A Study on Dimensionality Reduction Algorithms

Dr Basheer Mohamed
Computer Engineering, St Peter's Engineering College

Abstract **- Dimensionality Reduction.**

INTRODUCTION

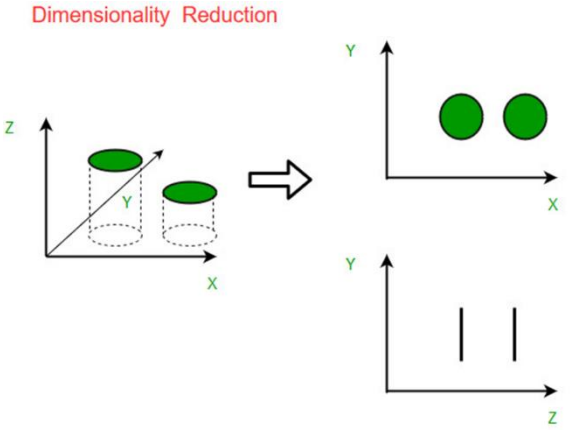
Machine Learning: Machine learning is nothing but a field of study which allows computers to "learn" like humans without any need of explicit programming.

Predictive Modeling: Predictive modeling is a probabilistic process that allows us to forecast outcomes, on the basis of some predictors. These predictors are basically features that come into play when deciding the final result, i.e. the outcome of the model.

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Importance of Dimensionality Reduction in Machine Learning and Predictive Modeling

An intuitive example of dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc. However, some of these features may overlap. In another condition, a classification problem that relies on both humidity and rainfall can be collapsed into just one underlying feature, since both of the aforementioned are correlated to a high degree. Hence, we can reduce the number of features in such problems. A 3-D classification problem can be hard to visualize, whereas a 2-D one can be mapped to a simple 2 dimensional space, and a 1-D problem to a simple line. The below figure illustrates this concept, where a 3-D feature space is split into two 1-D feature spaces, and later, if found to be correlated, the number of features can be reduced even further.



There are two components of dimensionality reduction:

- Feature selection: In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:

[1] Filter
[2] Wrapper
[3] Embedded

- Feature extraction: This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Methods of Dimensionality Reduction

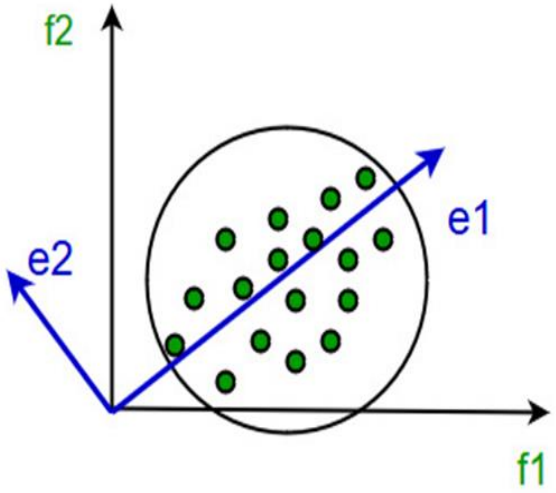
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Gaussian Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used. The prime

linear method, called Principal Component Analysis, or PCA, is discussed below.

Principal Component Analysis

This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.



It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues which are used to reconstruct a large fraction of variance of the original data.

Hence, It has been left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

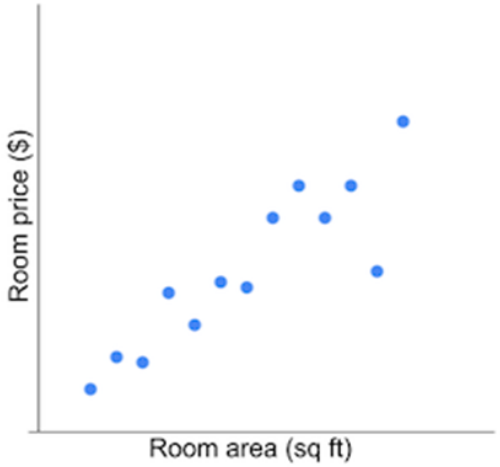
PCA 2D example

First, consider a dataset in only two dimensions, like (height, weight). This dataset can be plotted as points in a plane. But if it is requied to tease out variation, PCA finds a new coordinate system in which every point has a new (x,y) value. The axes don't actually mean anything physical; they're combinations of height and weight called "principal components" that are chosen to give one axes lots of variation.

Drag the points around in the following visualization to see PC coordinate system adjusts.

Index	Room Area (sq ft)	Room price ($)
1	210	75
2	250	110
3	300	125
4	320	170
.	.	.
20	340	175

The 2-dimensional feature space can be plot as shown below:



The goal is to reduce the 2 dimensions that are represented as x-axis and y-axis respectively, into one. Note that A 2D dataset has been chosen because it is easy to visualize. In a real world scenario, there may be thousands or more features and therefore there is a need for dimensionality reduction. There is no prediction going on here.

Mathematically, let X and Y be mxn matrices that are related by a transformation P. Matrix X is the original dataset with n features and Y is the new dataset with n *new features* or *principal components*.

PX=Y

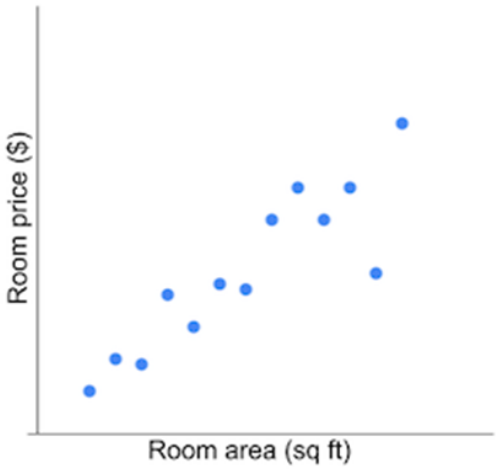
- P is a matrix that transforms X into Y.
- Columns of X are the *old basis* or original features.
- Rows of transformation matrix P are a set of *new basis vectors* for expressing the columns of X.
- Matrix Y is a re-representation of X, its columns being the new features or *principal components.*

Now that we have a mathematical representation of the goal, It should be known that what is the ideal choice for the transformation matrix P which depends on what are the properties required for the features of matrix Y to exhibit.

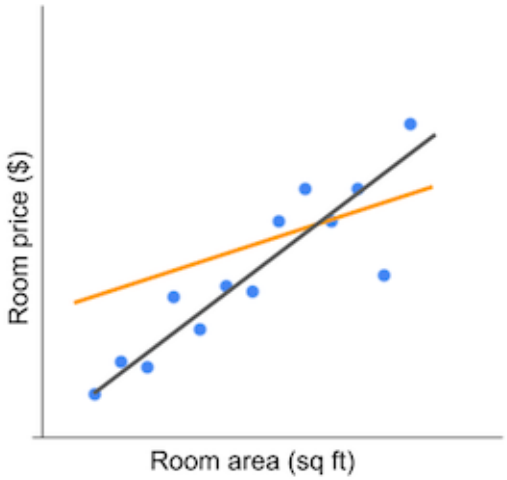
Variance and ***Redundancy***

Projection error and Variance

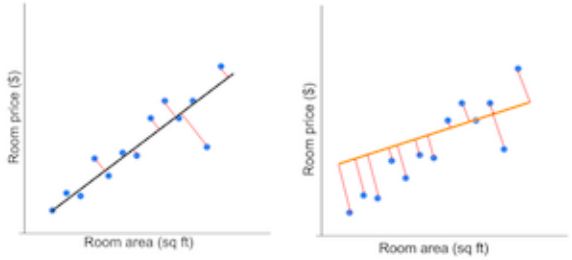
The 2D feature space of our example dataset is again shown below,



We want to re express this data in one dimension or compress the two features into one. Thus what is required, is draw a line passing through the data cloud and project each data point on the line. Of all the various possible lines that we can draw, It has been represented two lines in black and orange.



Which line best represents the data? The black line can be chosen as it is closer to most of the data points (see figure below). The point where the red tick intersects the black/orange line is the projection of corresponding blue data point. Here attention is paid to the projection distances or the size of red ticks that connect each data point to the black and orange lines respectively. The black line minimizes the cumulative projection distances of the data points. In other words, the black vector minimizes the projection error or information loss as it is moved from representing the data from 2D to 1D



Thus a desirable property of matrix Y is that the new feature or its first principal component should be along the line that minimizes projection error, while simultaneously maximizing variance of the projected data.

Redundancy

PCA exploits the inherent redundancy in a dataset to reduce dimensions. Consider the following plots of 2D feature spaces covering the possible spectrum of data redundancies.

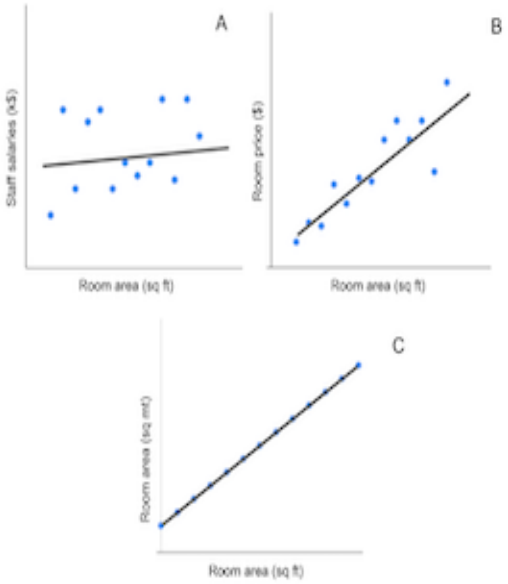
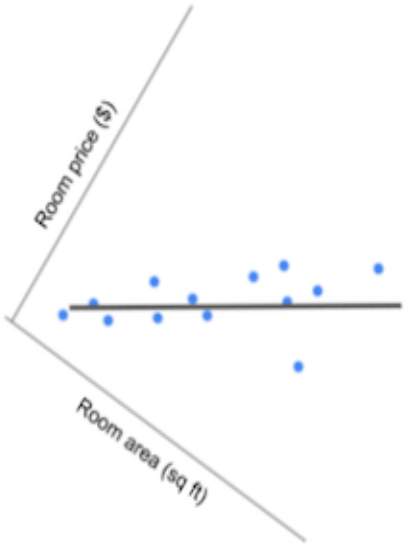


Figure A plots ***Staff salaries*** vs ***Room area (sq ft)***, that are uncorrelated with each other. The two dimensions in Figure A do not exhibit any redundancy and cannot be subject to dimension reduction by PCA. On the extreme side, Figure C is a plot of ***Room area in square metres*** vs ***Room area in square feet***. There is complete correlation between the two features, therefore in this scenario, it is safe to eliminate one of the two as they both are essentially giving the same information.

An ideal scenario where the role of PCA is appreciated is Figure B which is the same plot as our previous example, ***Room price ($)*** vs ***Room area (sq ft).*** Figure B shows some correlation between the two features, indicating that the data can be re expressed by a new feature that is a linear combination of the old features. Thus, when the basis is changed from 2D to 1D by projecting each data point onto the black line as shown previously, we are also eliminating feature redundancy. As observed in Figure B, the data points become uncorrelated precisely along the black line passing through the data cloud. The same is demonstrated in the reoriented figure below,



Variance is the spread of data for one variable, whereas *Covariance* is a measure of how two variables vary together. If we denote the features Room area (sq ft) and Room price ($) as variables x and y respectively,

$$\text{Variance of } x = \sigma^2 x$$

$$\text{Variance of } y = \sigma^2 y$$

$$\text{Covariance of } x,y = \sigma(x,y) = \sigma(y,x)$$

and the Covariance matrix can be computed as follows,

$$\begin{pmatrix} \sigma^2 x & \sigma(y,x) \\ \sigma(x,y) & \sigma^2 y \end{pmatrix}$$

In our example of Room area (sq ft) vs Room prices ($), once we change the basis and reduce dimensions from 2D to 1D; the features become uncorrelated to each other or in other words the covariance is 0. The covariance matrix is therefore a diagonal matrix.

$$\begin{pmatrix} \sigma^2 x & 0 \\ 0 & \sigma^2 y \end{pmatrix}$$

Summarizing, the properties that we want the features/ principal component of matrix Y to exhibit are:

- The principal component should be along the direction that maximizes the variance of projected data.
- Features of matrix Y should be uncorrelated with each other, *i.e.* its covariance matrix should be a diagonal matrix.

Let us revisit the mathematical representation of the goal for PCA that we derived earlier,

PX=Y

X is the original dataset with n numbers of features. P is a transformation matrix that is applied to matrix X. Matrix Y is the new dataset with n numbers of new features/principal components. We have established the properties of features in matrix Y. The goal is to reduce redundancy or more precisely the covariance matrix of matrix Y (let's call it S*y*) is diagonal. Therefore, in our equation PX = Y, the choice of matrix P should be such that S*y* is diagonalized.

We know that a symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors.
In theory, PCA assumes that all the basis vectors i.e. the rows of matrix P are orthonormal eigenvectors of covariance matrix of X. Applying a transformation P to X results in a matrix Y such that S*y* is diagonalized. Secondly, it assumes that the directions with largest variances are the most important or 'most principal'. The rows of matrix P are rank ordered in terms of its corresponding variances or eigenvalues in this case. By eliminating the rows in matrix P with low eigenvalues, the directions in which variance is low are ignored. This makes it possible to effectively reduce the number of dimensions without significant loss in information.
If A is symmetric, then A is orthogonally diagonalizable and has only real eigenvalues.

Advantages of Dimensionality Reduction

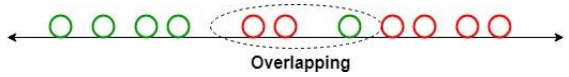
- It helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.

Disadvantages of Dimensionality Reduction

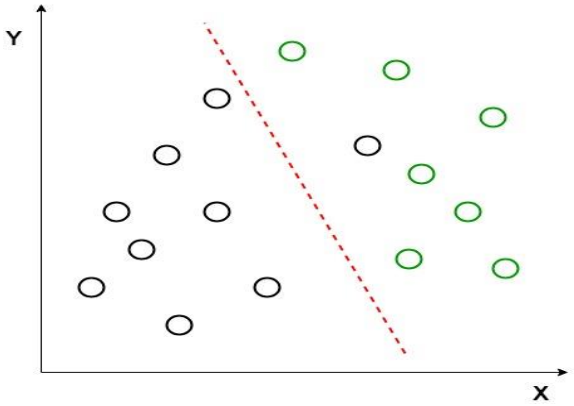
- It may lead to some amount of data loss.
- PCA tends to find linear correlations between variables, which is sometimes undesirable.
- PCA fails in cases where mean and covariance are not enough to define datasets.
- We may not know how many principal components to keep- in practice, some thumb rules are applied.

Linear Discriminant Analysis

Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space. For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, we will keep on increasing the number of features for proper classification.

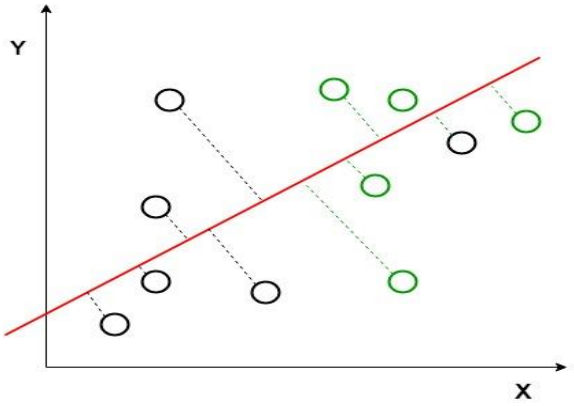


Example:
Suppose we have two sets of data points belonging to two different classes that we want to classify. As shown in the given 2D graph, when the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of the data points completely. Hence, in this case, LDA (Linear Discriminant Analysis) is used which reduces the 2D graph into a 1D graph in order to maximize the separability between the two classes.



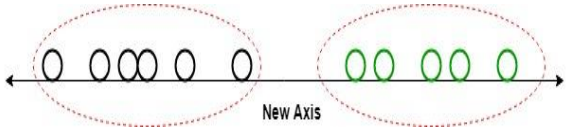
Here, Linear Discriminant Analysis uses both the axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximize the separation of the two categories and hence, reducing the 2D graph into a 1D graph.
Two criteria are used by LDA to create a new axis:

1. Maximize the distance between means of the two classes.
2. Minimize the variation within each class.



In the above graph, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each

class. In simple terms, this newly generated axis increases the separation between the data points of the two classes. After generating this new axis using the above-mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in the figure given below.



But Linear Discriminant Analysis fails when the mean of the distributions are shared, as it becomes impossible for LDA to find a new axis that makes both the classes linearly separable. In such cases, we use non-linear discriminant analysis.

Assumptions:

LDA makes some assumptions about the data:

- Assumes the data to be distributed normally or Gaussian distribution of data points i.e. each feature must make a bell-shaped curve when plotted.
- Each of the classes has identical covariance matrices.

However, it is worth mentioning that LDA performs quite well even if the assumptions are violated.

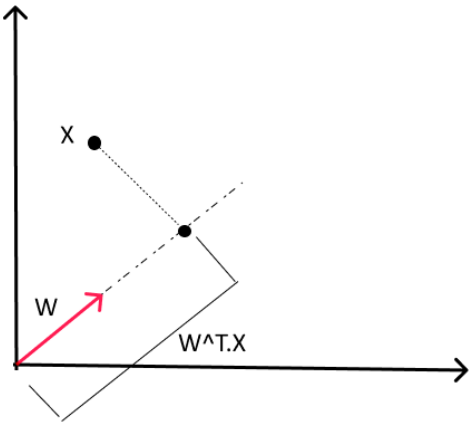
Fisher's Linear Discriminant:

LDA is a generalized form of FLD. Fisher in his paper used a discriminant function to classify between two plant species *Iris Setosa* and *Iris Versicolor*.

The basic idea of FLD is to project data points onto a line to maximize the between-class scatter and minimize the within-class scatter.

This might sound a bit cryptic but it is quite straightforward. So, before delving deep into the derivation part we need to get familiarized with certain terms and expressions.

- Let's suppose we have d-dimensional data points $x_1 \ldots x_n$ with 2 classes $C_{i=1,2}$ each having N_1 & N_2 samples.
- Let W be a unit vector onto which the data points are to be projected (took unit vector as we are only concerned with the direction).
- Number of samples : N = N1 + N2
- If x(n) are the samples on the feature space then WTx(n) denotes the data points after projection.
- Means of classes before projection: m_i
- Means of classes after projection: $M_i = W^T m_i$



Datapoint X before and after projection

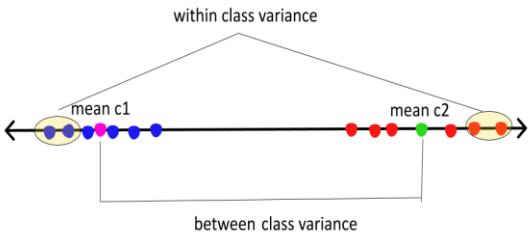
Scatter matrix: Used to make estimates of the covariance matrix. IT is a m X m positive semi-definite matrix.

Given by: sample variance * no. of samples.

Note: Scatter and variance measure the same thing but on different scales. So, we might use both words interchangeably. So, do not get confused.

Here we will be dealing with two types of scatter matrices

- Between class scatter = S_b = measures the distance between class means
- Within class scatter = S_w = measures the spread around means of each class



As per Fisher's LDA :

$\arg\max J(W) = (M_1 - M_2)^2 / S_1^2 + S_2^2$ …….(1)

The numerator here is between class scatter while the denominator is within-class scatter. So to maximize the function we need to maximize the numerator and minimize the denominator, simple math. To maximize the above function we need to first express the above equation in terms of W

Numerator:

$(M_1 - M_2)^2$

$= (W^T m_1 - W^T m_2)(W^T m_1 - W^T m_2)^T \quad = W^T (m_1 - m_2)(m_1 - m_2)^T W$

$= W^T S_b W$ ..... (2)

S_b = between class scatter

Denominator:

For denominator we have $S_1^2 + S_2^2$

Class Scatter before projection

$$S_i = \sum_{x(n) \in C_i} (x(n) - m_i)(x(n) - m_i)^T$$

Scatter for projected samples:

$$S_i^2 = \sum_{x(n) \in C_i} (W^T x(n) - M_i)(W^T x(n) - M_i)^T$$

$$M_i = W^T m_i$$

With little re-arrangement we will get:

$$S_i^2 = W^T \left(\sum_{x(n) \in C_i} (x(n) - m_i)(x(n) - m_i)^T \right) W$$

$$S_i^2 = W^T S_i W$$

So,

$$S_1^2 + S_2^2 = W^T (S_1 + S_2) W = W^T S_W W \quad \text{..... (3)}$$

S_W = within class scatter

Now, we have both the numerator and denominator expressed in terms of W

$J(W) = W^T S_b W / W^T S_w W$

Upon differentiating the above function w.r.t W and equating with 0, we get a generalized eigenvalue-eigenvector problem

$S_b W = v S_w W$

S_w being a full-rank matrix , inverse is feasible

$=> S_w^{-1} S_b W = vW$

Where v = eigen value

W = eigen vector

Gaussian Discriminant Analysis

There are two types of Supervised Learning algorithms used for classification in Machine Learning.

1. Discriminative Learning Algorithms

2. Generative Learning Algorithms

Discriminative Learning Algorithms include Logistic Regression, Perceptron Algorithm, etc. which try to find a decision boundary between different classes during the learning process. For example, given a classification problem to predict whether a patient has malaria or not a Discriminative Learning Algorithm will try to create a classification boundary to separate two types of patients, and when a new example is introduced it is checked on which side of the boundary the example lies to classify it. Such algorithms try to model P(y|X) i.e. given a feature set X for a data sample what is the probability it belongs to the class 'y'.

On the other hand, Generative Learning Algorithms follow a different approach, they try to capture the distribution of each class separately instead of finding a decision boundary among classes. Considering the previous example, a Generative Learning Algorithm will look at the distribution of infected patients and healthy patients separately and try to learn each of the distribution's features separately, when a new example is introduced it is compared to both the distributions, the class to which the data example resembles the most will be assigned to it. Such algorithms try to model P(X|y) for a given P(y) here, P(y) is known as a class prior.

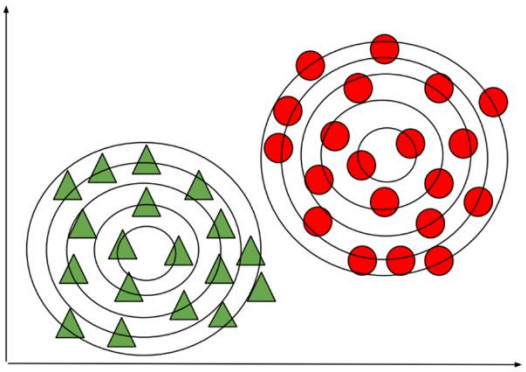
The predictions for generative learning algorithms are made using Bayes Theorem as follows:

$$P(y|X) = \frac{P(X|y).P(y)}{P(X)}$$

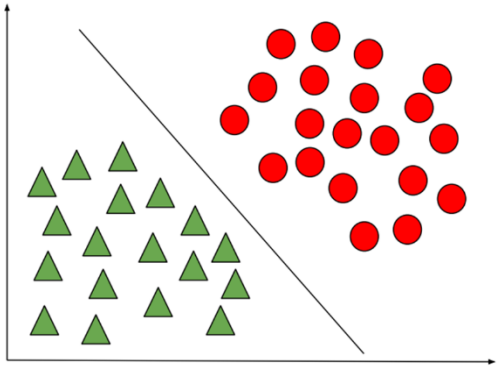
$$where, P(X) = P(X|y=1).P(y=1) + P(X|y=0).P(y=0)$$

Using only the values of P(X|y) and P(y) for the particular class we can calculate P(y|X) i.e given the features of a data sample what is the probability it belongs to the class 'y'.

Gaussian Discriminant Analysis is a Generative Learning Algorithm and in order to capture the distribution of each class, it tries to fit a Gaussian Distribution to every class of the data separately. The below images depict the difference between the Discriminative and Generative Learning Algorithms. The probability of a prediction in the case of the Generative learning algorithm will be high if it lies near the centre of the contour corresponding to its class and decreases as we move away from the centre of the contour



Generative Learning Algorithm (GDA)



Discriminative Learning Algorithm

Let's consider a binary classification problem such that all the data samples are IID (Independently and Identically distributed), therefore to calculate P(X|y) we can use Multivariate Gaussian Distribution to form a probability density function for each individual class. And to calculate P(y) or class prior for each class we can use Bernoulli distribution as all the data samples in binary classification can either take value 1 or 0.

Therefore, the probability distribution and class prior to a data sample can be defined using the general form of Gaussian and Bernoulli distribution respectively:

$$P(x|y=0) = \frac{1}{(2\pi)^{n/2} * |\Sigma|^{1/2}} exp(-1/2(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)) \quad - \textbf{Eq 1}$$

$$P(x|y=1) = \frac{1}{(2\pi)^{n/2} * |\Sigma|^{1/2}} exp(-1/2(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)) \quad - \textbf{Eq 2}$$

$$P(y) = \phi^y.(1-\phi)^{1-y} \quad - \textbf{Eq 3}$$

In order to view the probability distributions as a function of parameters mentioned above, we can define a Likelihood function which is equal to the product of probability distribution and class prior to each data sample (Taking product of the probabilities is reasonable as all the data samples are considered IID).

$$L(\phi, \mu_0, \mu_1, \Sigma) = \Pi_{i=1}^m P(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma)$$
$$= \Pi_{i=0}^m P(x^{(i)}|y^{(i)}).P(y^{(i)}) \quad - \textbf{Eq 4}$$

According to the principle of Maximum Likelihood estimation we have to choose the value of parameters in a way to maximize the probability function given in Eq 4. To do so instead of maximizing the Likelihood function we can maximize Log-Likelihood Function which is a strictly increasing function.

$$Therefore,\ Log - Likelihood\ function = log(L(\phi, \mu_0, \mu_1, \Sigma))$$

$$On\ maximizing\ Log - Likelihood\ following\ parameters\ are\ obtained$$

$$\phi = \frac{1}{m}\Sigma_{i=1}^m 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\Sigma_{i=1}^m \mathbb{1}\{y^{(i)} = 0\}.x^{(i)}}{\Sigma_{i=1}^m \mathbb{1}\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\Sigma_{i=1}^m \mathbb{1}\{y^{(i)} = 1\}.x^{(i)}}{\Sigma_{i=1}^m \mathbb{1}\{y^{(i)} = 1\}}$$

$$\Sigma = \frac{1}{m}\Sigma_{i=1}^m (x^{(i)} - \mu_{y^{(i)}}).(x^{(i)} - \mu_{y^{(i)}})^T$$

In the above equations, the function "1{condition}" is the indicator function which returns 1 if the condition is true else returns 0. For example, 1{y=1} will return 1 only when the class of that data sample is 1 else returns 0 similarly, in the case of 1{y=0} will return 1 only when the class of that data sample is 0 else it returns 0.

The values of the parameters obtained can be plugged in Eq 1, 2, and 3 to find the probability distribution and class prior to all the data samples. These values obtained can be further multiplied to find the Likelihood function given in Eq 4. As mentioned earlier the likelihood function i.e P(X|y). P(y) can be plugged into the Bayes formula to predict P(y|X) (i.e predict the class 'y' of a data sample for the given features 'X')

Therefore, Gaussian Discriminant Analysis works quite well for a small amount of data (say a few thousand examples) and can be more robust compared to Logistic Regression if our underlying assumptions about the distribution of the data are true

REFERENCE

[1] Anannya Uberoi, GeeksforGeeks.org
[2] A Tutorial on Principal Component Analysis by Jonathon Shlens on 3 Apr 2014