UNIT – IV

Dimensionality Reduction – Linear Discriminant Analysis – Principal Component Analysis – Factor Analysis – Independent Component Analysis – Locally Linear Embedding – Isomap – Least Squares Optimization

Evolutionary Learning – Genetic algorithms – Genetic Offspring: - Genetic Operators – Using Genetic Algorithms

Dimensionality reduction is the process of reducing the number of features (or dimensions) in a dataset while retaining as much information as possible. This can be done for a variety of reasons, such as to reduce the complexity of a model, to improve the performance of a learning algorithm, or to make it easier to visualize the data. There are several techniques for dimensionality reduction, including principal component analysis (PCA), singular value decomposition (SVD), and linear discriminant analysis (LDA). Each technique uses a different method to project the data onto a lower-dimensional space while preserving important information.

In machine learning,
high-dimensional data refers to data with a large number of features or variables. The curse of dimensionality is a common problem in machine learning, where the performance of the model deteriorates as the number of features increases. This is because the complexity of the model increases with the number of features, and it becomes more difficult to find a good solution.
In addition, high-dimensional data can also lead to overfitting, where the model fits the training data too closely and does not generalize well to new data.

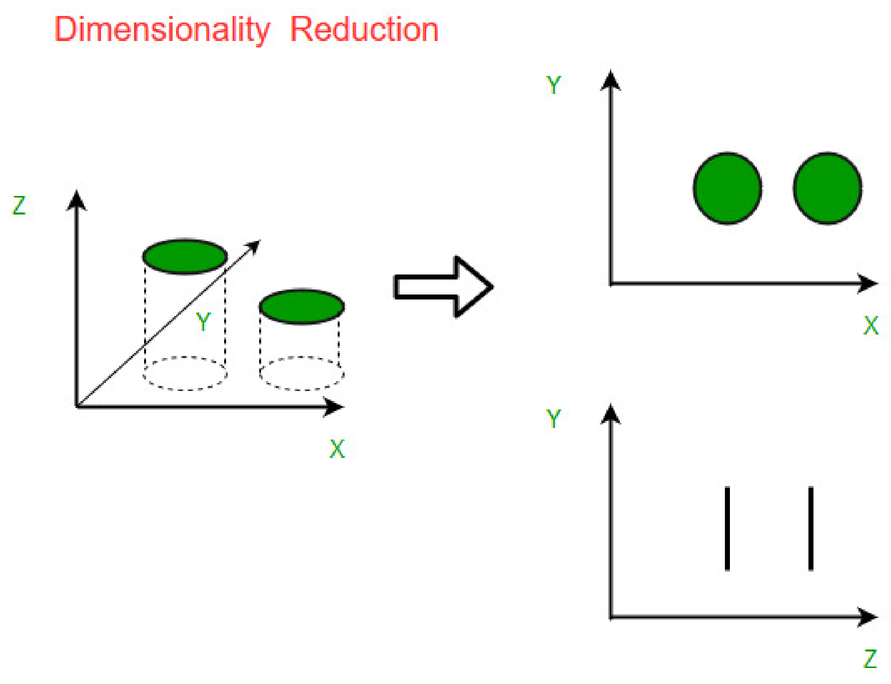
There are two main approaches to dimensionality reduction: feature selection and feature extraction.
Feature Selection:
Feature selection involves selecting a subset of the original features that are most relevant to the problem at hand. The goal is to reduce the dimensionality of the dataset while retaining the most important features. There are several methods for feature selection, including filter methods, wrapper methods, and embedded methods. Filter methods rank the features based on their relevance to the target variable, wrapper methods use the model performance as the criteria for selecting features, and embedded methods combine feature selection with the model training process.
There are two main approaches to dimensionality reduction: feature selection and feature extraction.

Feature Extraction:

Feature extraction involves creating new features by combining or transforming the original features. The goal is to create a set of features that captures the essence of the original data in a lower-dimensional space. There are several methods for feature extraction, including principal component analysis (PCA), linear discriminant analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE). PCA is a popular technique that projects the original features onto a lower-dimensional space while preserving as much of the variance as possible.

Why is Dimensionality Reduction important 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 2-D feature spaces, and later, if found to be correlated, the number of features can be reduced even further.



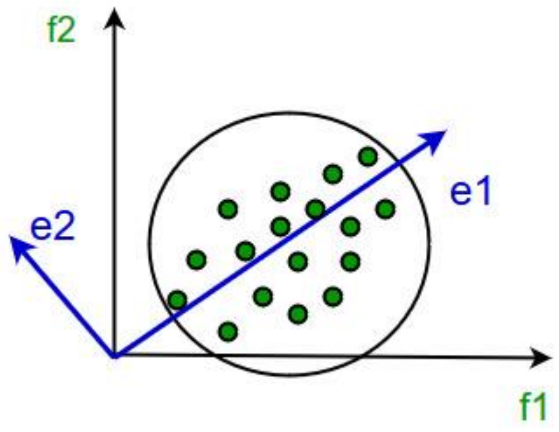
Methods of Dimensionality Reduction
The various methods used for dimensionality reduction include:

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

Dimensionality reduction may be both linear and non-linear, depending upon the method used. The prime linear method, called Principal Component Analysis, or PCA,

Principal Component Analysis
This method was introduced by Karl Pearson. It works on the 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 are used to reconstruct a large fraction

of variance of the original data.

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.
- Improved Visualization: High dimensional data is difficult to visualize, and dimensionality reduction techniques can help in visualizing the data in 2D or 3D, which can help in better understanding and analysis.
- Overfitting Prevention: High dimensional data may lead to overfitting in machine learning models, which can lead to poor generalization performance. Dimensionality reduction can help in reducing the complexity of the data, and hence prevent overfitting.
- Feature Extraction: Dimensionality reduction can help in extracting important features from high dimensional data, which can be useful in feature selection for machine learning models.
- Data Preprocessing: Dimensionality reduction can be used as a preprocessing step before applying machine learning algorithms to reduce the dimensionality of the data and hence improve the performance of the model.

- Improved Performance: Dimensionality reduction can help in improving the performance of machine learning models by reducing the complexity of the data, and hence reducing the noise and irrelevant information in the data.

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.
- Interpretability: The reduced dimensions may not be easily interpretable, and it may be difficult to understand the relationship between the original features and the reduced dimensions.
- Overfitting: In some cases, dimensionality reduction may lead to overfitting, especially when the number of components is chosen based on the training data.
- Sensitivity to outliers: Some dimensionality reduction techniques are sensitive to outliers, which can result in a biased representation of the data.
- Computational complexity: Some dimensionality reduction techniques, such as manifold learning, can be computationally intensive, especially when dealing with large datasets.

What is Factor Analysis?

Factor analysis, a method within the realm of statistics and part of the general linear model (GLM), serves to condense numerous variables into a smaller set of factors. By doing so, it captures the maximum shared variance among the variables and condenses them into a unified score, which can subsequently be utilized for further analysis.Factor analysis operates under several assumptions: linearity in relationships, absence of multicollinearity among variables, inclusion of relevant variables in the analysis, and genuine correlations between variables and factors. While multiple methods exist, principal component analysis stands out as the most prevalent approach in practice.

What does Factor mean in Factor Analysis?

In the context of factor analysis, a “factor” refers to an underlying, unobserved variable or latent construct that represents a common source of variation among a set of observed variables. These observed variables, also known as indicators or manifest variables, are the measurable variables that are directly observed or measured in a study.

How to do Factor Analysis (Factor Analysis Steps)?

Factor analysis is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors. Here are the general steps involved in conducting a factor analysis:

Factor analysis
Is a ML technique
Condense data
from
Larger number of features
TO
Smaller number of factors

1. Determine the Suitability of Data for Factor Analysis

- **Bartlett’s Test:** Check the significance level to determine if the correlation matrix is suitable for factor analysis.
- **Kaiser-Meyer-Olkin (KMO) Measure:** Verify the sampling adequacy. A value greater than 0.6 is generally considered acceptable.

2. Choose the Extraction Method

- **Principal Component Analysis (PCA):** Used when the main goal is data reduction.
- **Principal Axis Factoring (PAF):** Used when the main goal is to identify underlying factors.

3. Factor Extraction

- Use the chosen extraction method to identify the initial factors.

- Extract eigenvalues to determine the number of factors to retain. Factors with eigenvalues greater than 1 are typically retained in the analysis.
- Compute the initial factor loadings.

4. Determine the Number of Factors to Retain

- **Scree Plot:** Plot the eigenvalues in descending order to visualize the point where the plot levels off (the "elbow") to determine the number of factors to retain.
- **Eigenvalues:** Retain factors with eigenvalues greater than 1.

5. Factor Rotation

- **Orthogonal Rotation (Varimax, Quartimax):** Assumes that the factors are uncorrelated.
- **Oblique Rotation (Promax, Oblimin):** Allows the factors to be correlated.
- Rotate the factors to achieve a simpler and more interpretable factor structure.
- Examine the rotated factor loadings.

6. Interpret and Label the Factors

- Analyze the rotated factor loadings to interpret the underlying meaning of each factor.
- Assign meaningful labels to each factor based on the variables with high loadings on that factor.

7. Compute Factor Scores (if needed)

- Calculate the factor scores for each individual to represent their value on each factor.

8. Report and Validate the Results

- Report the final factor structure, including factor loadings and communalities.
- Validate the results using additional data or by conducting a confirmatory factor analysis if necessary.

Types of Factor Analysis

There are two main types of Factor Analysis used in data science:

1. Exploratory Factor Analysis (EFA)

Exploratory Factor Analysis (EFA) is used to uncover the underlying structure of a set of observed variables without imposing preconceived notions about how many factors there are or how the variables are related to each factor. It explores complex interrelationships among items and aims to group items that are part of unified concepts or constructs.

- Researchers do not make a priori assumptions about the relationships among factors, allowing the data to reveal the structure organically.
- Exploratory Factor Analysis (EFA) helps in identifying the number of factors needed to account for the variance in the observed variables and understanding the relationships between variables and factors.

2. Confirmatory Factor Analysis (CFA)

Confirmatory Factor Analysis (CFA) is a more structured approach that tests specific hypotheses about the relationships between observed variables and latent factors based on prior theoretical knowledge or expectations. It uses structural equation modeling techniques to test a measurement model, wherein the observed variables are assumed to load onto specific factors.

- Confirmatory Factor Analysis (CFA) assesses the fit of the hypothesized model to the actual data, examining how well the observed variables align with the proposed factor structure.
- This method allows for the evaluation of relationships between observed variables and unobserved factors, and it can accommodate measurement error.
- Researchers hypothesize the relationships between variables and factors before conducting the analysis, and the model is tested against empirical data to determine its validity.

In summary, while Exploratory Factor Analysis (EFA) is more exploratory and flexible, allowing the data to dictate the factor structure, Confirmatory Factor Analysis (CFA) is more confirmatory, testing specific hypotheses about how the observed variables are related to latent factors. Both methods are valuable tools in understanding the underlying structure of data and have their respective strengths and applications.

Types of Factor Extraction Methods

Some of the Type of Factor Extraction methods are dicussed below:

1. **Principal Component Analysis (PCA)**:
    - PCA is a widely used method for factor extraction.

    - It aims to extract factors that account for the maximum possible variance in the observed variables.
    - Factor weights are computed to extract successive factors until no further meaningful variance can be extracted.
    - After extraction, the factor model is often rotated for further analysis to enhance interpretability.
2. **Canonical Factor Analysis**:
    - Also known as Rao's canonical factoring, this method computes a similar model to PCA but uses the principal axis method.
    - It seeks factors that have the highest canonical correlation with the observed variables.
    - Canonical factor analysis is not affected by arbitrary rescaling of the data, making it robust to certain data transformations.
3. **Common Factor Analysis**:
    - Also referred to as Principal Factor Analysis (PFA) or Principal Axis Factoring (PAF).
    - This method aims to identify the fewest factors necessary to account for the common variance (correlation) among a set of variables.
    - Unlike PCA, common factor analysis focuses on capturing shared variance rather than overall variance.

Assumptions of Factor Analysis

Let's have a closer look onto the assumptions of factorial analysis, that are as follows:

1. **Linearity**: The relationships between variables and factors are assumed to be linear.
2. **Multivariate Normality**: The variables in the dataset should follow a multivariate normal distribution.
3. **No Multicollinearity**: Variables should not be highly correlated with each other, as high multicollinearity can affect the stability and reliability of the factor analysis results.

4. **Adequate Sample Size**: Factor analysis generally requires a sufficient sample size to produce reliable results. The adequacy of the sample size can depend on factors such as the complexity of the model and the ratio of variables to cases.
5. **Homoscedasticity**: The variance of the variables should be roughly equal across different levels of the factors.
6. **Uniqueness**: Each variable should have unique variance that is not explained by the factors. This assumption is particularly important in common factor analysis.
7. **Independent Observations**: The observations in the dataset should be independent of each other.
8. **Linearity of Factor Scores**: The relationship between the observed variables and the latent factors is assumed to be linear, even though the observed variables may not be linearly related to each other.
9. **Interval or Ratio Scale**: Factor analysis typically assumes that the variables are measured on interval or ratio scales, as opposed to nominal or ordinal scales.

Violation of these assumptions can lead to biased parameter estimates and inaccurate interpretations of the results. Therefore, it’s important to assess the data for these assumptions before conducting factor analysis and to consider potential remedies or alternative methods if the assumptions are not met.

What is Independent Component Analysis?

Independent Component Analysis (ICA) is a statistical and computational technique used in machine learning to separate a multivariate signal into its independent non-Gaussian components. The goal of ICA is to find a linear transformation of the data such that the transformed data is as close to being statistically independent as possible.
The heart of ICA lies in the principle of statistical independence. ICA identify components within mixed signals that are statistically independent of each other.

Statistical Independence Concept:
It is a probability theory that if two random variables X and Y are statistically independent. The joint probability distribution of the pair is equal to the product of their individual probability distributions, which means that knowing the outcome of one variable does not change the probability of the other outcome.

or

Assumptions in ICA

1. The first assumption asserts that the source signals (original signals) are statistically independent of each other.
2. The second assumption is that each source signal exhibits non-Gaussian distributions.

Mathematical Representation of Independent Component Analysis

The observed random vector is , representing the observed data with m components. The hidden components are

represented by the random vector , where n is the number of hidden sources.

Linear Static Transformation

The observed data X is transformed into hidden components S using a linear static transformation representation by the matrix W.

Here, W = transformation matrix.
The goal is to transform the observed data x in a way that the resulting hidden components are independent. The independence is measured by

some function . The task is to find the optimal transformation matrix W that maximizes the independence of the hidden components.

Advantages of Independent Component Analysis (ICA):

- ICA is a powerful tool for **separating mixed signals** into their independent components. This is useful in a variety of applications, such as signal processing, image analysis, and data compression.

- ICA is a **non-parametric approach**, which means that it does not require assumptions about the underlying probability distribution of the data.
- ICA is an **unsupervised learning technique**, which means that it can be applied to data without the need for labeled examples. This makes it useful in situations where labeled data is not available.
- ICA can be **used for feature extraction**, which means that it can identify important features in the data that can be used for other tasks, such as classification.

Disadvantages of Independent Component Analysis (ICA):

- ICA assumes that the underlying sources are non-Gaussian, which may not always be true. If the underlying sources are Gaussian, ICA may not be effective.
- ICA assumes that the sources are mixed linearly, which may not always be the case. If the sources are mixed nonlinearly, ICA may not be effective.
- ICA can be computationally expensive, especially for large datasets. This can make it difficult to apply ICA to real-world problems.
- ICA can suffer from convergence issues, which means that it may not always be able to find a solution. This can be a problem for complex datasets with many sources.

Cocktail Party Problem

Consider *Cocktail Party Problem* or *Blind Source Separation* problem to understand the problem which is solved by independent component analysis.

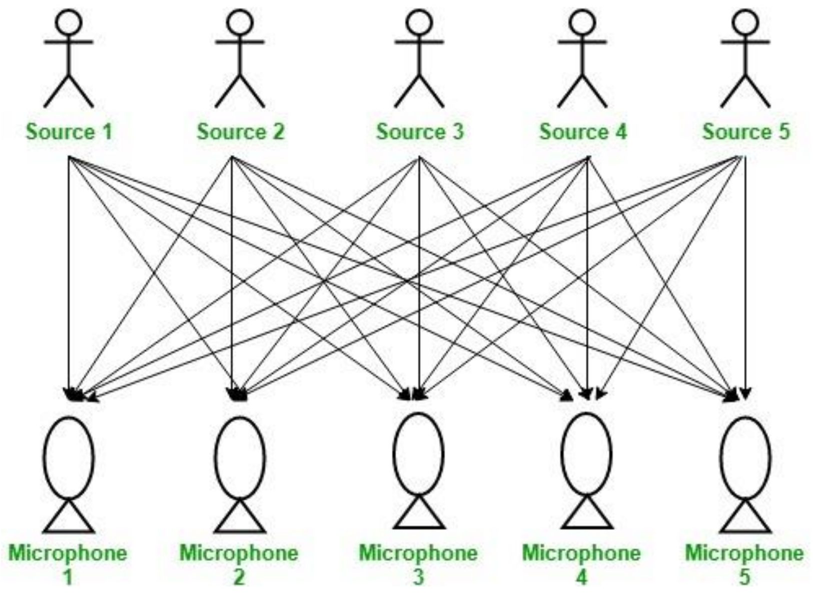
Problem: To extract independent sources' signals from a mixed signal composed of the signals from those sources.

Given: Mixed signal from five different independent sources.

Aim: To decompose the mixed signal into independent sources:

- Source 1
- Source 2
- Source 3
- Source 4
- Source 5

Solution: Independent Component Analysis



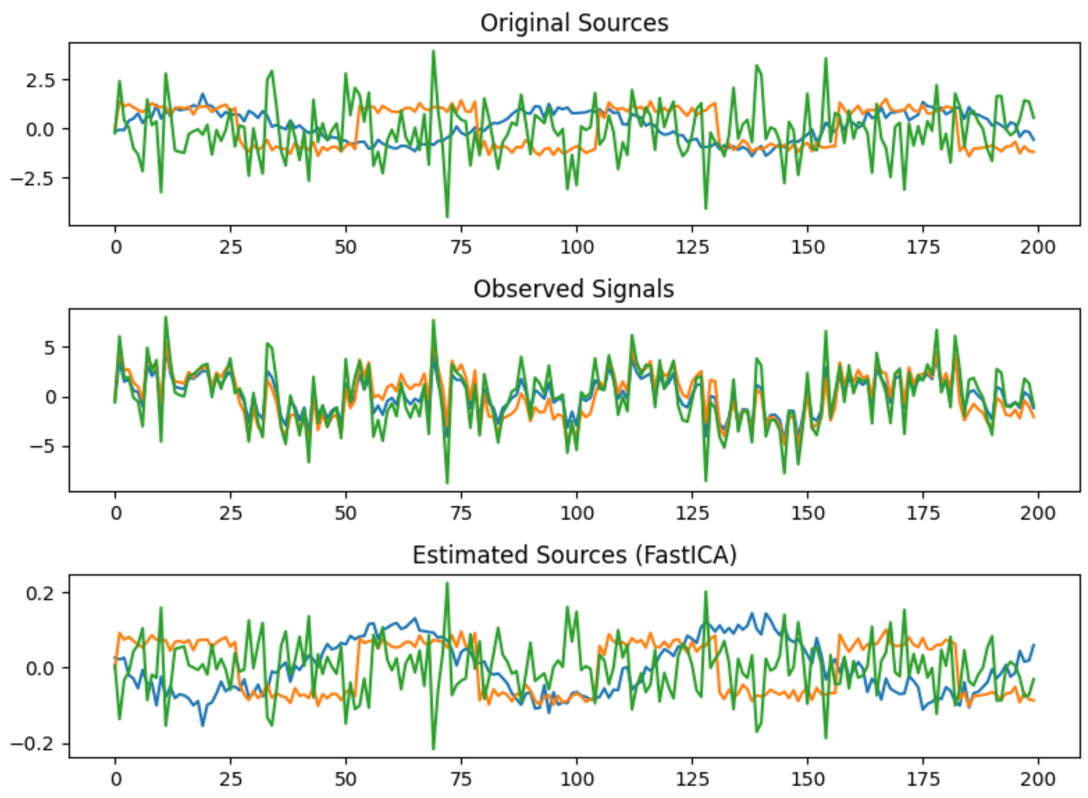
Here, there is a party going into a room full of people. There is ‘n’ number of speakers in that room, and they are speaking simultaneously at the party. In the same room, there are also ‘n’ microphones placed at different distances from the speakers, which are recording ‘n’ speakers’ voice signals. Hence, the number of speakers is equal to the number of microphones in the room. **Step 4: Visualize the signals**

- Python3

```
# Plot the results
plt.figure(figsize=(8, 6))
 plt.subplot(3, 1, 1)
plt.title('Original Sources')
plt.plot(S)
plt.subplot(3, 1, 2)
plt.title('Observed Signals')
plt.plot(X)
plt.subplot(3, 1, 3)
plt.title('Estimated Sources (FastICA)')
plt.plot(S_)
 plt.tight_layout()
plt.show()
```

Now, using these microphones' recordings, we want to separate all the 'n' speakers' voice signals in the room, given that each microphone recorded the voice signals coming from each speaker of different intensity due to the difference in distances between them. Decomposing the mixed signal of each microphone's recording into an independent source's speech signal can be done by using the machine learning technique, independent component analysis.

where, X1, X2, ..., Xn are the original signals present in the mixed signal and Y1, Y2, ..., Yn are the new features and are independent components that are independent of each other.



LLE**(Locally Linear Embedding)** is an unsupervised approach designed to transform data from its original high-dimensional space into a lower-dimensional representation, all while striving to retain the essential geometric characteristics of the underlying non-linear feature structure. LLE operates in several key steps:

- Firstly, it constructs a nearest neighbors graph to capture these local relationships. Then, it optimizes weight values for each data point, aiming to minimize the reconstruction error when expressing a point as a linear combination of its neighbors. This weight matrix reflects the strength of connections between points.
- Next, LLE computes a lower dimensional representation of the data by finding eigenvectors of a matrix derived from the weight matrix. These eigenvectors represent the most relevant directions in the reduced space. Users can specify the desired dimensionality for the output space, and LLE selects the top eigenvectors accordingly.

Mathematical Implementation of LLE Algorithm

The key idea of LLE is that locally, in the vicinity of each data point, the data lies approximately on a linear subspace. LLE attempts to unfold or unroll the data while preserving these local linear relationships.

Here is a mathematical overview of the LLE algorithm:

Minimize: $\sum i|xi \quad - \sum j \; Wijxj| \quad ^2$

Subject to : $\sum jwij = 1$

Where:

- x_i represents the i-th data point.
- w_{ij} are the weights that minimize the reconstruction error for data point x_i using its neighbors.

It aims to find a lower-dimensional representation of data while preserving local relationships. The mathematical expression for LLE involves minimizing the reconstruction error of each data point by expressing it as a weighted sum of its k nearest neighbors' contributions. This optimization is subject to constraints ensuring that the weights sum to 1 for each data point. Locally Linear Embedding (LLE) is a dimensionality reduction technique used in machine learning and data analysis. It focuses on preserving local relationships between data points when mapping high-dimensional data to a lower-dimensional space. Here, we will explain the LLE algorithm and its parameters.

Mathematical Implementation of LLE Algorithm

The key idea of LLE is that locally, in the vicinity of each data point, the data lies approximately on a linear subspace. LLE attempts to unfold or unroll the data while preserving these local linear relationships.
Here is a mathematical overview of the LLE algorithm:

Minimize:

Subject to :
Where:

- x_i represents the i-th data point.
- w_{ij} are the weights that minimize the reconstruction error for data point x_i using its neighbors.

Locally Linear Embedding Algorithm

The LLE algorithm can be broken down into several steps:

- **Neighborhood Selection:** For each data point in the high-dimensional space, LLE identifies its k-nearest neighbors. This step is crucial because LLE assumes that each data point can be well approximated by a linear combination of its neighbors.
- **Weight Matrix Construction:** LLE computes a set of weights for each data point to express it as a linear combination of its neighbors. These weights are determined in such a way that the reconstruction error is minimized. Linear regression is often used to find these weights.
- **Global Structure Preservation:** After constructing the weight matrix, LLE aims to find a lower-dimensional representation of the data that best preserves the local linear relationships. It does this by seeking a set of coordinates in the lower-dimensional space for each data point that minimizes a cost function. This cost function evaluates how well each data point can be represented by its neighbors.
- **Output Embedding:** Once the optimization process is complete, LLE provides the final lower-dimensional representation of the data. This representation captures the essential structure of the data while reducing its dimensionality.

Parameters in LLE Algorithm

LLE has a few parameters that influence its behavior:

- **k (Number of Neighbors):** This parameter determines how many nearest neighbors are considered when constructing the weight matrix. A larger k captures more global relationships but may introduce noise. A smaller k focuses on local relationships but can be sensitive to outliers. Selecting an appropriate value for k is essential for the algorithm's success.
- **Dimensionality of Output Space:** You can specify the dimensionality of the lower-dimensional space to which the data will be mapped. This is often chosen based on the problem's requirements and the trade-off between computational complexity and information preservation.
- **Distance Metric:** LLE relies on a distance metric to define the proximity between data points. Common choices include Euclidean distance, Manhattan distance, or custom-defined distance functions. The choice of distance metric can impact the results.
- **Regularization (Optional):** In some cases, regularization terms are added to the cost function to prevent overfitting. Regularization can be useful when dealing with noisy data or when the number of neighbors is high.
- **Optimization Algorithm (Optional):** LLE often uses optimization techniques like Singular Value Decomposition (SVD) or eigenvector methods to find the lower-dimensional representation. These optimization methods may have their own parameters that can be adjusted.

Advantages of LLE

The dimensionality reduction method known as locally linear embedding (LLE) has many benefits for data processing and visualization. The following are LLE's main benefits:

- **Preservation of Local Structures**: LLE is excellent at maintaining the in-data local relationships or structures. It successfully captures the inherent geometry of nonlinear manifolds by maintaining pairwise distances between nearby data points.

- **Handling Non-Linearity**: LLE has the ability to capture nonlinear patterns and structures in the data, in contrast to linear techniques like Principal Component Analysis (PCA). When working with complicated, curved, or twisted datasets, it is especially helpful.
- **Dimensionality Reduction**: LLE lowers the dimensionality of the data while preserving its fundamental properties. Particularly when working with high-dimensional datasets, this reduction makes data presentation, exploration, and analysis simpler.

Disavantages of LLE

- **Curse of Dimensionality**: LLE can experience the "curse of dimensionality" when used with extremely high-dimensional data, just like many other dimensionality reduction approaches. The number of neighbors required to capture local interactions rises as dimensionality does, potentially increasing the computational cost of the approach.
- **Memory and computational Requirements**: For big datasets, creating a weighted adjacency matrix as part of LLE might be memory-intensive. The eigenvalue decomposition stage can also be computationally taxing for big datasets.
- **Outliers and Noisy data**: LLE is susceptible to anomalies and jittery data points. The quality of the embedding may be affected and the local linear relationships may be distorted by outliers.

Implementation of Locally Linear Embedding

```
#importing Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_swiss_roll
from sklearn.manifold import LocallyLinearEmbedding
```

# Generating a Synthetic Dataset (Swiss Roll)

```
# Code for Generating a synthetic dataset (Swiss Roll)

n_samples = 1000

# Define the number of neighbors for LLE
```

n_neighbors = 10

X, _ = make_swiss_roll(n_samples=n_samples)

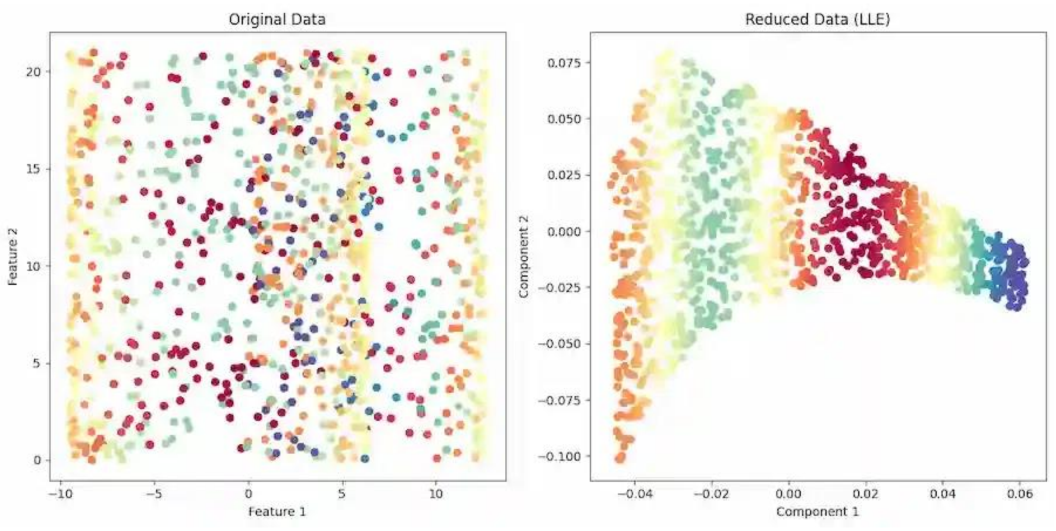
#It generates a synthetic dataset resembling a Swiss Roll using the make_swiss_roll function from scikit-learn.
n_samples specifies the number of data points to generate.
n_neighbors defines the number of neighbors used in the LLE algorithm.
#Applying Locally Linear Embedding (LLE)

```
# Including Locally Linear Embedding

lle = LocallyLinearEmbedding(n_neighbors=n_neighbors, n_components=2)

X_reduced = lle.fit_transform(X)
```

An instance of the LLE algorithm is created with LocallyLinearEmbedding. The n_neighbors parameter determines the number of neighbors to consider during the embedding process.
The LLE algorithm is then fitted to the original data X using the fit_transform method. This step reduces the dataset to two dimensions (n_components=2).

#Visualizing the Original and Reduced Data

```
# Code for Visualizing the original Versus reduced data

plt.figure(figsize=(12, 6))


plt.subplot(121)

plt.scatter(X[:, 0], X[:, 1], c=X[:, 2], cmap=plt.cm.Spectral)

plt.title("Original Data")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")


plt.subplot(122)

plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=X[:, 2], cmap=plt.cm.Spectral)

plt.title("Reduced Data (LLE)")
plt.xlabel("Component 1")
plt.ylabel("Component 2")

plt.tight_layout()
plt.show()
```



ISOMAP

A nonlinear dimensionality reduction method used in data analysis and machine learning is called isomap, short for isometric mapping. Isomap was developed to maintain the inherent geometry of high-dimensional data as a substitute for conventional techniques like Principal Component Analysis (PCA). Isomap creates a low-dimensional representation, usually a two- or three-dimensional map, by focusing on the preservation of pairwise distances between data points.
This technique works especially well for extracting the underlying structure from large, complex datasets, like those from speech recognition, image analysis, and biological systems. Finding patterns and insights in a variety of scientific and engineering domains is made possible by Isomap's capacity to highlight the fundamental relationships found in data.

Isomap

An understanding and representation of complicated data structures are crucial for the field of machine learning. To achieve this, Manifold Learning, a subset of unsupervised learning, has a significant role to play. Among the manifold learning techniques, ISOMAP (Isometric Mapping) stands out for its prowess in capturing the intrinsic geometry of high-dimensional data. In the case of situations in which linear methods are lacking, they have proved particularly efficient.

ISOMAP is a flexible tool that seamlessly blends multiple learning and dimensionality reduction intending to obtain more detailed knowledge of the underlying structure of data. This article takes a look at ISOMAP's inner workings and sheds light on its parameters, functions, and proper implementation with SkLearn.

Isometric mapping is an approach to reduce the dimensionality of machine learning.

Relation between Geodesic Distances and Euclidean Distances

Understanding the distinction between equatorial and elliptic distances is of vital importance for ISOMAP. The geodesic distance considers the shortest path along the curved surface of the manifold, as opposed to Euclidean distances which are measured by measuring straight Line distances in the input space. In order to provide a more precise representation of the data's internal structure, ISOMAP exploits these quantum distances.

ISOMAP Parameters

ISOMAP comes with several parameters, each influencing the dimensionality reduction process:

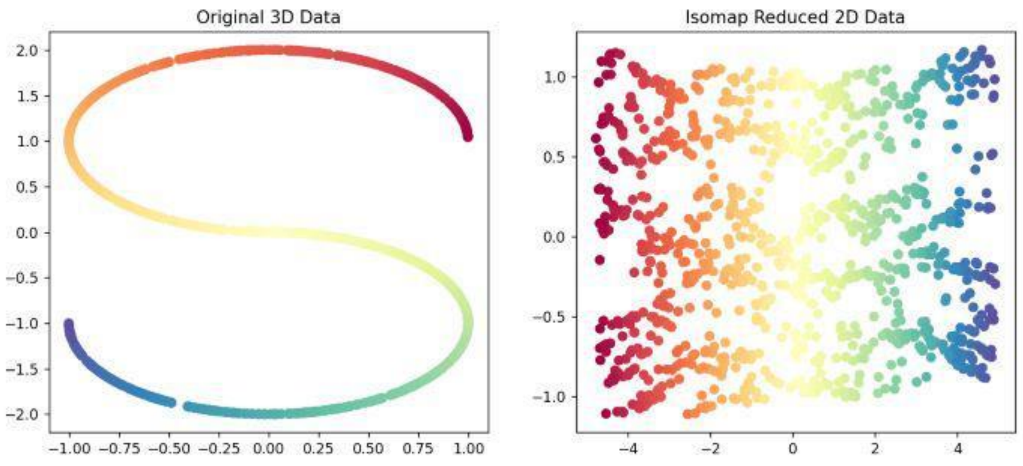
- **n_neighbors**: Determines the number of neighbors used to approximate geodesic distances. Higher values may make it possible to achieve higher results, but they still require more computing power.
- **n_components**: Determines the number of dimensions in a low dimensional representation.
- **eigen_solve**r: Determines the method used for decomposing an Eigenvalue. There are options such as "auto", "arpack" and "dense."

- **radius**: You can designate a radius within which neighbors are taken into account in place of using a set number of neighbors. Outside of this range, data points are not regarded as neighbors.
- **tol**: tolerance in the eigenvalue solver to attain convergence. While a lower value might result in a more accurate solution, it might also lengthen the computation time.
- **max_iter**: The maximum number of times the eigenvalue solver can run. It continues if None is selected, unless convergence or additional stopping conditions are satisfied.
- **path_method**: chooses the approximation technique for geodesic distances on the graph. 'auto' (automatic selection) and 'FW' (Floyd-Warshall algorithm) are available options.
- **neighbors_algorithm**: A method for calculating the closest neighbors. 'Auto', 'ball_tree', 'kd_tree', and 'brute' are among the available options. 'auto' selects the best algorithm according to the input data.
- **metric**: The nearest neighbor search's distance metric. 'Minkowski' is the default; however, 'euclidean','manhattan', and several other options are also available.

Working of ISOMAP

- **Calculate the pairwise distances:** The algorithm starts by calculating the Euclidean distances between the data points.
- **Find nearest neighbors according to these distances:** For each data point, its k nearest neighbor is determined by that distance.
- **Create a neighborhood plot:** the edges of each point are aligned with their closest neighbors, which creates a diagram that represents the data's regional structure.
- **Calculate geodesic distances:** The Floyd algorithm sorts through all the pairs of data points in a neighborhood graph and finds the most distant paths. geodesic distances are represented by these shortest paths.
- **Perform dimensional reduction:** Classical Multi Scaling MDS is used for geodesic distance matrices that result in low dimensional embedding of data.

```
from sklearn.datasets import make_s_curve
from sklearn.manifold import Isomap
import matplotlib.pyplot as plt

# Generate S-curve data
X, color = make_s_curve(n_samples=1000, random_state=42)

# Apply Isomap
isomap = Isomap(n_neighbors=10, n_components=2)
X_isomap = isomap.fit_transform(X)

# Plot the original and reduced-dimensional data
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

ax[0].scatter(X[:, 0], X[:, 2], c=color, cmap=plt.cm.Spectral)
ax[0].set_title('Original 3D Data')

ax[1].scatter(X_isomap[:, 0], X_isomap[:, 1], c=color,
cmap=plt.cm.Spectral)
ax[1].set_title('Isomap Reduced 2D Data')

plt.show()
```



This sample of code illustrates how to apply the dimensionality reduction method Isomap to a dataset of S curves. Plotting the original 3D data next to the reduced 2D data for visualization follows the generation of S-curve data with 3D coordinates using Isomap. The fundamental connections between data points in a lower-dimensional space are preserved by the Isomap transformation, which captures the underlying geometric structure. With the inherent patterns in the data still intact, the resultant visualization shows how effective Isomap is at unfolding the S-curve structure in a more manageable 2D representation.

Advantages and Disadvantages of Isomap

Advantages

- **Capturing non linear relationships:** Unlike linear dimensional reduction techniques such as PCA, Isomap is able to capture the underlying non linear structure of the data.
- **Global structure**: Isomap's goal is to preserve the overall relationship between data points, which will give a better representation of the entire manifold.
- **Globally optimised:** The algorithm guarantees that on the built neighborhood graph, where geodesic distances are defined, a global optimal solution will be found.

Disadvanatges

- **Computational cost**: for large datasets, computation of geodesic distance using Floyd's algorithm can be computationally expensive and lead to a longer run time.
- **Sensitive to parameter settings**: incorrect selection of the parameters may lead to a distortion or misleading insert.
- May be difficult for manifolds with holes or topological complexity, which may lead to inaccurate representations: Isomap is not capable of performing well in a manifold that contains holes or other topological complexity.

Applications of Isomap

- **Visualization**: High-dimensional data like face images can be visualized in a lower-dimensional space, enabling easier exploration and understanding.
- **Data exploration**: Isomap can help identify clusters and patterns within the data that are not readily apparent in the original high-dimensional space.
- **Anomaly detection**: Outliers that deviate significantly from the underlying manifold can be identified using Isomap.
- **Machine learning tasks**: Isomap can be used as a pre-processing step for other machine learning tasks, such as classification and clustering, by improving the performance and interpretability of the models.

- **Least Square method** is a fundamental mathematical technique widely used in **data analysis, statistics, and regression modeling** to identify the **best-fitting curve or line** for a given set of data points. This method ensures that the overall error is reduced, providing a highly accurate model for predicting future data trends.
- In statistics, when the data can be represented on a cartesian plane by using the independent and dependent variable as the x and y coordinates, it is called **scatter data.** This data might not be useful in making interpretations or predicting the values of the dependent variable for the independent variable. So, we try to get an **equation of a line that fits best to the given data points** with the help of the **Least Square Method**.

What is the Least Square Method?

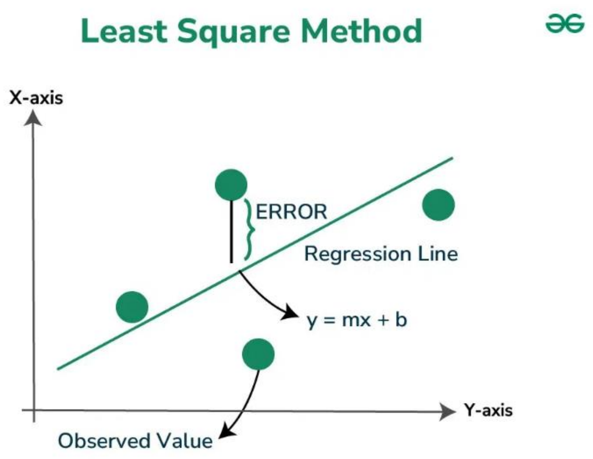
Least Square Method is used to derive a generalized linear equation between two variables. when the value of the dependent and independent variable is represented as the x and y coordinates in a 2D cartesian coordinate system. Initially, known values are marked on a plot. The plot obtained at this point is called a scatter plot.

Then, we try to represent all the marked points as a straight line or a **linear equation**. The equation of such a line is obtained with the help of the Least Square method. This is done to get the value of the dependent variable for an independent variable for which the value was initially unknown. This helps us to make predictions for the value of dependent variable.

Least Square Method Definition

Least Squares method is a statistical technique used to find the equation of best-fitting curve or line to a set of data points by minimizing the sum of the squared differences between the observed values and the values predicted by the model.

This method aims at minimizing the sum of squares of deviations as much as possible. The line obtained from such a method is called a **regression line** or **line of best fit.**



Formula for Least Square Method

Least Square Method formula is used to find the best-fitting line through a set of data points. For a simple linear regression, which is a line of the form *y*=m*x*+*c*, where *y* is the dependent variable, *x* is the independent variable, *a* is the slope of the line, and *b* is the y-intercept, the formulas to calculate the slope (*m*) and intercept (*c*) of the line are derived from the following equations:

1. **Slope (*m*) Formula:** $m = n(\sum xy) - (\sum x)(\sum y) / n(\sum x^2) - (\sum x)^2$
2. **Intercept (c) Formula:** $c = (\sum y) - a(\sum x) / n$

Where:

- *n* is the number of data points,
- $\sum xy$ is the sum of the product of each pair of *x* and *y* values,
- $\sum x$ is the sum of all *x* values,
- $\sum y$ is the sum of all *y* values,
- $\sum x^2$ is the sum of the squares of *x* values.

Formula for Least Square Method

Least Square Method formula is used to find the best-fitting line through a set of data points. For a simple linear regression, which is a line of the form *y*=m*x*+*c*, where *y* is the dependent variable, *x* is the independent variable, *a* is the slope of the line, and *b* is the y-intercept, the formulas to calculate the slope (*m*) and intercept (*c*) of the line are derived from the following equations:

1. **Slope (*m*) Formula:** $m = n(\sum xy) - (\sum x)(\sum y) / n(\sum x^2) - (\sum x)^2$
2. **Intercept (c) Formula:** $c = (\sum y) - a(\sum x) / n$

Where:

- *n* is the number of data points,
- $\sum xy$ is the sum of the product of each pair of *x* and *y* values,
- $\sum x$ is the sum of all *x* values,
- $\sum y$ is the sum of all *y* values,
- $\sum x^2$ is the sum of the squares of *x* values.

The steps to find the line of best fit by using the least square method is discussed below:

- **Step 1:** Denote the independent variable values as x_i and the dependent ones as y_i.
- **Step 2:** Calculate the average values of x_i and y_i as X and Y.
- **Step 3:** Presume the equation of the line of best fit as y = mx + c, where m is the slope of the line and c represents the intercept of the line on the Y-axis.
- **Step 4:** The slope m can be calculated from the following formula:

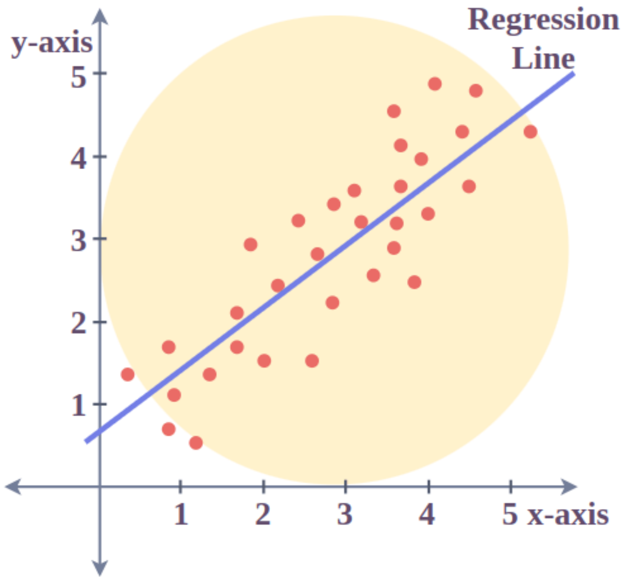
$$m = [\sum (X - x_i) \times (Y - y_i)] / \sum (X - x_i)^2$$

- **Step 5:** The intercept c is calculated from the following formula:
$$c = Y - mX$$

Thus, we obtain the line of best fit as y = mx + c, where values of m and c can be calculated from the formulae defined above.
These formulas are used to calculate the parameters of the line that best fits the data according to the criterion of the least squares, minimizing the sum of the squared differences between the observed values and the values predicted by the linear model.

Least Square Method Graph

Let us have a look at how the data points and the line of best fit obtained from the Least Square method look when plotted on a graph.



The red points in the above plot represent the data points for the sample data available. **Independent variables are plotted as x-coordinates and dependent ones are plotted as y-coordinates**. The equation of the line of best fit obtained from the Least Square method is plotted as the red line in the graph.

We can conclude from the above graph that how the **Least Square method helps us to find a line that best fits the given data points** and hence can be used to make further predictions about the value of the dependent variable where it is not known initially.

Limitations of the Least Square Method

The Least Square method assumes that the data is evenly distributed and doesn't contain any outliers for deriving a line of best fit. But, this method doesn't provide accurate results for unevenly distributed data or for data containing outliers.

Check: Least Square Regression Line

Least Square Method Solved Examples

Problem 1: Find the line of best fit for the following data points using the Least Square method: (x,y) = (1,3), (2,4), (4,8), (6,10), (8,15).

Solution:

Here, we have x as the independent variable and y as the dependent variable. First, we calculate the means of x and y values denoted by X and Y respectively.

X = (1+2+4+6+8)/5 = 4.2

Y = (3+4+8+10+15)/5 = 8

i	y_i	$X - x_i$	$Y - y_i$	$(X-x_i)*(Y-y_i)$	$(X - x_i)^2$
1	**3**	**3.2**	**5**	**16**	**10.24**
2	**4**	**2.2**	**4**	**8.8**	**4.84**
4	**8**	**0.2**	**0**	**0**	**0.04**
6	**10**	**-1.8**	**-2**	**3.6**	**3.24**
8	**15**	**-3.8**	**-7**	**26.6**	**14.44**
Sum (∑)		**0**	**0**	**55**	**32.8**

The slope of the line of best fit can be calculated from the formula as follows:

$$m = (\sum (X - x_i)*(Y - y_i)) / \sum(X - x_i)^2$$

$$m = 55/32.8 = 1.68 \text{ (rounded upto 2 decimal places)}$$

Now, the intercept will be calculated from the formula as follows:

$$c = Y - mX$$

$$c = 8 - 1.68*4.2 = 0.94$$

Thus, the equation of the line of best fit becomes, $y = 1.68x + 0.94$.

Genetic Algorithms

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random searches provided with historical data to direct the search into the region of better performance in solution space. **They are commonly used to generate high-quality solutions for optimization problems and search problems.**

Genetic algorithms simulate the process of natural selection which means those species that can adapt to changes in their environment can survive and reproduce and go to the next generation. In simple words, they simulate "survival of the fittest" among individuals of consecutive generations to solve a problem. **Each generation consists of a population of individuals** and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.

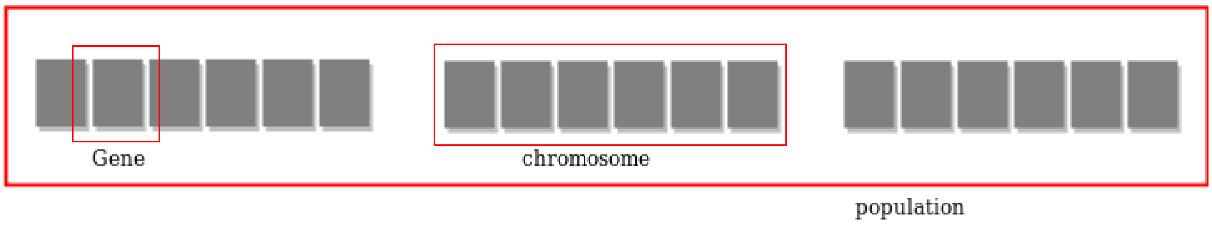
Foundation of Genetic Algorithms

Genetic algorithms are based on an analogy with the genetic structure and behavior of chromosomes of the population. Following is the foundation of GAs based on this analogy –

1. Individuals in the population compete for resources and mate
2. Those individuals who are successful (fittest) then mate to create more offspring than others
3. Genes from the "fittest" parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.
4. Thus each successive generation is more suited for their environment.

Search space

The population of individuals are maintained within search space. Each individual represents a solution in search space for given problem. Each individual is coded as a finite length vector (analogous to chromosome) of components. These variable components are analogous to Genes. Thus a chromosome (individual) is composed of several genes (variable components).



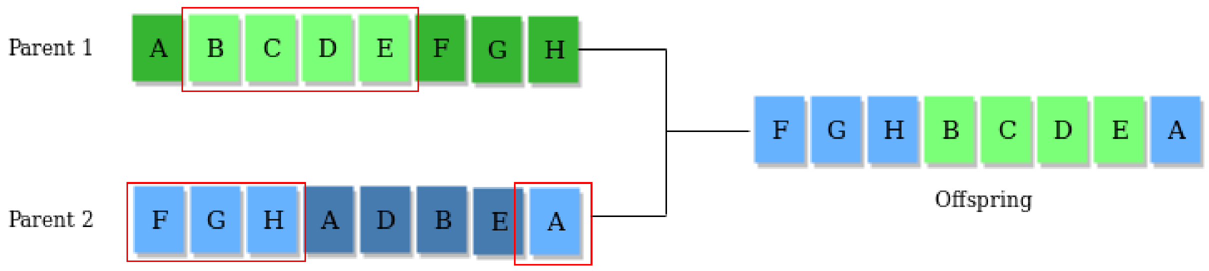
Fitness Score

A Fitness Score is given to each individual which **shows the ability of an individual to "compete"**. The individual having optimal fitness score (or near optimal) are sought.
The GAs maintains the population of n individuals (chromosome/solutions) along with their fitness scores.The individuals having better fitness scores are given more chance to reproduce than others. The individuals with better fitness scores are selected who mate and produce **better offspring** by combining chromosomes of parents. The population size is static so the room has to be created for new arrivals. So, some individuals die and get replaced by new arrivals eventually creating new generation when all the mating opportunity of the old population is exhausted. It is hoped that over successive generations better solutions will arrive while least fit die.
Each new generation has on average more "better genes" than the individual (solution) of previous generations. Thus each new generations have better **"partial solutions"** than previous generations. Once the offspring produced having no significant difference from offspring produced by previous populations, the population is converged. The algorithm is said to be converged to a set of solutions for the problem.

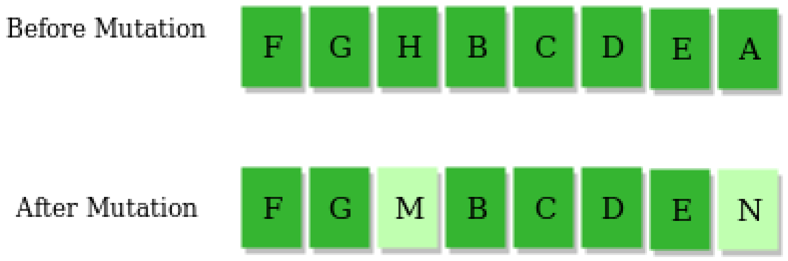
Operators of Genetic Algorithms

Once the initial generation is created, the algorithm evolves the generation using following operators –
1) Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations.
2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are

exchanged thus creating a completely new individual (offspring). For example –



Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in the population to avoid premature convergence. For example –



The whole algorithm can be summarized as –

```
1) Randomly initialize populations p
2) Determine fitness of population
3) Until convergence repeat:
      a) Select parents from population
      b) Crossover and generate new population
      c) Perform mutation on new population
      d) Calculate fitness for new population
```

Why use Genetic Algorithms

- They are Robust
- Provide optimisation over large space state.
- Unlike traditional AI, they do not break on slight change in input or presence of noise

Application of Genetic Algorithms

Genetic algorithms have many applications, some of them are –

- Recurrent Neural Network
- Mutation testing

- Code breaking
- Filtering and signal processing
- Learning fuzzy rule base etc

Genetic Algorithm Solved Example

- Consider the function of maximizing the function

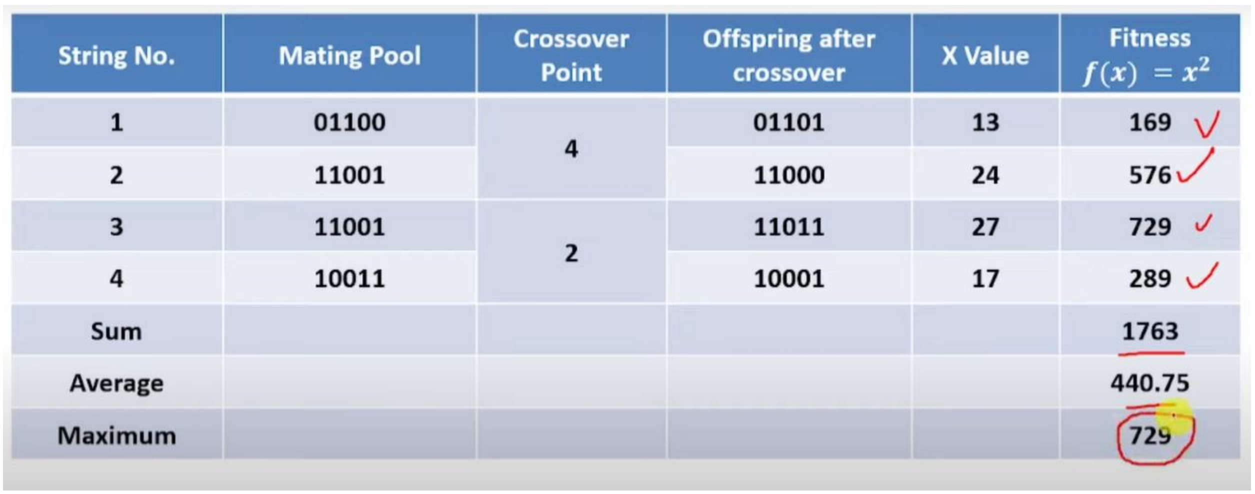
$$f(x) = x^2$$

- where x in permitted to vary between 0 to 31
- **Select Encoding Technique**
- The minimum value is 0 and maximum value is 31 ✓
- Using a five-bit binary integer, numbers between **0 (00000)** and **31 (11111)** can be obtained.
- The objective function here *is* $f(x) = x^2$,which *is* to be maximized.

Select Initial Population

- To start with, select initial population are random.
- Here initial population of size 4 is chosen, but any number of populations can be selected based on the requirement and application.
- Table shows an initial population randomly selected.

String No.	Initial Population (Randomly Selected)	X Value	Fitness $f(x) = x^2$	Prob	% Prob	Expected Count	Actual Count
1	01100	12	144	0.1247	12.47	0.4987	1 ✓
2	11001	25	625	0.5411	54.11	2.1645	2
3	00101	5	25	0.0216	2.16	0.0866	0
4	10011	19	181	0.3126	31.26	1.2502	1
Sum			1155	1.0	100	4	4
Average			288.75	0.25	25	1	1
Maximum			625	0.5411	54.11	2.1645	2



String No.	Mating Pool	Crossover Point	Offspring after crossover	X Value	Fitness $f(x) = x^2$
1	01100	4	01101	13	169 ✓
2	11001		11000	24	576 ✓
3	11001	2	11011	27	729 ✓
4	10011		10001	17	289 ✓
Sum					1763
Average					440.75
Maximum					729