cdots & \cdots \\ \cdots & \cdots \\ b_{m1} & b_{m2} \end{bmatrix} \qquad (1)$$

Mathematical Operations(Cont...)

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

$$\mu_3 = p_1 \times \mu_1 + p_2 \times \mu_2 \tag{2}$$

where and 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

$$S_w = \sum_j p_j \times (cov_j) \tag{3}$$

Therefore, for the two-class problem,

$$S_w = 0.5 \times cov_1 + 0.5 \times cov_2 \tag{4}$$

Mathematical Operations(Cont...)

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

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

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

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

Mathematical Operations(Cont...)

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

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

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

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

- 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,

$$transformed\_set\_j = transform\_j^T \times set\_j \tag{9}$$

For the class independent LDA,

$$transformed\_set = transform\_spec^T \times data\_set^T \tag{10}$$

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]