Fundamentals of Machine Learning

Roozbeh Sanaei

June 30, 2024

Contents

1 Linear Algebra **4**
- 1.1 Matrix Decomposition Techniques . . . 4
  - 1.1.1 Eigenvalue Decomposition . . . 4
  - 1.1.2 Singular Value Decomposition (SVD) . . . 5
  - 1.1.3 Orthogonality and Orthonormality . . . 6

2 Fundamental Concepts of Machine Learning **7**
  - 2.0.1 Different Learning Paradigms . . . 7
  - 2.0.2 Overfitting and its Mitigation . . . 8
  - 2.0.3 Feature Selection . . . 10

3 Dimensionality Reduction **11**
- 3.1 Independent Component Analysis (ICA) . . . 11
  - 3.1.1 ICA Overview . . . 11
  - 3.1.2 Different Algorithms in ICA . . . 12
  - 3.1.3 Infomax . . . 13
  - 3.1.4 What is Whitening? . . . 14
  - 3.1.5 Fast Independent Component Analysis . . . 15
  - 3.1.6 JADE . . . 16
- 3.2 SNE, t-SNE, UMAP . . . 17
  - 3.2.1 SNE . . . 17
  - 3.2.2 Comparative Analysis of SNE, t-SNE, and UMAP . . . 18
  - 3.2.3 SNE, t-SNE and UMAP Comparison . . . 19
- 3.3 Linear Discriminant Analysis (LDA) . . . 20
- 3.4 Sparse Dictionary Learning Overview . . . 21
- 3.5 Non-negative Matrix Factorization(NMF) . . . 22
- 3.6 Multidimensional Scaling (MDS) . . . 23
- 3.7 Isomap (Isometric Mapping) . . . 24

4 Clustering **25**
4.1 K-Means . . . 25
4.1.1 Elbow Method for Determining Optimal Number of Clusters in K-means . . . 26
4.1.2 Silhouette Analysis for Determining Optimal Clusters in K-means . . . 27
4.2 The Canopy Method . . . 29
4.3 Gaussian Mixture Models (GMMs) . . . 30
4.3.1 Challenges in Gaussian Mixture Models (GMM) . . . 31
4.3.2 Comparison of GMMs and K-means . . . 32
4.4 DBSCAN: Density-Based Spatial Clustering of Applications with Noise . . . 33
4.5 OPTICS(Ordering Points To Identify the Clustering Structure) Algorithm . . . 34
4.6 Spectral Clustering Algorithms . . . 35
4.7 Markov Chain Clustering (MCL) . . . 36
4.8 Agglomerative Clustering Algorithm . . . 37

5 Supervised Machine Learning **38**
5.1 Linear Regression Model . . . 38
5.1.1 Assumptions Of Linear Regression . . . 38
5.2 Ordinary Least Squares (OLS) . . . 39
5.2.1 OLS as Projection . . . 40
5.2.2 Applying SVD to OLS and Ridge Regression . . . 41
5.2.3 Relationship between CEF and Regression . . . 42
5.3 Method of Moments . . . 43
5.3.1 General Framework of Moment Conditions . . . 44
5.3.2 Instrumental Variables in Regression Models . . . 45
5.3.3 Generalized Method of Moments (GMM) . . . 46
5.4 Maximum Likelihood . . . 47
5.4.1 Maximum Likelihood Estimation (MLE) . . . 47
5.4.2 OLS Estimator using Maximum Likelihood . . . 48
5.5 Logistic Regression . . . 52
5.6 Generalized Linear Models (GLMs) . . . 53
5.7 Support Vector Machines (SVMs) . . . 54
5.7.1 Lagrangian SVMs . . . 55
5.7.2 Comparison Between OLS and SVM . . . 56
5.8 Decision Tree Algorithms . . . 57
5.8.1 ID3 . . . 57
5.8.2 Comparison of ID3 and C4.5 Algorithms . . . 58
5.8.3 Decision Tree Pruning Methods . . . 60
5.8.4 Decision Tree Splitting Criteria . . . 61
5.9 Ensemble Models . . . 62
5.9.1 Multivariate Adaptive Regression Splines (MARS) . . . 63
5.9.2 Ensemble Models Comparison . . . 66
5.9.3 AdaBoost . . . 67

5.9.4 Gradient Boosting . . . 68

6 Evaluation **70**
6.1 Evaluation Approaches . . . 70
6.1.1 K-fold Validation . . . 70
6.1.2 The ROC Curve in Binary Classification . . . 71
6.1.3 Accuracy Metrics . . . 72
6.1.4 Lift and Drift Charts . . . 73

7 Anomalies and Outliers **74**
7.1 Anomalies and Outliers . . . 74
7.2 Isolation Forest Algorithm . . . 75
7.3 Cook's Distance . . . 76
7.4 Quartiles and Interquartile Range (IQR) . . . 77
7.5 Local Outlier Factor . . . 78
7.6 Mahalanobis Distance . . . 79
7.7 Minimum Covariance Determinant (MCD) Method . . . 80
7.8 Single-Class SVM . . . 81

8 Information Theory **82**
8.1 Shannon Uncertainty Formula . . . 82
8.2 Boltzmann's Entropy Formula . . . 83
8.3 Jensen's Inequality . . . 84
8.4 Fisher's Score and Fisher's Information . . . 85
8.5 Kullback-Leibler Divergence . . . 86
8.6 Mutual Information . . . 87

9 Time Series Forecasting **88**
9.1 Common Components of a Time Series . . . 88
9.2 Autocorrelation Function (ACF) . . . 89
9.3 Partial Autocorrelation Function (PACF) . . . 90
9.4 Autoregressive Integrated Moving Average (ARIMA) . . . 91
9.5 SARIMAX Model . . . 92
9.6 Simple Exponential Smoothing (SES) . . . 93
9.7 Damped Trend Method . . . 94
9.8 Holt's linear trend method . . . 95
9.9 Exponential Smoothing Methods . . . 96

Linear Algebra

1.1 Matrix Decomposition Techniques

1.1.1 Eigenvalue Decomposition

Eigenvalue decomposition is a matrix decomposition technique used in linear algebra. It expresses a given square matrix in terms of its eigenvalues and eigenvectors. The decomposition is essential in various scientific and engineering applications.

Definition of Eigenvalues and Eigenvectors

Given a square matrix A, an eigenvector v and an eigenvalue λ satisfy the equation:

$$Av = \lambda v$$

where v is a non-zero vector, and λ is a scalar.

Finding Eigenvalues

Eigenvalues of A are found by solving the characteristic equation:

$$\det(A - \lambda I) = 0$$

where det denotes the determinant of a matrix, and I is the identity matrix. The roots of the characteristic polynomial (a polynomial in λ) are the eigenvalues of A.

Finding Eigenvectors

For each eigenvalue λ, eigenvectors are found by solving:

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

This is a system of linear equations. Non-zero solutions for v are the eigenvectors corresponding to λ.

Eigenvalue Decomposition

If A has n linearly independent eigenvectors $\{v_1, v_2, \ldots, v_n\}$ corresponding to eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, then A can be factorized as:

$$A = VDV^{-1}$$

where V is the matrix whose i-th column is the eigenvector v_i, and D is the diagonal matrix with eigenvalues λ_i on the diagonal.

1.1.2 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a critical technique in linear algebra, utilized in various fields such as signal processing, statistics, and machine learning. It decomposes any matrix into three distinct matrices.

Definition of SVD

Given an $m \times n$ matrix A, SVD is defined as:

$$A = U\Sigma V^T$$

where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and Σ is an $m \times n$ diagonal matrix.

Matrices in SVD

- U (left-singular vectors): Columns of U are eigenvectors of AA^T.
- V (right-singular vectors): Columns of V are eigenvectors of A^TA.
- Σ (singular values): Diagonal entries are the square roots of the non-negative eigenvalues of A^TA or AA^T.

Computing SVD

- Compute eigenvalues and eigenvectors of AA^T and A^TA.
- Singular values in Σ are square roots of non-zero eigenvalues of A^TA.
- Columns of U are normalized eigenvectors of AA^T.
- Columns of V are normalized eigenvectors of A^TA.

Properties of SVD

- SVD exists for any $m \times n$ matrix.
- Singular values are non-negative and usually in descending order.
- U and V are orthogonal matrices.

1.1.3 Orthogonality and Orthonormality

Orthogonality

Definition: Two vectors $\mathbf{u}$ and $\mathbf{v}$ in a vector space are orthogonal if their dot product is zero:

$$\mathbf{u} \cdot \mathbf{v} = 0$$

In $\mathbb{R}^n$, this is:

$$u_1v_1 + u_2v_2 + \cdots + u_nv_n = 0$$

Importance: Orthogonal vectors minimize errors and dependencies in computations and are used in methods like the Gram-Schmidt process for orthogonal bases.

Orthonormality

Definition: A set of vectors is orthonormal if all vectors are orthogonal to each other and each vector is of unit length. For vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{u} \cdot \mathbf{v} = 0 \, (\text{if } \mathbf{u} \neq \mathbf{v})$$

$$\|\mathbf{u}\| = \|\mathbf{v}\| = 1$$

Importance: Orthonormal vectors simplify computations and are used in Fourier series, quantum mechanics, and signal processing.

Applications in Linear Transformations

In matrix terms, a matrix A with orthonormal columns satisfies:

$$A^TA = I$$

where A^T is the transpose of A, and I is the identity matrix. This property is crucial in preserving lengths and angles in transformations.

Fundamental Concepts of Machine Learning

2.0.1 Different Learning Paradigms

Supervised Learning

Training on labeled data to learn the mapping from input to output.

Application: Used in regression and classification tasks.

Example: Predicting house prices based on features like size and location.

Unsupervised Learning

Learning patterns from unlabeled data without explicit instruction on what to predict.

Application: Clustering, association, dimensionality reduction.

Example: Identifying customer segments in marketing data.

Semi-Supervised Learning

Combines both labeled and unlabeled data for training.

Application: Useful in scenarios where labeled data are limited.

Example: Language translation models with limited annotated data.

Self-Supervised Learning

Uses unlabeled data and generates its own labels from the data's structure.

Application: Gaining traction in computer vision and natural language processing.

Example: Pretext tasks in deep learning models like predicting the next word in a sentence.

2.0.1.1 Reinforcement Learning

Learning to make decisions by performing actions to achieve a goal.

Application: Robotics, gaming, navigation.

Example: A robot learning to navigate through a maze.

2.0.2 Overfitting and its Mitigation

Overfitting

Definition: Overfitting occurs when a model learns both the underlying patterns and the noise in the training data, leading to poor generalization to new data.

Consequence and Impact: The model exhibits high accuracy on training data but poor performance on unseen data.

Reason: Often due to excessive complexity in the model relative to the amount of training data, leading to the learning of noise.

Methods to Overcome Overfitting

Regularization: Adding a penalty to the model's loss function to constrain its complexity.

Dropout: Randomly ignoring neurons during training in neural networks to prevent over-reliance on specific patterns.

Early Stopping: Halting training when the model's performance on validation data starts to worsen.

Data Augmentation: Artificially increasing training data diversity through transformations.

Ensembling: Combining predictions from multiple models to average out errors.

Feature Selection: Choosing the most relevant features for training to reduce model complexity.

Model Simplification: Reducing the number of layers or parameters in the model.

Increasing Dataset Size: Expanding the training dataset to provide more comprehensive examples.

Bayesian Neural Networks: Incorporating probabilistic approaches in neural networks to manage uncertainty.

Bias and Variance in Machine Learning

- **Bias:**
  - Measures the difference between the model's average prediction and the true values.
  - $\text{Bias}^2[\hat{f}(x)] = \left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2$
    - Where $\hat{f}(x)$ is the model's prediction, $f(x)$ is the true function, and $\mathbb{E}[\hat{f}(x)]$ is the expected value of the model's predictions.
  - High bias can lead to underfitting, indicating a model too simple to capture the data's complexity.
- **Variance:**
  - Measures the variation of model predictions for a given data point.
  - $\text{Variance}[\hat{f}(x)] = \mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right]$
    - Where $\mathbb{E}[\hat{f}(x)]$ is the expected value of the model's predictions.
  - High variance can lead to overfitting, where the model captures noise as if it were a significant signal.
- **Bias-Variance Trade-Off:**
  - Improving the model to reduce bias typically increases its variance, and vice versa.
  - $\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$
    - Irreducible Error represents the error inherent in the problem itself, due to factors like noise.

2.0.3 Feature Selection

Filter Methods

These methods assess features independently of any learning algorithm, using statistical measures.

- *Correlation Coefficient*: Measures linear relationship between features and the target.
- *Chi-squared Test*: Assesses independence between categorical features and the target.
- *Information Gain*: Evaluates reduction in entropy from adding a feature.

Wrapper Methods

These methods evaluate feature subsets through a specific machine learning algorithm, focusing on model performance.

- *Recursive Feature Elimination (RFE)*: Eliminates least important features iteratively.
- *Forward Selection*: Adds most significant feature iteratively from an empty set.
- *Backward Elimination*: Removes least significant feature iteratively from a full set.

Embedded Methods

These methods integrate feature selection within the learning algorithm during model training.

- *Lasso Regression*: Uses L1 regularization to eliminate irrelevant features.
- *Elastic Net*: Combines L1 and L2 regularization for balanced feature selection.
- *Random Forest Importance*: Assesses feature importance based on their contribution to random forest performance.

Advantages of Feature Selection

- Reducing dimensionality
- Improving model performance
- Enhancing interpretability
- Reducing computational complexity
- Improving data quality
- Enhancing model transparency
- Addressing multicollinearity

Dimensionality Reduction

3.1 Independent Component Analysis (ICA)

3.1.1 ICA Overview

Basics of ICA

- **Objective:** Decomposing a multivariate signal into independent non-Gaussian components.
- **Use Cases:** Applied in blind source separation, image processing, and complex data analysis.

How ICA Works

- **Statistical Independence:** Components assumed to be statistically independent.
- **Non-Gaussianity:** Non-Gaussian sources sum up to a more Gaussian-like distribution, aiding separation.

Process

- **Input:** Linear mixtures of unknown independent components.
- **Algorithm:** Adjusts weights to maximize independence of output signals.
- **Output:** Independent components for signal reconstruction and source identification.

Applications

- **Audio Processing:** Separating voices in the 'cocktail party problem'.
- **Medical Imaging:** Identifying brain activities and artifacts in fMRI.
- **Financial Analysis:** Analyzing complex financial data to extract underlying factors.

Limitations

- **Assumptions:** Relies on specific assumptions that may not always hold.
- **Model Order Determination:** Difficulty in deciding the number of components.
- **Ambiguity:** Possibility of permutation and scaling ambiguities in solutions.

3.1.2 Different Algorithms in ICA

- **FastICA:** Known for its computational efficiency, FastICA uses a fixed-point iteration scheme to maximize the non-Gaussianity of projected data.
- **Infomax and Extended Infomax:** These algorithms maximize the information transfer from the input to the output of a neural network. Extended Infomax can handle both sub- and super-Gaussian sources.
- **JADE (Joint Approximate Diagonalization of Eigenmatrices):** JADE operates by jointly diagonalizing a set of covariance matrices to extract the independent components.
- **CUmulative Distributions-based ICA (CUDI):** CUDI maximizes non-Gaussianity using cumulative distribution functions.
- **Second-order blind identification (SOBI):** SOBI utilizes second-order statistics over different time delays for ICA, particularly effective for time-dependent signals.
- **Probabilistic ICA:** This method incorporates a probabilistic model, often using a likelihood function, to estimate independent components, and is closely related to factor analysis.
- **Temporal ICA:** Designed specifically for time-series data, it focuses on the temporal structure of the data to separate sources.
- **Complex-valued ICA:** This variant is used for complex-valued data (with real and imaginary parts), as found in some signal processing applications.
- **Nonlinear ICA:** Deals with mixtures that are nonlinear combinations of the source signals, unlike traditional linear ICA models.

3.1.3 Infomax

Infomax is a principle used in information theory and neural network training, aiming to maximize the mutual information between the input and output of a system. This principle is often applied in unsupervised learning, where the goal is to find a representation of the input data that preserves as much information as possible.

Entropy (H)

- Entropy measures the amount of uncertainty or randomness in a random variable.
- For a random variable X, it is defined as:

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

Mutual Information (I)

- Mutual information quantifies the shared information between two random variables, X and Y.
- It is defined as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

- Alternatively, it can be expressed using entropy:

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

$$I(X;Y) = H(Y) - H(Y|X)$$

Objective of Infomax

- The goal is to maximize the mutual information $I(X;Y)$.
- This involves maximizing $H(Y)$ and minimizing $H(Y|X)$.

Application in Neural Networks

- The output Y is a function of the input X and the network weights W:

$$Y = f(WX)$$

- The objective is to find the optimal W that maximizes $I(X;Y)$.

Optimization

- Typically involves using gradient-based optimization techniques.
- Adjusts weights W by:

$$W_{\text{new}} = W_{\text{old}} + \eta \frac{\partial I(X;Y)}{\partial W}$$

Constraints and Regularization

- Additional constraints, such as weight regularization, are often applied.
- These help prevent overfitting and ensure a more robust learning process.

3.1.4 What is Whitening?

- **Whitening: Transformation of Data**
  - Whitening is a process that transforms data so that the covariance matrix of the resulting dataset is the identity matrix.
  - the means that the features are uncorrelated and each feature has unit variance.
  - This involves decorrelating the features and normalizing their variance.
- **Compute the Covariance Matrix**
  - Given a dataset X with n features and m samples.
  - Calculate the covariance matrix Σ as:

$$\Sigma = \frac{1}{m}(X - \bar{X})^T(X - \bar{X})$$

  - $\bar{X}$ is the mean vector for the dataset.
- **Perform Eigenvalue Decomposition**
  - Apply eigenvalue decomposition to the covariance matrix Σ:

$$\Sigma = VDV^T$$

  - V is the matrix of eigenvectors, D is the diagonal matrix of eigenvalues.
- **Whiten the Data**
  - Transform the data to obtain the whitened data X_{white}:

$$X_{\text{white}} = ED^{-\frac{1}{2}}V^T(X - \bar{X})$$

  - $D^{-\frac{1}{2}}$ is obtained by taking the reciprocal square root of each non-zero element in D.
  - E is an optional scaling matrix, often the identity matrix I.

3.1.5 Fast Independent Component Analysis

Fast Independent Component Analysis (Fast ICA) is an algorithm used for the separation of a multivariate signal into additive subcomponents. It's often used in the context of blind source separation, where the goal is to separate a set of signals that have been mixed together.

Basic Model

The basic model of ICA can be represented as:

$$\mathbf{x} = \mathbf{A}\mathbf{s}$$

where

- $\mathbf{x}$ is the observed mixed signal.
- $\mathbf{A}$ is the mixing matrix.
- $\mathbf{s}$ is the vector of independent source signals.

Goal of Fast ICA

The goal is to estimate the matrix $\mathbf{W}$, which is the unmixing matrix, such that:

$$\mathbf{s} \approx \mathbf{W}\mathbf{x}$$

Fast ICA Algorithm

1. **Centering and Whitening:** First, the observed signals are centered and whitened. Centering involves subtracting the mean, and whitening is done to transform the variables into uncorrelated variables with unit variance.
2. **Maximization of Non-Gaussianity:** The core of Fast ICA is to find a linear combination of the whitened variables that maximizes non-Gaussianity. Non-Gaussianity can be measured in several ways, such as using kurtosis or negentropy.
3. **Iterative Fixed-Point Algorithm:** The Fast ICA algorithm finds the independent components by iterating the following steps:
   (a) Choose an initial weight vector $\mathbf{w}$.
   (b) Update $\mathbf{w}$ using the formula:
   $$\mathbf{w}^+ = E\{\mathbf{x}g(\mathbf{w}^T\mathbf{x})\} - E\{g'(\mathbf{w}^T\mathbf{x})\}\mathbf{w}$$
   where g is a non-linear function, which is chosen based on the measure of non-Gaussianity (like kurtosis or negentropy) and g' is its derivative.
   (c) Normalize the weight vector:
   $$\mathbf{w} = \frac{\mathbf{w}^+}{\|\mathbf{w}^+\|}$$
   (d) Repeat until convergence.
4. **Extraction of Independent Components:** Once the algorithm converges, the independent components are given by:
   $$\mathbf{s} = \mathbf{W}\mathbf{x}$$

3.1.6 JADE

JADE ICA is a statistical technique used to separate a multivariate signal into additive subcomponents that are maximally independent from each other. It is particularly useful in blind source separation tasks like separating audio signals.

1. Centering and Whitening

Centering: Removes the mean from the data to ensure zero mean.
For data matrix XX and its mean $E[X]$:

$$X_{\text{centered}} = X - E[X]$$

Whitening: Transforms variables into uncorrelated variables with unit variance, reducing ICA to finding an orthogonal matrix.
For whitened matrix Z, diagonal matrix of eigenvalues D, and matrix of eigenvectors E:

$$Z = D^{-\frac{1}{2}} E^T X_{\text{centered}}$$

2. Cumulant Calculation

Fourth-order cumulants capture non-Gaussian features like kurtosis.
For components z_i, z_j, z_k, z_l of whitened data Z:

$$\begin{aligned}\text{cum}(z_i, z_j, z_k, z_l) = {} & E[z_i z_j z_k z_l] - E[z_i]E[z_j z_k z_l] \\ & - E[z_i]E[z_j]E[z_k z_l] - E[z_i]E[z_j z_k]E[z_l] \\ & + 2E[z_i]E[z_j]E[z_k]E[z_l]\end{aligned}$$

Where cum is the fourth-order cumulant function.

$$K_{ijkl} = \text{cum}(z_i, z_j, z_k, z_l)$$

Where K_{ijkl} is the cumulant matrix for the quadruple (z_i, z_j, z_k, z_l).

3. Diagonalization

Finding an orthogonal transformation to make the cumulant matrices as diagonal as possible, indicating statistical independence.
For orthogonal unmixing matrix W and cumulant matrices K:

$$W^* = \arg\min_W \sum (\text{off-diagonal elements of } WKW^T)^2$$

4. Independent Components

Obtaining independent components from observed data by transforming whitened data with the matrix from diagonalization.
For matrix of independent components S and optimal unmixing matrix W:

$$S = WZ$$

3.2 SNE, t-SNE, UMAP

3.2.1 SNE

Similarity in the Original Space

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

where $p_{j|i}$ is the probability of picking x_j as a neighbor of x_i, x_i, x_j are data points in the high-dimensional space, $||x_i - x_j||$ is the Euclidean distance between x_i and x_j, σ_i is the variance of the Gaussian distribution around x_i, and k is an index for summation over all points except i.

Symmetrized Similarities

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

where p_{ij} is the joint probability symmetrizing $p_{j|i}$ and $p_{i|j}$, and N is the total number of data points.

Similarity in the Reduced Space

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

where q_{ij} is the probability of y_i and y_j being neighbors in the low-dimensional space, y_i, y_j are data points in the low-dimensional space, and k, l are indices for summation over all distinct pairs of points.

Minimization of Kullback-Leibler Divergence

$$KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

where $KL(P||Q)$ is the Kullback-Leibler divergence between the high-dimensional and low-dimensional representations, p_{ij}, q_{ij} are the joint probabilities in the high and low-dimensional spaces, respectively.

Gradient Descent

$$\frac{\delta KL(P||Q)}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$$

where $\frac{\delta KL(P||Q)}{\delta y_i}$ is the gradient of the Kullback-Leibler divergence with respect to point y_i, y_i, y_j are data points in the low-dimensional space, p_{ij}, q_{ij} are the joint probabilities in the high and low-dimensional spaces, respectively.

3.2.2 Comparative Analysis of SNE, t-SNE, and UMAP

High Dimensional Probabilities Calculation

- **SNE:** Utilizes scaled Euclidean distance, leading to non-symmetric dissimilarities due to the variance parameter σ_i.
- **t-SNE Difference:** Implements symmetrization to make high-dimensional probabilities symmetric.
- **UMAP Difference:** Works with similarities instead of probabilities, using a different metric function.

Low Dimensional Probabilities Calculation

- **SNE:** Uses Gaussian neighborhoods with fixed variance for low-dimensional probabilities.
- **t-SNE Difference:** Adopts the Student t-distribution to solve the crowding problem.
- **UMAP Difference:** Does not normalize low-dimensional similarities, improving performance.

Cost Function and Optimization

- **SNE:** Employs Kullback-Leibler divergence, facing optimization challenges.
- **t-SNE Difference:** Retains KL divergence but modifies probabilities approach.
- **UMAP Difference:** Uses cross-entropy and stochastic gradient descent, capturing more global structure.

Focus on Data Structure Preservation

- **SNE:** Aims to preserve both local and global structures, but has challenges.
- **t-SNE Difference:** Prioritizes local structure conservation, effective in visualizing clusters.
- **UMAP Difference:** Preserves more global structure due to its methodological approach.

3.2.3 SNE, t-SNE and UMAP Comparison

Similarity

- **SNE Similarity:** The SNE algorithm uses a Gaussian kernel to model the probability that a point x_i in a high-dimensional space would choose another point x_j as its neighbor:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

  Variables: x_i, x_j, High-dimensional data points; σ_i^2, Variance of the Gaussian kernel centered at x_i.

- **t-SNE Symmetrization:** t-SNE modifies the SNE approach by symmetrizing the probabilities to address the original asymmetry:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

  Variables: $p_{j|i}, p_{i|j}$, Conditional probabilities from SNE; N, Total number of data points.

- **UMAP Similarity Measure:** UMAP employs a fuzzy set approach for similarity, focusing on both local and global data structures:

$$\mu_{ij} = \exp(-\max(0, d(x_i, x_j) - \rho_i)/\sigma_i)$$

  Variables: $d(x_i, x_j)$, User-defined metric for distance; ρ_i, σ_i, Parameters for local neighborhood adjustment.

Similarity in Reduced Space

- **SNE Similarity in Reduced Space:** SNE in the reduced space uses a similar approach to the high-dimensional space but with fixed variance:

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)}$$

  Variables: y_i, y_j, Low-dimensional embeddings of the high-dimensional data points x_i, x_j. This calculates the probability of the low-dimensional embeddings of points, emphasizing their relative proximities.

- **t-SNE Symmetrization in Reduced Space:** t-SNE uses the Student t-distribution for probabilities in the reduced space:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}}$$

  Variables: y_i, y_j, Low-dimensional embeddings. The use of t-distribution helps in mitigating the crowding problem and allows t-SNE to better model the relationships between points in the reduced space.

- **UMAP Similarity Measure in Reduced Space:** UMAP uses a different formulation for low-dimensional similarities:

$$\nu_{ij} = \frac{1}{1 + a||y_i - y_j||^{2b}}$$

  Variables: y_i, y_j, Low-dimensional embeddings; a, b, Parameters learned during the optimization process. This measure helps UMAP to preserve the topological structure of the data in the reduced space, ensuring a balance between local and global structures.

Loss Functions

- **SNE Loss Function:** SNE uses the Kullback-Leibler divergence as its loss function:

$$C_{\text{SNE}} = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

  This function measures the mismatch between the high-dimensional and low-dimensional probabilities, aiming to preserve local structures in the reduced space.

- **t-SNE Loss Function:** t-SNE also uses the Kullback-Leibler divergence, but with symmetrized probabilities:

$$C_{\text{t-SNE}} = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

  This function aims to minimize the difference between high-dimensional and low-dimensional representations, focusing on local neighborhood structures.

- **UMAP Loss Function:** UMAP utilizes a cross-entropy loss function:

$$C_{\text{UMAP}} = \sum_{ij} w_{ij} \log(\sigma(d_{ij})) + (1 - w_{ij}) \log(1 - \sigma(d_{ij}))$$

  Variables: $\sigma(d_{ij})$, Logistic sigmoid function of the distance between points i and j; w_{ij}, Weight derived from the high-dimensional graph. UMAP's loss function balances attractive and repulsive forces, aiming to preserve both local and global data structures.

3.3 Linear Discriminant Analysis (LDA)

- **Primary Goal of LDA:**
  - Identify a linear combination of features.
  - Differentiate or segregate multiple classes of objects or events.
- **Optimization Approach:**
  - Optimize the ratio of determinants between the between-class scatter matrix (S_B) and the within-class scatter matrix (S_W).
  - Calculate mean vectors for each class:

    $m_i = \frac{1}{n_i} \sum_{x \in D_i} x,$

    where m_i is the mean vector for class i, n_i is the number of samples in class i, and D_i is the set of data points in class i.
  - Between-Class Scatter Matrix:

    $S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T,$

    where m_i is the mean vector for class i, m is the overall mean vector, N_i is the number of samples in class i, and c is the total number of classes.
  - Within-Class Scatter Matrix:

    $S_W = \sum_{i=1}^{c} \sum_{x \in D_i} (x - m_i)(x - m_i)^T.$
  - Fisher's Criterion $J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}$, where W is the projection matrix.
- **Solution and Eigenvalue Problem:**
  - Maximize $J(W)$ by solving the eigenvalue problem $S_W^{-1} S_B v = \lambda v$, where v are the eigenvectors, λ are the eigenvalues, S_B is the between-class scatter matrix, and S_W is the within-class scatter matrix.
  - Focus on the eigenvectors corresponding to the largest eigenvalues for maximum variance between classes.
- **Projection of Data:**
  - Determine W for data projection, where W contains the selected eigenvectors.
  - Project data points x into a lower-dimensional space to achieve optimal class separation: $y = W^T x$, where y is the projected data and x is the original data.

3.4 Sparse Dictionary Learning Overview

1. **Objective**:
   - Goal: Find a dictionary D and a sparse representation X to approximate a given dataset Y as $Y \approx DX$.
   - Components:
     - Y: A matrix where each column represents a data sample.
     - D: The dictionary matrix with each column as a dictionary atom.
     - X: A sparse matrix where each column is the sparse representation of the corresponding column in Y.
2. **Mathematical Formulation**:
   - $$\min_{D,X} \frac{1}{2}\|Y - DX\|_F^2$$
     subject to $\|x_i\|_0 \leq T \ \forall i$.
   - Elements:
     - $\|\cdot\|_F$: Frobenius norm, measuring the difference between Y and DX.
     - $\|x_i\|_0$: l_0-norm of the i-th column of X, counting non-zero entries to enforce sparsity.
     - T: A threshold dictating the maximum number of non-zero entries in each column of X.
3. **Optimization Challenge**:
   - Problem Nature: Generally non-convex and NP-hard due to the l_0-norm constraint.
   - Practical Approaches:
     - Relaxing the l_0-norm to an l_1-norm, promoting sparsity while being convex.
     - Using greedy algorithms like Orthogonal Matching Pursuit (OMP).
4. **Alternate Minimization**:
   - Strategy:
     - Fix D and optimize X: For each column y_i of Y, find the sparse representation x_i using D.
     - Fix X and optimize D: Update D while keeping X fixed.
5. **Regularization and Constraints**:
   - In Practice:
     - Adding constraints like normalizing the columns of D to prevent scaling issues.
     - Incorporating regularization terms to control overfitting.

3.5 Non-negative Matrix Factorization(NMF)

Non-negative Matrix Factorization (NMF) is a powerful technique in data analysis and linear algebra. It aims to factorize a non-negative matrix V into two non-negative matrices W and H, where k is the desired rank or number of components. This factorization is useful for various applications, including dimensionality reduction, feature extraction, and source separation.

Key Points about NMF

- **NMF Objective**: NMF aims to factorize a non-negative matrix V $(m \times n)$ into two non-negative matrices W $(m \times k)$ and H $(k \times n)$, where k is the desired rank or number of components.

- **Factorization**:

$$V \approx WH$$

V : Original non-negative matrix.
W : Matrix of non-negative basis vectors.
H : Matrix of non-negative coefficients.

- **Cost Function**: The cost function typically used in NMF is the squared Euclidean distance (Frobenius norm) between V and WH:

$$\text{Cost} = ||V - WH||^2$$

where $||A||^2$ represents the Frobenius norm, the sum of the squares of all elements in matrix A.

- **Optimization**: NMF employs iterative optimization algorithms, like multiplicative update rules, to find optimal values for W and H. These rules are applied iteratively until convergence:

$$\text{For } W: \quad W_{\text{new}} = W \odot \left(\frac{V \odot H'}{WH \odot H'} \right)$$
$$\text{For } H: \quad H_{\text{new}} = H \odot \left(\frac{W' \odot V}{W' \odot WH} \right)$$

where $\odot$ represents element-wise multiplication, $'$ represents matrix transpose, and / represents element-wise division.

- **Non-Negativity Constraint**: The non-negativity constraint ensures that all elements in W and H are non-negative, making NMF suitable for data where negative values lack meaningful interpretations, such as images, text, and audio.

- **Applications**: NMF is applied in various domains, including image processing, text mining, audio source separation, and dimensionality reduction, to extract interpretable components from data.

- **Interpretability**: NMF provides interpretable factors in the form of non-negative basis vectors (W) and coefficients (H), making it valuable for feature extraction and dimensionality reduction tasks.

- **Sparsity**: Regularization techniques, like L1 regularization (similar to Lasso), can be added to promote sparsity in the factor matrices, leading to non-negative sparse coding.

3.6 Multidimensional Scaling (MDS)

- **Similarity/Dissimilarity Matrix (D):**
  - The matrix D contains elements d_{ij}, each representing the distance or dissimilarity between objects i and j.
  - It's typically symmetric, with zeros on the diagonal (indicating zero dissimilarity of an object with itself).
- **Distance Matrix in Low-Dimensional Space (X):**
  - To find a set of points in a low-dimensional space (2D or 3D) that represent the objects.
  - The matrix X contains elements x_{ij}, each representing the distance between points i and j in the low-dimensional space.
- **Stress Function:**
  - Measures the goodness of fit between the distances x_{ij} in the low-dimensional space and the original dissimilarities d_{ij}.
  - Stress $= \sqrt{\frac{\sum_{i<j}(d_{ij}-x_{ij})^2}{\sum_{i<j} d_{ij}^2}}$, where $\sum_{i<j}$ sums over all unique pairs of points.
- **Optimization:**
  - Iteratively adjust the coordinates of points in the low-dimensional space to minimize the stress function.
  - Often done using numerical optimization techniques, such as gradient descent.
- **Result Interpretation:**
  - The configuration of points reflects the relative similarities or dissimilarities among the objects.
  - Objects more similar are closer together in the MDS space, while less similar objects are farther apart.
  - Dimension interpretation is not straightforward and often requires domain-specific knowledge.
- **Non-Uniqueness:**
  - MDS does not provide a unique solution; different starting configurations can lead to different final configurations with similar stress values.

3.7 Isomap (Isometric Mapping)

Isomap (Isometric Mapping) is a nonlinear dimensionality reduction method designed to uncover the underlying manifold structure in a high-dimensional dataset by approximating the geodesic distances among points.

Neighborhood Graph Construction:

- Construct a neighborhood graph G. Each point x_i in the dataset is connected to its K nearest neighbors, or to all points within a fixed radius ϵ.
- The distance between connected points is typically the Euclidean distance: $d(x_i, x_j) = \|x_i - x_j\|$.

Shortest Path Calculation:

- Compute the shortest path distances between all pairs of points in the graph G using algorithms like Floyd-Warshall or Dijkstra's.
- Let $D_G(x_i, x_j)$ denote the shortest path distance between x_i and x_j in the graph.

Constructing the Distance Matrix:

- Construct a matrix D where each element D_{ij} represents the graph distance $D_G(x_i, x_j)$.

Double Centering and Eigenvalue Decomposition:

- Perform double centering on D to create a matrix B:

$$B = -\frac{1}{2} H D^2 H$$

  where H is the centering matrix $H = I - \frac{1}{n} 1 1^T$, I is the identity matrix, and 11 is a vector of ones.

- B is then subjected to eigenvalue decomposition:

$$B = V \Lambda V^T$$

  where Λ is the diagonal matrix of eigenvalues and V contains the corresponding eigenvectors.

Embedding into Lower-dimensional Space:

- Select the top d eigenvectors (corresponding to the largest d eigenvalues) to form the matrix V_d.
- The coordinates of the data in the lower-dimensional space are then given by:

$$Y = \Lambda_d^{\frac{1}{2}} V_d^T$$

  where Λ_d contains the top d eigenvalues.

Clustering

4.1 K-Means

The K-means algorithm is a widely used method in unsupervised machine learning for clustering data. It partitions a dataset into K distinct, non-overlapping subgroups or clusters.

1. **Initialization**:
   - Choose k initial centroids randomly.
   - These centroids $C = \{c_1, c_2, ..., c_k\}$ are the starting points for each of the clusters.
2. **Assignment Step**:
   - Assign each data point x_i to the nearest centroid.
   - The assignment of a data point to a cluster is based on the minimum distance from the centroids, typically calculated using the Euclidean distance.
   - The assignment function is represented as:

$$S_i = \{x_p : \|x_p - c_i\| \leq \|x_p - c_j\| \, \forall j, 1 \leq j \leq k\}$$

   - Here, S_i is the set of points assigned to the i-th cluster, x_p is a data point, and $\|x_p - c_i\|$ is the Euclidean distance between x_p and centroid c_i.
3. **Update Step**:
   - Recalculate the centroids of the clusters based on the current assignment of data points.
   - The new centroid c_i for cluster i is the mean of all points assigned to that cluster:

$$c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$$

   - Here, $|S_i|$ is the number of data points in cluster i, and x_j are the data points in cluster i.
4. **Convergence Check**:
   - Repeat the assignment and update steps until the centroids no longer change significantly, or a maximum number of iterations is reached.
   - This convergence is often checked by seeing if the sum of the squared distances between data points and their corresponding centroids is minimized.

4.1.1 Elbow Method for Determining Optimal Number of Clusters in K-means

Calculate Within-Cluster Sum of Squares (WSS) for Various k

Perform K-means clustering for each value of k (the number of clusters).

Plot the WSS Values

Create a plot with the number of clusters k on the x-axis and the corresponding WSS on the y-axis.

Identify the Elbow Point

The "elbow" is the point in the plot where the WSS starts to decrease at a slower rate. It is visually identified as a point of inflection on the curve. As k increases, the average distortion per cluster decreases because the clusters are smaller. However, beyond a certain k (the elbow point), this decrease starts to diminish.

Choose k at the Elbow Point

The optimal number of clusters k is chosen at this elbow point. This choice represents a balance between maximizing the number of clusters (to reduce WSS) and keeping the model simple and generalizable (by not having too many clusters).

4.1.2 Silhouette Analysis for Determining Optimal Clusters in K-means

1. **Calculate the Silhouette Coefficient for Each Data Point**:
   - **Compute** $a(i)$:
     - $$a(i) = \frac{1}{|S_i| - 1} \sum_{x \in S_i, x \neq i} \|x - i\|$$
     - Average distance from the i-th data point to all other points in the same cluster S_i.
   - **Compute** $b(i)$:
     - $$b(i) = \min_{j \neq i} \left(\frac{1}{|S_j|} \sum_{x \in S_j} \|x - i\| \right)$$
     - Smallest average distance from the i-th data point to all points in any other cluster, excluding the one to which i belongs.
2. **Compute the Silhouette Coefficient for Each Point**:
   - **Silhouette Coefficient**:
     $$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$
   - Measures how similar a data point is within its own cluster compared to other clusters. Ranges from -1 to 1.
3. **Interpret the Results**:
   - **High Value**: Indicates good matching within its own cluster and poor matching to neighboring clusters.
   - **Low/Negative Value**: Suggests incorrect clustering or too many/few clusters.

- **Silhouette Coefficient for a Single Data Point:**
  - The silhouette coefficient for a single data point is a measure of how similar that point is to points in its own cluster compared to points in other clusters.
  - This coefficient helps determine the appropriateness of the clustering.
- **Mean Silhouette Coefficient as a Quality Measure:**
  - The mean of the silhouette coefficient for all points is a measure used to evaluate the quality of clustering in a dataset.
  - It provides insight into how well each object lies within its cluster.
- **Importance:**
  - Helps in assessing the separation distance between clusters.
  - Distinct clusters lead to better definitions and a higher average silhouette score.

Gap Statistics for Determining Optimal Clusters in K-means

1. **Cluster the Data and Compute Within-Cluster Dispersion**:
   - For each k, perform K-means clustering and calculate the WSS.
   - **WSS**:
   $$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r$$
   Where D_r is the sum of pairwise distances for all points in cluster r, and n_r is the number of points in cluster r.
2. **Generate Reference Data Sets**:
   - Generate B reference datasets with a random uniform distribution.
   - Each dataset should match the original in terms of number of observations and features.
3. **Compute the Expected Dispersion for Reference Data**:
   - Apply K-means clustering to each reference dataset for different k and compute the WSS.
   - **Expected Dispersion**:
   $$\mathrm{E}^* \log(W_k) = \frac{1}{B} \sum_{b=1}^{B} \log(W_k^b)$$
   Where W_k^b is the WSS for the b-th reference dataset.
4. **Calculate the Gap Statistic**:
   - **Gap Statistic**:
   $$\mathrm{Gap}(k) = \mathrm{E}^* \log(W_k) - \log(W_k)$$
   The Gap Statistic measures the difference between the expected dispersion and the observed dispersion.
5. **Choose Optimal k**:
   - Select the k where the Gap Statistic reaches its maximum.
   - Alternatively, choose the smallest k where Gap Statistic is within one standard deviation of the Gap Statistic at $k+1$.

Importance of Gap Statistics

- Provides an objective approach to determine the number of clusters.
- Compares clustering results against a random uniform distribution to identify significant clustering structures.

4.2 The Canopy Method

The Canopy Method is a pre-clustering method used in data mining for speeding up clustering operations on large data sets. It involves creating 'canopies' or rough groupings, followed by more precise clustering algorithms like K-means. This method is particularly effective for large datasets as it reduces computational costs by limiting the number of distance calculations.

Steps

1. **Select Distance Metrics**:
   - Choose distance metrics suitable for your data, such as Euclidean, Manhattan, or Cosine distance.
2. **Set Thresholds $T1$ and $T2$**:
   - Define two distance thresholds, $T1$ and $T2$ (with $T1 > T2$).
   - These thresholds determine how close points must be to form and belong to a canopy.
3. **Create Canopies**:
   - Randomly select a data point as a canopy center.
   - **Canopy Formation Rule**:
     - Include any point within distance $T1$ of the center in the canopy.
     - Remove any point within distance $T2$ from the dataset to prevent it from being a future canopy center.
4. **Repeat the Process**:
   - Continue until all points are either in a canopy or removed from the dataset.
5. **Use Canopies for Further Clustering**:
   - Apply a more precise clustering algorithm, like K-means, to each canopy.

- **Distance Calculation**:
  - The distance depends on the chosen metric. For Euclidean distance between two points x and y:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

    Where n is the number of dimensions, and x_i, y_i are the coordinates of x, y in the i-th dimension.
- **Threshold Application**:
  - For a canopy center c and a data point p:
    - Include p in the canopy if $d(c, p) < T1$.
    - Remove p from the dataset if $d(c, p) < T2$.

Importance

- **Efficiency**: Reduces computational costs for clustering large datasets.
- **Scalability**: Suitable for datasets too large for algorithms like K-means.
- **Flexibility**: Compatible with various distance metrics and clustering algorithms.

4.3 Gaussian Mixture Models (GMMs)

Gaussian Mixture Models are a probabilistic model used to represent normally distributed subpopulations within an overall population, often used in clustering.

1. Gaussian (Normal) Distribution

A Gaussian distribution is defined by its mean (μ) and variance (σ^2). Its probability density function (PDF) for a single variable is:

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

For a multivariate context, the PDF becomes:

$$f(\mathbf{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

where k is the dimensionality of the data.

2. Mixture of Gaussians

A Gaussian Mixture Model is a weighted sum of M Gaussian distributions:

$$p(\mathbf{x}) = \sum_{i=1}^{M} \pi_i f(\mathbf{x}|\boldsymbol{\mu}_i,\Sigma_i)$$

where π_i are the mixing coefficients, and $f(\mathbf{x}|\boldsymbol{\mu}_i,\Sigma_i)$ is the PDF of the i-th Gaussian component.

3. Expectation-Maximization (EM) Algorithm

The EM algorithm for GMMs involves the following steps:

E-step: Compute responsibilities:

$$\gamma(z_{ik}) = \frac{\pi_k f(\mathbf{x}_i|\boldsymbol{\mu}_k,\Sigma_k)}{\sum_{j=1}^{M}\pi_j f(\mathbf{x}_i|\boldsymbol{\mu}_j,\Sigma_j)}$$

M-step: Update the parameters:

$$\pi_k^{new} = \frac{1}{N}\sum_{i=1}^{N}\gamma(z_{ik}) \tag{4.1}$$

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{i=1}^{N}\gamma(z_{ik})\mathbf{x}_i}{\sum_{i=1}^{N}\gamma(z_{ik})} \tag{4.2}$$

$$\Sigma_k^{new} = \frac{\sum_{i=1}^{N}\gamma(z_{ik})(\mathbf{x}_i-\boldsymbol{\mu}_k^{new})(\mathbf{x}_i-\boldsymbol{\mu}_k^{new})^T}{\sum_{i=1}^{N}\gamma(z_{ik})} \tag{4.3}$$

4.3.1 Challenges in Gaussian Mixture Models (GMM)

Choosing the Number of Components

Difficulty in determining the optimal number of Gaussian components. Techniques like BIC, AIC, or cross-validation may not always provide clear guidance.

Sensitivity to Initialization

Results can vary significantly based on the initial choice of parameters. Poor initialization can lead to suboptimal clustering solutions.

Convergence to Local Optima

The EM algorithm may converge to local rather than global optima, resulting in finding suboptimal solutions for the GMM.

Model Complexity

Increasing the number of components adds complexity to the model. A high-dimensional parameter space can be difficult to optimize and interpret.

Assumption of Gaussian Components

Assumes each cluster follows a Gaussian distribution, which may not be suitable for data that does not fit this assumption.

Covariance Structure

Choosing the right covariance structure (spherical, diagonal, tied, or full) is challenging and affects both the model's flexibility and computational complexity.

High-Dimensional Data

Performance can degrade in high-dimensional spaces due to sparsity, making it difficult to accurately estimate parameters.

Overfitting

There's a risk of overfitting, especially with a large number of components or overly complex models. Requires careful model validation and possibly regularization.

Computational Complexity

The EM algorithm can be computationally intensive, especially for large datasets and complex models.

Interpretability

Higher model complexity can lead to challenges in interpreting and explaining the model.

4.3.2 Comparison of GMMs and K-means

Overlapping Clusters

GMMs are better suited to handling overlapping clusters than K-means.

Non-Spherical Clusters

GMMs are better equipped to handle clusters that are not spherical in shape than K-means.

Number of Clusters

GMMs can estimate the number of clusters in the data using model selection techniques such as the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC), while K-means requires the user to specify the number of clusters a priori.

Shape and Variances

GMMs can model clusters with different shapes and variances, while K-means assumes that the variance of the data within each cluster is the same and that the clusters are spherical in shape.

Computational Demand

GMMs are more computationally intensive than K-means.

Assumption Limitations

GMMs presume Gaussian distribution in each cluster, a condition not always met in datasets, unlike K-means.

4.4 DBSCAN: Density-Based Spatial Clustering of Applications with Noise

DBSCAN is a popular clustering algorithm used in data analysis, particularly effective for identifying clusters of varying shapes in a dataset with noise (i.e., outliers).

Core Concepts

- ε (Epsilon): The radius of a neighborhood around a given point.
- MinPts: The minimum number of points required to form a dense region.
- Epsilon-Neighborhood of a Point: $N_\varepsilon(p) = \{q \in D \mid \text{dist}(p, q) \leq \varepsilon\}$. Here, $N_\varepsilon(p)$ represents the ε-neighborhood of a point p, consisting of all points q within the dataset D that are within a distance ε from p.
- Core Point: A point p is a core point if its ε-neighborhood contains at least MinPts, i.e., $|N_\varepsilon(p)| \geq \text{MinPts}$.
- Border Point: A point p is a border point if its ε-neighborhood contains fewer than MinPts but p is within the ε-neighborhood of a core point.
- Noise Point: A point p is a noise point if it is neither a core point nor a border point.

The Algorithm

1. Start: Pick an arbitrary point p from the dataset.
2. Core Point Check: Check if p is a core point. If yes, create a new cluster and add all points in $N_\varepsilon(p)$ to this cluster.
3. Expand Cluster: For each point q in the cluster, if q is a core point, add its ε-neighborhood to the cluster.
4. Iterate: Repeat steps 2 and 3 for each point in the dataset until all points are either assigned to a cluster or marked as noise.
5. Result: The output is a set of clusters with core and border points, and a set of noise points.

Key Properties

- DBSCAN does not require specifying the number of clusters in advance.
- It can discover clusters of arbitrary shape.
- Points not belonging to any cluster are treated as noise, making DBSCAN robust to outliers.
- DBSCAN is an effective clustering method for spatial data analysis and is widely used in various fields like astronomy, geospatial analysis, and bioinformatics.

4.5 OPTICS(Ordering Points To Identify the Clustering Structure) Algorithm

Core Concepts

1. **Core Distance:** For a point p in the dataset, the core distance is the smallest distance such that p is a core point with respect to ε and MinPts.

$$\text{Core-Distance}_{\varepsilon,\text{MinPts}}(p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < \text{MinPts}, \\ \text{dist}(p,q) & \text{otherwise}, \end{cases}$$

   where q is the MinPts-th nearest neighbor of p within ε.

2. **Reachability Distance:** The reachability distance of a point p from a point o is the maximum of the core distance of o and the Euclidean distance between o and p.

$$\text{Reachability-Distance}_{\varepsilon,\text{MinPts}}(o,p) = \max(\text{Core-Distance}_{\varepsilon,\text{MinPts}}(o), \text{dist}(o,p))$$

   for o, p in the dataset, where $\text{dist}(o,p)$ is the Euclidean distance between o and p.

The Algorithm

1. **Start:** Pick an unprocessed point p from the dataset.
2. **Retrieve Neighbors:** Find the ε-neighborhood of p and calculate the core distance for p.
3. **Ordering Points:** If p is a core point, update the reachability distances of its neighbors and process them in increasing order of their reachability distance, recursively.
4. **Create Reachability Plot:** The reachability distances are plotted for all points, creating a reachability plot. This plot represents the clustering structure of the data.
5. **Extract Clusters:** Clusters can be extracted from the reachability plot based on a threshold value for reachability distance, which can be user-defined or automatically determined.

Key Properties

- OPTICS does not produce explicit clusters; instead, it creates an ordering of the data points representing its density-based clustering structure.
- It handles varying densities better than DBSCAN.
- It is particularly useful for datasets where clusters of different densities exist, and the noise level is high.

4.6 Spectral Clustering Algorithms

Spectral clustering algorithms can be broken down into the following key steps:

1. **Similarity Graph Construction**
   - Create a similarity graph where each node represents a data point.
   - Define edges based on the similarity, often using a Gaussian similarity function:

     $$S(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

     where x_i, x_j are data points and σ is a scaling parameter.
2. **Graph Laplacian**
   - Construct a weighted adjacency matrix W, where W_{ij} indicates the similarity between nodes i and j.
   - Create a degree matrix D, a diagonal matrix with D_{ii} being the sum of the weights of the edges connected to node i:

     $$D_{ii} = \sum_j W_{ij}$$

   - Define the graph Laplacian L as:

     $$L = D - W$$
3. **Eigenvalue Decomposition**
   - Perform eigenvalue decomposition on the Laplacian matrix L.
   - Extract eigenvalues and their corresponding eigenvectors.
4. **Forming the Feature Vector**
   - Construct a matrix U using the k eigenvectors corresponding to the k smallest eigenvalues.
   - U becomes an $n \times k$ matrix, where n is the number of data points.
5. **Clustering**
   - Treat each row of U as a point in R^k.
   - Use a standard clustering algorithm like k-means to cluster these points.
   - The resulting clusters correspond to the clusters in the original dataset.

4.7 Markov Chain Clustering (MCL)

Markov Chain Clustering (MCC), specifically the Markov Cluster Algorithm (MCL), is a process for finding clusters (i.e., groups of related items) in graphs. It's based on the idea of random walks on the graph, which can be described using Markov chains. Here's a basic overview of how the algorithm works, including the key equations involved:

1. Representation of Graph

A graph is represented by its adjacency matrix A. In this matrix, the element A_{ij} represents the weight of the edge from node i to node j. If there is no edge, $A_{ij} = 0$.

2. Normalization

The adjacency matrix is converted into a stochastic matrix M by normalizing each row to sum to 1. This is done by dividing each element by the sum of its row:

$$M_{ij} = \frac{A_{ij}}{\sum_k A_{ik}}$$

This normalization ensures that each row of M represents a probability distribution, consistent with the idea of a Markov chain, where the transition from one node to another is based on probabilities.

3. Random Walk Simulation

The core idea of MCL is to simulate random walks on the graph. This is done by repeatedly multiplying the matrix M by itself:

$$M^{(2)} = M \times M, \quad M^{(3)} = M^{(2)} \times M, \quad \ldots$$

Repeated multiplication corresponds to taking longer and longer random walks on the graph.

4. Expansion and Inflation

Two key operations are performed during each iteration:

- **Expansion (E):** This involves raising the matrix to a power (matrix multiplication), which corresponds to the random walk process. The expansion step tends to spread out the probability mass, which can lead to flow between different regions of the graph.

  $$M^{(\text{expanded})} = M^n$$

  Here, n is usually 2, but can be adjusted.

- **Inflation (I):** After expansion, the inflation step is applied. This involves raising each element of the matrix to a power r (inflation factor) and then re-normalizing the rows to sum to 1. Inflation strengthens intra-cluster probabilities and weakens inter-cluster probabilities.

  $$M_{ij}^{(\text{inflated})} = \frac{(M_{ij}^{(\text{expanded})})^r}{\sum_k (M_{ik}^{(\text{expanded})})^r}$$

  The inflation parameter r is crucial; a higher value strengthens the contrast between strong and weak connections, leading to more distinct clusters.

5. Convergence

The process of expansion and inflation is repeated until the matrix M converges, i.e., it no longer changes significantly with further iterations. The final matrix reveals the clusters: nodes that end up in the same row with high probabilities are considered to be in the same cluster.

4.8 Agglomerative Clustering Algorithm

Initialization

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$, where each x_i is a data point. Initially, each data point x_i is considered as a separate cluster C_i, thus having n clusters.

Finding the Closest Pair

Calculate the distances between every pair of clusters and identify the pair with the minimum distance.

Distance Computation

Compute the distance between two clusters, C_i and C_j, using one of the following distance metrics:

- **Single-linkage**:

$$d_{\text{single}}(C_i, C_j) = \min\{||a - b|| : a \in C_i, b \in C_j\}$$

- **Average-linkage**:

$$d_{\text{average}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{a \in C_i} \sum_{b \in C_j} ||a - b||$$

- **Complete-linkage**:

$$d_{\text{complete}}(C_i, C_j) = \max\{||a - b|| : a \in C_i, b \in C_j\}$$

- **Centroid-linkage**:

$$d_{\text{centroid}}(C_i, C_j) = ||\mu_{C_i} - \mu_{C_j}||$$

  where μ_{C_i} and μ_{C_j} are the centroids of clusters C_i and C_j, respectively.

- **Ward's Method**:

$$d_{\text{Ward}}(A, B) = \sum_{x \in A \cup B} ||x - \mu_C||^2 - \left(\sum_{x \in A} ||x - \mu_A||^2 + \sum_{x \in B} ||x - \mu_B||^2 \right)$$

  where μ_A, μ_B, and μ_C are the centroids of clusters A, B, and the merged cluster C, respectively.

Merging

Identify and merge the two clusters C_i and C_j that have the smallest distance between them as calculated by the chosen metric.

Update Distance Matrix

After merging, the distance matrix is updated. If a new cluster C_k is formed from C_i and C_j, update its distances to all other clusters.

Repeat

Repeat steps 2-5 until only one cluster remains or until a single cluster reached.

Cut the Dendrogram

The dendrogram, a tree-like diagram representing the series of merges, can be cut at different levels to obtain a specific number of clusters.

Supervised Machine Learning

5.1 Linear Regression Model

Linear regression is a statistical technique used to model and analyze the relationship between a dependent variable and one or more independent variables by fitting a straight line to the data.

5.1.1 Assumptions Of Linear Regression

Linearity

The relationship between the independent and dependent variables is linear, meaning changes in the independent variables result in proportional changes in the dependent variable.

Independence

Each observation in the dataset is independent of the others, ensuring that the value of one observation doesn't influence or depend on another.

Homoscedasticity

The variance of the error terms (residuals) is constant across all levels of the independent variables, indicating uniform dispersion of residuals.

Normal Distribution of Residuals

The residuals (differences between observed and predicted values) are normally distributed, which is especially crucial for small sample sizes.

No or Little Multicollinearity

There is minimal or no multicollinearity, meaning that the independent variables are not highly correlated with each other, to avoid inflated variances in coefficient estimates.

No Auto-correlation

In the residuals, there is an absence of auto-correlation, particularly important in time series data, where one time period's errors shouldn't influence another's.

Fixed Independent Variables

The independent variables are assumed to be measured without significant error, implying that any measurement error is negligible.

5.2 Ordinary Least Squares (OLS)

Least squares is a mathematical method used to minimize the sum of squared differences between observed data points and model predictions.

Ordinary least squares (OLS) is a specific type of least squares method used in linear regression. It finds the best-fitting linear equation by minimizing the sum of squared errors between the observed values and the values predicted by the linear model.

Basic Model

The linear regression model is expressed as:

$$y_i = x_i'\beta + u_i$$

Where y_i is the outcome variable, x_i is a vector of regressor variables, β is the coefficient vector, and u_i is the error term.

Assumptions

- Linearity: The relationship between regressors and the dependent variable is linear.
- Conditional Independence: $E(U|X) = 0$, the expectation of the error term, given the regressors, is zero.
- No Multi-collinearity: The matrix X has full rank k, indicating no perfect collinearity among regressors.
- Homoskedasticity: $\text{Var}(U|X) = \sigma^2 I_n$, the variance of the error term is constant.

Objective

Minimize the sum of squared errors:

$$Q = \sum_{i=1}^{n} (y_i - x_i'\beta)^2$$

Solution

Derive $\hat{\beta}$ by setting the derivative of Q with respect to β to zero:

- Differentiate Q: $-2X'Y + 2X'X\beta = 0$
- Solve for β: $X'X\beta = X'Y$
- Obtain $\hat{\beta}$: $\hat{\beta} = (X'X)^{-1}X'Y$

5.2.1 OLS as Projection

The Ordinary Least Squares (OLS) method projects the outcome variable y onto the space spanned by the regressors X, analogous to Ax in linear algebra.

Orthogonality Principle

The residual $(y - X\beta)$ is orthogonal to the span of X, and lies in the left nullspace of X.

Deriving the OLS Estimator β

From orthogonality, we have:

$$X'(y - X\beta) = 0$$

which leads to:

$$X'X\beta = X'y$$

Solving for β gives:

$$\beta = (X'X)^{-1}X'y$$

Projection Matrices

The orthogonal projection matrix is defined as:

$$P_x = X(X'X)^{-1}X'$$

The residuals are expressed as:

$$\hat{u} = y - X\beta = y - P_x y = (I_n - P_x)y = M_x y$$

where M_x projects onto the space orthogonal to the span of X.

5.2.2 Applying SVD to OLS and Ridge Regression

Standard OLS Formula

The standard OLS formula to estimate the fitted values $\hat{\beta}$ is:

$$X\hat{\beta} = X(X'X)^{-1}X'y$$

Applying SVD to OLS

By substituting the data input matrix X with its SVD components UDV', the formula becomes:

$$X\hat{\beta} = UDV'(VD^2V')^{-1}VD'U'y$$

Simplifying further:

$$X\hat{\beta} = UD(D^2)^{-1}DU'y = UU'y$$

In this revised formula, D is a diagonal matrix with the square root of the eigenvalues, and U and V are orthonormal matrices. This transformation shows that the fitted values in OLS regression can be computed with respect to the orthonormal basis U.

Ridge Regression Formula

Ridge regression modifies the OLS regression by adding a penalty term to the size of the coefficients. The objective is to minimize the penalized sum of squares. The solution to ridge regression is given by:

$$\hat{\beta}_{ridge} = (X'X + \lambda I_k)^{-1}X'Y$$

Applying SVD to Ridge Regression

Substituting the SVD formula into the ridge regression formula, the fitted values become:

$$X\hat{\beta}_{ridge} = UD(D^2 + \lambda I)^{-1}DU'y$$

This results in a clearer understanding of regularization. The predicted values are shrunk by the factor $\frac{d_j^2}{d_j^2+\lambda}$. Greater shrinkage is applied to variables explaining a lower fraction of the variance, in line with the principles of Principal Component Analysis (PCA) and ridge regression. Ridge regression applies a weighted shrinkage method, as opposed to PCA, which truncates variables below a certain threshold.

5.2.3 Relationship between CEF and Regression

1. The Conditional Expectation Function (CEF) is defined as $E[Y|X]$, representing the expected value of Y given X.

2. In regression, the dependent variable Y_i is decomposed as $Y_i = E[Y_i|X_i] + \epsilon_i$, where ϵ_i is the error term, orthogonal to X_i. The primary goal in regression is to find a function of X, say $m(X)$, that minimizes the squared mean error, $\min E[(Y_i - m(X_i))^2]$, where the optimal choice for $m(X)$ turns out to be the CEF.

3. In OLS regression, the aim is to linearly approximate the CEF. The regression equation $\beta = \arg\min_b E[(E[Y_i|X_i] - X_i'b)]$ indicates that minimizing the squared differences between Y_i and $X_i'b$ is equivalent to approximating the CEF linearly.

5.3 Method of Moments

Introduction

The Method of Moments is a statistical technique for estimating the parameters of a probability distribution or a model. This approach compares theoretical moments from a probability distribution, like mean and variance, with empirical moments derived from data.

Understanding Moments

- *Definition:* Moments are quantitative measures of a function's shape.
- *Types of Moments:*
  - *Raw Moments:* The nth raw moment of a random variable x, denoted as μ_n, is $E[x^n]$.
  - *Central Moments:* The nth central moment, denoted as μ'_n, is $E[(x-\mu)^n]$, where μ is the mean.

5.3.0.1 Method of Moments Estimator

- **Theoretical Moment:** Calculated for the entire population using its probability distribution, it represents population measures like mean or variance. An example is the theoretical mean $E[X]$, derived using the expectation operator.
- **Sample Moment:** An empirical measure calculated from a data sample, used to approximate theoretical moments. The sample mean $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ is a common example, where n is the sample size and X_i are the sample values.
- **Replacing theoretical moments with sample moments:** This substitution is necessary because theoretical moments often depend on unknown population parameters, while sample moments can be directly computed from the available data.
- **The principle:** With a sufficiently large and representative sample, the sample moments should be good approximations of the theoretical moments.

Example 1: Estimator for Sample Mean

- **Population Moment:** $\mu = E[X]$
- **Objective:** Find an estimator for the sample mean.
- **Process:**
  - Sample Analogue: Replace the expected value $E[X]$ with a sample mean $\bar{X}$.
  - Estimator for μ: $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}$

Example 2: Normal Distribution

- **Given:** $X_1, X_2, \ldots, X_n \sim N(\mu, \sigma^2)$
- **Objective:** Find estimators for the parameters μ and σ^2.
- **Process:**
  - **First Moment (Mean):** Population Moment: $E[X] = \mu$, Sample Analogue: $\bar{X}$, Estimator for μ: $\hat{\mu} = \bar{X}$.
  - **Second Moment (Variance):** Population Moment: $E[X^2] = \mu^2+\sigma^2$, Expand using μ's estimator, Sample Analogue: $\frac{1}{n}\sum_{i=1}^{n} X_i^2$, Estimator for σ^2: $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2 - \bar{X}^2$.

Example 3: Poisson Distribution

- **Given:** $X_1, X_2, \ldots, X_n \sim \text{Poisson}(\lambda)$
- **Objective:** Find estimators for the parameter λ.
- **Process:**
  - Equality in Poisson Distribution: Population Moment: $E[X] = \text{var}(X) = \lambda$.
  - Estimators for λ:
    - Estimator 1: $\hat{\lambda}_1 = \bar{X}$.
    - Estimator 2: $\hat{\lambda}_2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$.

  Given that we have two moment conditions but only one parameter to estimate, it's necessary to find a method to effectively 'merge' these conditions. Relying on just one of these conditions would result in underutilizing the available information.

5.3.1 General Framework of Moment Conditions

Moment Conditions

- Moment conditions in regression are expressed as a function $g(X_i, \beta)$.
- X_i represents the observed data, including dependent variables y_i, independent variables X_i, and any instruments Z_i.
- β is a vector of parameters to estimate, with a length of k.

Model Identification

- A model is identified if the solution for β is unique.
- Uniqueness is expressed as $E[g(X_i, \beta)] = 0$ and $E[g(X_i, \hat{\beta})] = 0$ implying $\beta = \hat{\beta}$.
- At least as many restrictions (moment conditions) as parameters (k) are needed to identify the model.

Application in OLS Regression

- OLS Moment Condition: $E[X_i U_i] = 0$ or $E[X_i(y_i - X_i'\beta)] = 0$, where U_i is the error term.
- This condition is used to solve for the OLS estimator $\hat{\beta}$.

Extension to More Complex Models (IV Regression)

- Instrumental Variables (IV) Regression: Useful when the model is overidentified (l instruments for k parameters).
- IV Estimator Formula: Derived by solving the moment condition, $\hat{\beta}_{IV} = (Z'X)^{-1}Z'y$.
- IV regression uses instruments Z_i to resolve issues in the OLS model.

5.3.2 Instrumental Variables in Regression Models

- Instrumental variables are used in statistical analysis to address endogeneity issues, such as omitted variables that affect both X and Y, where explanatory variables in a regression model are correlated with the error term.
- They provide a way to estimate causal relationships by using a variable (the instrument) that is correlated with the explanatory variable but not with the error term.

Example: To assess education's impact on income without bias from individual ability, use an instrumental variable like proximity to a university. This approach isolates the influence of education on income, separate from individual ability.

Moment Conditions with Instrumental Variables

- The moment conditions involving instrumental variables can be represented as $g(X_i, \beta) = Z_i(y_i - X_i'\beta)$ or $E[Z_iU_i] = 0$.
- Here, Z_i are the instrumental variables, y_i the dependent variables, X_i the independent variables, β the parameters, and U_i the error terms.
- The condition $E[Z_iU_i] = 0$ implies that the instruments are uncorrelated with the error term, a critical requirement for valid instrumental variables.

Solving the Moment Condition

- The moment condition for IV regression is given by $0 = \sum_{i=1}^{n} Z_i(y_i - X_i'\hat{\beta}_{IV})$.
- Solving this condition yields the IV estimator $\hat{\beta}_{IV} = (Z'X)^{-1}Z'y$.

Role of Instrumental Variables in Addressing Endogeneity

- IV regression substitutes problematic OLS moments with new ones that incorporate instruments, addressing the endogeneity bias.
- Instruments Z_i are selected for their correlation with endogenous independent variables and their lack of correlation with the error term.
- This substitution allows the IV estimator to isolate the variation in the explanatory variable that is unaffected by endogeneity.

Condition of Perfect Identification

- For a model to be perfectly identified, the number of instruments (l) should be equal to the number of parameters (k).
- Perfect identification ensures that there is just enough information to uniquely identify the model parameters.

Application in Complex Models

- The IV approach extends beyond simple linear models and can be applied to more complex regression models where standard OLS assumptions do not hold.
- It is particularly useful in models where endogeneity is a concern and the model's identification relies on the validity of the instruments.

5.3.3 Generalized Method of Moments (GMM)

General Concept of GMM

- **Overidentified Models:**
  - In scenarios where the number of restrictions (l) is greater than the number of parameters to estimate (k), i.e., $l > k$, the model is said to be overidentified.
  - In such cases, traditional methods like Ordinary Least Squares (OLS) or Instrumental Variable (IV) regression cannot be directly applied to estimate the parameter vector β.
- **Combining Restrictions:**
  - GMM seeks to find an estimate of β that brings the sample moments as close to zero as possible. This involves combining multiple moment conditions in an optimal way.
  - The moment conditions for all restrictions are still equal to zero, but the sample approximations may not be exactly zero due to finite sample sizes.

GMM Estimation

The GMM estimator, $\hat{\beta}_{GMM}$, is defined as the value of β that minimizes the weighted distance of $\sum_{i=1}^{n} g(X_i, \beta)$, where $g(X_i, \beta)$ is a vector of functions representing the moment conditions. The GMM estimation equation can be expressed as:

$$\hat{\beta}_{GMM} = \arg\min_{\beta \in B} \left(\sum_{i=1}^{n} g(X_i, \beta)' W \sum_{i=1}^{n} g(X_i, \beta) \right)$$

Where:

- W is an $l \times l$ matrix of weights used to select the ideal linear combination of instruments.
- The function $g(X_i, \beta)$ represents the moment conditions, which may involve observed data, endogenous variables, and instruments.

GMM Formula for Linear Regression Models

In the context of linear regression models that are overidentified, the general GMM formula is given by:

$$\hat{\beta}_{GMM} = \left((X'Z)W(Z'X)\right)^{-1} (X'Z)W(Z'y)$$

Here:

- X and Z represent the observed data and instruments, respectively.
- W is again the weighting matrix.
- X' and Z' are transposes of X and Z respectively.

Optimal Choice of Weighting Matrix

- The choice of the weighting matrix W is crucial in GMM. For instance, when $W = (Z'Z)^{-1}$, the GMM estimator becomes equivalent to the Instrumental Variable (IV) estimator.
- The optimal choice of W depends on the specifics of the model and the nature of the data.

5.4 Maximum Likelihood

5.4.1 Maximum Likelihood Estimation (MLE)

MLE is a statistical method used to estimate the parameters of a model, aiming to find the parameter values that make the observed data most probable.

Basic Idea

- Given data and a statistical model with parameters, MLE seeks to find the parameter values that maximize the likelihood of observing the data.

Likelihood Function

- The likelihood function $L(\theta)$ is a function of the parameters θ and represents the probability of the observed data given these parameters.
- For independent and identically distributed observations $X = \{x_1, x_2, \ldots, x_n\}$, the likelihood function is the product of the individual observations' PDFs or PMFs:

$$L(\theta) = f(x_1, x_2, \ldots, x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

Log-Likelihood

- The log-likelihood, $\ln L(\theta)$, transforms the product into a sum, facilitating differentiation:

$$\ln L(\theta) = \sum_{i=1}^{n} \ln f(x_i|\theta)$$

- Maximizing $\ln L(\theta)$ is equivalent to maximizing $L(\theta)$ as the logarithm is monotonic.

Maximizing the Log-Likelihood

- Find the parameter values that maximize the log-likelihood by differentiating $\ln L(\theta)$ with respect to θ and setting it to zero. Numerical methods may be required for complex cases:

$$\frac{d}{d\theta} \ln L(\theta) = 0$$

Example: Normal Distribution

- For observations assumed to be normally distributed with unknown mean μ and known variance σ^2, the log-likelihood is:

$$\ln L(\mu) = \sum_{i=1}^{n} \ln \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2} \right) \right]$$

- Maximizing this log-likelihood with respect to μ yields the MLE for the mean of a normal distribution.

5.4.2 OLS Estimator using Maximum Likelihood

Model Specification

Start with the linear regression model:

$$y_i = x_i'\beta + u_i$$

where y_i is the dependent variable, x_i is the vector of independent variables, β is the vector of coefficients, and u_i is the error term.

Assumption about Error Term

Assume that the error terms u_i, conditional on x_i, are normally distributed with mean 0 and unknown variance σ^2:

$$u_i|x_i \sim \mathcal{N}(0, \sigma^2)$$

Probability Density Function (PDF)

The PDF of a single observation is given by:

$$f(y_i, x_i; \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i'\beta)^2}{2\sigma^2}\right)$$

Likelihood Function

The likelihood (joint PDF) of observing all data is:

$$L(\beta, \sigma^2) = \prod_{i=1}^{n} f(y_i, x_i; \beta, \sigma^2)$$

Log-Likelihood Function

Convert the likelihood to log-likelihood for simplification:

$$\ln L(\beta, \sigma^2) = \sum_{i=1}^{n} \ln f(y_i, x_i; \beta, \sigma^2)$$

Maximizing Log-Likelihood

To find the maximum likelihood estimators, take the derivative of the log-likelihood function with respect to β and σ^2, and set them to zero.

Deriving the Estimators

Solving these equations will give you the maximum likelihood estimators for β and σ^2:

$$\hat{\beta}_{ML} = \left(\sum_{i=1}^{n} x_i x_i'\right)^{-1} \left(\sum_{i=1}^{n} x_i y_i\right)$$

$$\hat{\sigma}^2_{ML} = \frac{1}{n}\sum_{i=1}^{n} (y_i - x_i'\hat{\beta}_{ML})^2$$

Generalized Linear Models (GLMs)

Generalized Linear Models (GLMs) are an advanced form of linear regression models, characterized by their ability to handle a variety of response variable distributions and to establish a distinct relationship between response and predictor variables. The essence of GLMs lies in three core components:

1. **Random Component:** This defines the probability distribution of the response variable, Y. In GLMs, Y is assumed to follow a distribution from the exponential family, such as Normal, Binomial, or Poisson distributions.

2. **Systematic Component:** Represents the explanatory (independent) variables, $X_1, X_2, \ldots, X_n$, and their linear combination, often denoted as η. The equation for this component is:
$$\eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$
Here, $\beta_0, \beta_1, \ldots, \beta_n$ are the model's parameters (coefficients).

3. **Link Function:** Denoted as $g()$, this function connects the systematic component to the expected value of the response variable. It ensures that the model accommodates the distribution type of the response variable. The relationship is described as:
$$g(E(Y)) = \eta$$
where $E(Y)$ is the expected value of Y.

Full GLM

The GLM equation combines these components as:

$$E(Y) = g^{-1}(\eta) = g^{-1}(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)$$

Here, $g^{-1}()$ is the inverse of the link function, transforming the linear predictor η back to the response variable's scale.

Flexibility in Response Variable Distribution

GLMs are highly versatile, allowing for response variables from any member of the exponential family of distributions:

- **Normal Distribution:** Utilized in standard linear regression where the response variable can take any continuous value. In this simplest form of GLM, often equivalent to ordinary least squares regression, the model is:
$$Y = \beta_0 + \beta_1 X + \epsilon$$
where Y is continuous and normally distributed, and ϵ is the error term.

- **Binomial Distribution:** Employed in logistic regression, suitable for binary outcomes such as success/failure or yes/no. The logistic model predicts the probability of occurrence of an event and is expressed as:
$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$
where p represents the probability of one of the binary outcomes.

- **Poisson Distribution:** Applied in scenarios where the response variable represents count data, typically non-negative integers, such as the number of occurrences of an event. Poisson regression, used for modeling count data, is formulated as:

$$\log(E(Y)) = \beta_0 + \beta_1 X$$

- **Gamma Distribution:** Often used in situations where the response variable is positively skewed and continuous, such as for modeling time-to-event data (e.g., survival times). The Gamma distribution in GLMs typically uses the inverse or log link function. The model can be expressed as:

$$g(E(Y)) = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n$$

where $E(Y)$ is the expected value of the response variable, Y, which follows a Gamma distribution. The link function $g()$ could be the inverse link $g(E(Y)) = 1/E(Y)$ or the logarithmic link $g(E(Y)) = \log(E(Y))$.

- **Multinomial Distribution:** Used when the response variable can take on more than two categories, as in the case of multinomial logistic regression. This model is an extension of logistic regression to multiple categories. The Multinomial distribution in GLMs can be represented by a set of equations, one for each category. For a response variable with k categories, the model can be:

$$\log\left(\frac{p_i}{p_k}\right) = \beta_{i0} + \beta_{i1} X_1 + \ldots + \beta_{in} X_n$$

for $i = 1, 2, \ldots, k-1$, where p_i is the probability of the i^{th} category. The reference category k serves as the baseline, and the logit link is used.

Bernoulli Trial Estimator using Maximum Likelihood

Problem Setup

Consider a dataset consisting of results from a series of coin flips, where we aim to estimate the probability of the coin landing heads.

Bernoulli Distribution

Assume that each coin flip is an independent Bernoulli trial with probability p of landing heads.

Probability Mass Function (PMF)

The PMF for a single observation x_i (taking the value of 0 or 1) is given by:

$$f(x_i; p) = p^{x_i}(1-p)^{1-x_i}$$

Likelihood Function

The likelihood function for observing the entire dataset is the product of individual PMFs:

$$L(p) = \prod_{i=1}^{n} f(x_i; p)$$

Log-Likelihood Function

The log-likelihood function is:

$$\ln L(p) = \sum_{i=1}^{n} \ln f(x_i; p) = \sum_{i=1}^{n} (x_i \ln p + (1-x_i) \ln(1-p))$$

Maximizing Log-Likelihood

To find the maximum likelihood estimator for p, take the derivative of the log-likelihood function with respect to p and set it to zero.

Deriving the Estimator

Solving this equation gives the maximum likelihood estimator for p:

$$\hat{p}_{ML} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

which is the sample mean of the observed values.

5.5 Logistic Regression

Linear Regression

- Used for predicting a continuous outcome.
- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$
  - y: Dependent variable (the outcome to predict).
  - $\beta_0, \beta_1, \ldots, \beta_n$: Coefficients (weights) of the model.
  - $x_1, x_2, \ldots, x_n$: Independent variables (predictors).
  - ϵ: Error term.
- **Example Usage:** Predicting house prices based on features like size, location, number of bedrooms, etc.

Logistic Regression

- Used for binary classification (predicting a categorical outcome with two classes).
- $\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
  - p: Probability of the dependent variable being in one of the two classes.
  - Other terms are similar to linear regression.
- **Example Usage:** Predicting whether an email is spam or not based on characteristics like the frequency of certain words, sender, etc.

Assumption	**Linear Regression**	**Logistic Regression**
Nature of Dependent Variable	Assumes the dependent variable is continuous and normally distributed.	Assumes the dependent variable is categorical, typically binary.
Relationship Between Variables	Assumes a linear relationship between the dependent and independent variables.	Assumes a linear relationship between the log-odds (logit) of the dependent variable and the independent variables.
Distribution of Errors/Residuals	Assumes that the residuals (errors) are normally distributed.	Does not assume a normal distribution of residuals.
Homoscedasticity	Assumes homoscedasticity, meaning the variance around the regression line is the same for all values of the predictor variable.	This assumption is not applicable, as it deals with categorical outcomes.
Independence of Errors	Assumes no autocorrelation in the residuals. The errors are independent of each other.	Also assumes independence of errors (no autocorrelation).
Multicollinearity	Assumes little or no multicollinearity among the independent variables.	Similar to linear regression, it assumes little or no multicollinearity.
Sample Size	Can often work with smaller sample sizes, though the exact size depends on the number of predictors.	Typically requires a larger sample size, especially if the outcome is rare.

5.6 Generalized Linear Models (GLMs)

Generalized Linear Models (GLMs) are an advanced form of linear regression models, characterized by their ability to handle a variety of response variable distributions and to establish a distinct relationship between response and predictor variables. The essence of GLMs lies in three core components:

1. **Random Component:** This defines the probability distribution of the response variable, Y. In GLMs, Y is assumed to follow a distribution from the exponential family, such as Normal, Binomial, or Poisson distributions.

2. **Systematic Component:** Represents the explanatory (independent) variables, $X_1, X_2, \ldots, X_n$, and their linear combination, often denoted as η. The equation for this component is:

$$\eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

   Here, $\beta_0, \beta_1, \ldots, \beta_n$ are the model's parameters (coefficients).

3. **Link Function:** Denoted as $g()$, this function connects the systematic component to the expected value of the response variable. It ensures that the model accommodates the distribution type of the response variable. The relationship is described as:

$$g(E(Y)) = \eta$$

   where $E(Y)$ is the expected value of Y.

Full GLM

The GLM equation combines these components as:

$$E(Y) = g^{-1}(\eta) = g^{-1}(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)$$

Here, $g^{-1}()$ is the inverse of the link function, transforming the linear predictor η back to the response variable's scale.

Flexibility in Response Variable Distribution

GLMs are highly versatile, allowing for response variables from any member of the exponential family of distributions:

- **Normal Distribution:** Utilized in standard linear regression where the response variable can take any continuous value. In this simplest form of GLM, often equivalent to ordinary least squares regression, the model is:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

  where Y is continuous and normally distributed, and ϵ is the error term.

- **Binomial Distribution:** Employed in logistic regression, suitable for binary outcomes such as success/failure or yes/no. The logistic model predicts the probability of occurrence of an event and is expressed as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

  where p represents the probability of one of the binary outcomes.

- **Poisson Distribution:** Applied in scenarios where the response variable represents count data, typically non-negative integers, such as the number of occurrences of an event. Poisson regression, used for modeling count data, is formulated as:

$$\log(E(Y)) = \beta_0 + \beta_1 X$$

- **Gamma Distribution:** Often used in situations where the response variable is positively skewed and continuous, such as for modeling time-to-event data (e.g., survival times). The Gamma distribution in GLMs typically uses the inverse or log link function. The model can be expressed as:

$$g(E(Y)) = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n$$

  where $E(Y)$ is the expected value of the response variable, Y, which follows a Gamma distribution. The link function $g()$ could be the inverse link $g(E(Y)) = 1/E(Y)$ or the logarithmic link $g(E(Y)) = \log(E(Y))$.

- **Multinomial Distribution:** Used when the response variable can take on more than two categories, as in the case of multinomial logistic regression. This model is an extension of logistic regression to multiple categories. The Multinomial distribution in GLMs can be represented by a set of equations, one for each category. For a response variable with k categories, the model can be:

$$\log\left(\frac{p_i}{p_k}\right) = \beta_{i0} + \beta_{i1} X_1 + \ldots + \beta_{in} X_n$$

  for $i = 1, 2, \ldots, k-1$, where p_i is the probability of the i^{th} category. The reference category k serves as the baseline, and the logit link is used.

5.7 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are a class of supervised learning models that find a hyperplane in an N-dimensional space (N — number of features) that distinctly classifies data points. They aim to maximize the margin between data classes, with support vectors being the key data points nearest to the hyperplane. SVMs can efficiently perform a non-linear classification using the kernel trick, implicitly mapping inputs into high-dimensional feature spaces.

Representation of Data Points

- Each data point in an SVM is represented as a point in an n-dimensional space (where n is the number of features), with each feature being a particular coordinate. For a data point x_i, it can be represented in a D-dimensional space as a feature vector $x \in \mathbb{R}^D$.

Hyperplane

- The hyperplane equation, the decision boundary, is given by $w^T x + b = 0$ where w is the weight vector, b is the bias term, and x is the input feature vector.

Classification Decision Rule

- The decision function for classifying data points is based on the sign of $f(x) = w^T x + b$.
- If $f(x) > 0$, the data point is classified into one class (positive class); if $f(x) < 0$, into the other class (negative class).

Margin Maximization

- The SVM aims to maximize the margin, the distance between the hyperplane and the nearest data points of each class (support vectors), given by $\frac{2}{\|w\|}$.
- The optimization problem is to maximize $\frac{2}{\|w\|}$ subject to $y_i(w^T x_i + b) \geq 1$ for all i, where y_i is the label of x_i.

Kernel Trick

- For non-linearly separable data, SVMs use kernel functions to transform the data into a higher dimension for linear separability.
- The kernel function $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, where ϕ is the transformation function. Common kernels include Polynomial, RBF, and Sigmoid.

Handling Misclassifications (Soft Margin)

- Slack variables ξ_i allow misclassifications, modifying the optimization problem to minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$ subject to $y_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i, where C is the regularization parameter.

5.7.1 Lagrangian SVMs

The Lagrangian in SVMs combines the objective (minimizing $\frac{1}{2}\|w\|^2$) with the constraints (data points correctly classified):

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i + b) - 1]$$

Here, α_i are Lagrange multipliers for each constraint, ensuring each data point x_i is on the correct side of the margin.

Partial Derivatives

- Finding optimal values of w, b, and α involves taking partial derivatives of L and setting them to zero.

- **Derivative with Respect to** w This derivative leads to:

  $$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

  indicating the optimal weight vector w as a combination of the support vectors.

- **Derivative with Respect to** b Setting the derivative with respect to b to zero results in:

  $$\sum_{i=1}^{n} \alpha_i y_i = 0$$

  maintaining balance between the classes.

Dual Problem Formulation

- **Transformation to Dual Problem** By solving for w and substituting it back into the Lagrangian, we obtain the dual problem, which only involves the Lagrangian multipliers. This dual problem is easier to solve as it typically has fewer dimensions than the original problem α:

  $$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- The dual form is computationally efficient, especially for large feature spaces or kernel methods.

Incorporating Basis Functions

- **Basis Function Transformation** Basis functions $\phi(x)$ transform input data into a higher-dimensional space, facilitating separation of non-linearly separable data.

- **Kernel Function** The kernel function $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ computes inner products in the transformed space.

- **Modified Dual Lagrangian** The Lagrangian with kernel function:

  $$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

- **Optimization and Decision Function**

- **Optimization** Find α that maximizes the dual Lagrangian under constraints.

- **SVM Decision Function** The decision function for new inputs x:

  $$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b$$

  where b is determined using support vectors.

  This comprehensive process allows SVMs to efficiently classify both linearly and non-linearly separable data.

5.7.2 Comparison Between OLS and SVM

Comparing the approaches of Ordinary Least Squares (OLS) and Support Vector Machines (SVM) in various aspects.

Minimizing Coefficients

- **OLS:** Minimizes squared error to find coefficients that minimize the sum of squared residuals.
- **SVM:** Minimizes the $l2$-norm of the coefficient vector, seeking a sparse solution with many coefficients set to zero.

Error Term and Constraints

- **OLS:** Does not incorporate a margin for error, aiming to minimize squared residuals without a strict margin.
- **SVM:** Manages error using constraints with a predetermined margin (ϵ). Absolute errors are constrained to be less than or equal to ϵ, with deviations denoted as ξ.

Sensitivity to Outliers

- **OLS:** Sensitive to outliers as it gives equal weight to all data points.
- **SVM:** Less sensitive due to the margin concept (ϵ). Outliers have limited impact on the model.

Modeling Non-linear Relationships

- **OLS:** Assumes linear relationships, may struggle with non-linear relationships without transformations.
- **SVM:** Can model non-linear relationships effectively using kernel functions for higher-dimensional mapping.

Assumptions

- **OLS:** Relies on linearity, normality of errors, homoscedasticity, and independence of errors.
- **SVM:** Fewer assumptions; does not require linearity, normality, homoscedasticity, or independence, offering robustness in diverse scenarios.

Performance on Small Datasets

- **OLS:** May overfit on small datasets, potentially violating assumptions.
- **SVM:** Performs better on small datasets due to the margin (ϵ) and regularization, preventing overfitting.

5.8 Decision Tree Algorithms

5.8.1 ID3

1. **Start at the Root Node:**
   - Begin with the entire training set as the root.
2. **Selecting the Best Attribute:**
   - For each attribute A in the dataset, the ID3 algorithm calculates an attribute's effectiveness in classifying the training data.
   - The measure used for this purpose is the Information Gain, which is based on the concept of Entropy.
3. **Calculate Entropy:**
   - Entropy is a measure of the randomness or uncertainty in the data.
   - The entropy of the entire dataset S is given by:

     $$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

     where p_i is the proportion of the number of elements in class i to the number of elements in set S, and n is the number of classes.
4. **Calculate Information Gain for Each Attribute:**
   - Information Gain is calculated for each attribute. It is the difference in entropy before and after the dataset is split on that attribute.
   - The Information Gain (IG) for an attribute A is given by:

     $$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

     where $Values(A)$ are the different values of attribute A, S_v is the subset of S for which attribute A has value v, and $H(S_v)$ is the entropy of subset S_v.
5. **Choose the Attribute with the Maximum Information Gain:**
   - The attribute with the highest Information Gain is chosen as the decision node.
6. **Split the Dataset:**
   - The dataset is then split by the chosen attribute to produce subsets of the dataset.
7. **Recursion:**
   - The ID3 algorithm is then recursively applied to each subset with the remaining attributes.
8. **Termination Conditions:**
   - This process is repeated until one of the termination conditions is met, such as all samples belong to the same class, there are no more attributes left, or no further information gain is possible.

5.8.2 Comparison of ID3 and C4.5 Algorithms

- **Start with the Entire Dataset:**
  - ID3 and C4.5: Both start with the full dataset as the root of the tree.
- **Choose the Best Attribute:**
  - ID3: Selects the attribute with the highest information gain.
  - C4.5: Also considers the highest information gain, but normalizes it.
- **Calculate Entropy:**
  - ID3 and C4.5: Both use the same entropy formula.
- **Calculate and Normalize Information Gain:**
  - ID3: $IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$
  - C4.5: Uses the same as ID3, followed by normalization (Gain Ratio):
    $\text{GainRatio}(S, A) = \frac{IG(S,A)}{\text{SplitInfo}(S,A)}$
    with
    $\text{SplitInfo}(S, A) = -\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$
- **Handling Continuous Attributes:**
  - ID3: Does not handle continuous attributes.
  - C4.5: Handles continuous attributes by finding an optimal threshold.
- **Pruning:**
  - ID3: No pruning mechanism.
  - C4.5: Prunes trees to avoid overfitting.
- **Recursive Splitting:**
  - ID3 and C4.5: Apply the process recursively using the remaining attributes.
- **Termination Conditions:**
  - ID3 and C4.5: Recursion stops when all instances in a node are of the same class, there are no attributes left, or the subset is too small.

Features of C5.0 Algorithm

- **Efficiency Improvements:**
  - C5.0 is more memory efficient and faster than C4.5.
  - It can handle larger datasets more effectively.
- **Boosting:**
  - C5.0 introduces boosting, building multiple models (trees) sequentially.
  - Each new model focuses on correctly classifying instances misclassified by previous models.
- **Winnowing:**
  - C5.0 can perform a feature selection step, known as winnowing, before building trees.
  - This step helps in removing irrelevant attributes.
- **Handling Continuous and Categorical Attributes:**
  - Similar to C4.5, C5.0 can handle both continuous and categorical attributes.
  - Uses a similar mechanism for handling continuous attributes by finding thresholds.
- **Tree Pruning and Rule Generation:**
  - C5.0 uses tree pruning and can generate rules from decision trees.
  - This aspect is similar to what C4.5 offers.
- **Error-Based Pruning:**
  - Employs a more sophisticated error-based pruning method.
  - This approach can result in smaller and more accurate trees.

5.8.3 Decision Tree Pruning Methods

Pre-Pruning (Early Stopping)

- Stop growing the tree earlier, before it perfectly classifies the training data.
- Set a maximum depth, minimum number of samples per leaf, or a minimum improvement in the impurity measure.

Post-Pruning

- Grow the tree fully, then remove nodes that do not provide significant predictive power.
- **Cost Complexity Pruning:**
  - $$R_\alpha(T) = R(T) + \alpha \times |\text{leaves}|$$
  - $R(T)$: Misclassification rate of the tree T.
  - α: Complexity parameter.
  - $|\text{leaves}|$: Number of leaves in the tree.
    - Minimize $R_\alpha(T)$. Increasing α leads to simpler trees.

Pruning reduces the complexity of the final model, improving its generalizability and interpretiveness.

5.8.4 Decision Tree Splitting Criteria

Gini Impurity

$$Gini(S) = 1 - \sum_{i=1}^{n} (p_i)^2$$

- S: Dataset or subset.
- p_i: Proportion of instances in class i within S.
- n: Number of classes.

Gini impurity measures the likelihood of incorrect classification if you randomly pick an instance and classify it according to the distribution of classes in the subset. A Gini score of 0 indicates perfect purity.

Entropy:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Information Gain:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

- $H(S)$: Entropy of set S.
- p_i: Probability of an item in S belonging to class i.
- A: Attribute for splitting.
- $Values(A)$: All possible values for attribute A.
- S_v: Subset of S for attribute A with value v.

Entropy measures the level of disorder in the data. Information gain is the reduction in entropy from splitting the dataset, aiming to decrease uncertainty.

Classification Error

$$ClassificationError(S) = 1 - \max(p_i)$$

- $\max(p_i)$: Highest proportion of any class in dataset S.

Classification error calculates the error rate of classifying an instance if it were randomly classified according to the distribution of classes in the subset. It focuses on the most frequent class.

Chi-Squared Statistic

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

- O: Observed frequency.
- E: Expected frequency under independence.

The Chi-squared test assesses the independence between the splitting attribute and the target variable. A high value indicates a significant association, suggesting a beneficial split.

Reduction in Variance

$$VarianceReduction = TotalVariance - WeightedVariance$$

- Variance is calculated as the average squared deviation from the mean of the target variable.

Used in regression problems to find splits that reduce the variance of the target variable, indicating more homogeneity.

Information Gain Ratio

$$GainRatio(S, A) = \frac{IG(S, A)}{SplitInfo(S, A)}$$

SplitInfo:

$$SplitInfo(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

- $SplitInfo(S, A)$: Intrinsic information of a split on attribute A.

The information gain ratio is a normalization of information gain, reducing bias towards attributes with more distinct values. It balances a good split with not favoring attributes with more levels.

5.9 Ensemble Models

Bagging (Bootstrap Aggregating)

Bagging, short for Bootstrap Aggregating, is an ensemble machine learning algorithm designed to improve the stability and accuracy of machine learning algorithms. It involves creating multiple versions of a predictor model and using these to get an aggregated predictor. The method works by randomly selecting subsets of the training set with replacement, training a model on each, and then combining their predictions. This approach is particularly effective in reducing variance and avoiding overfitting.

- **Data Sampling:**
  - Given a training dataset D with N samples, create M new training sets $D_1, D_2, \ldots, D_M$, each of size N' (usually $N' = N$), by bootstrap sampling from D.
- **Model Training:**
  - Train a model f_m on each M dataset.
- **Aggregation:**
  - For regression, average all models: $f_{\text{bagging}}(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x)$.
  - For classification, use mode: $f_{\text{bagging}}(x) = \text{mode}\{f_1(x), f_2(x), \ldots, f_M(x)\}$.

Boosting

Boosting is an ensemble machine learning technique used to enhance the performance of predictive models. It builds a sequence of models in a way that each subsequent model attempts to correct the errors of the previous one. The final prediction is typically a weighted sum of the individual models. Boosting is especially known for increasing accuracy in both classification and regression tasks, often significantly reducing bias and variance compared to single models.

- **Initial Model:**
  - Train a weak model f_1 and compute errors e_1.
- **Sequential Training:**
  - For each subsequent model f_i, adjust training data weights based on e_{i-1}, focusing on mistakes.
- **Final Prediction:**
  - Output is a weighted sum: $f_{\text{boosting}}(x) = \sum_{i=1}^{M} \alpha_i f_i(x)$, with α_i based on accuracy.

5.9.1 Multivariate Adaptive Regression Splines (MARS)

Multivariate Adaptive Regression Splines (MARS) is a non-parametric regression technique that extends linear models by incorporating automatic variable selection and nonlinear relationships. It models relationships by fitting piecewise linear regressions, creating 'splines' that adjust to different ranges of the data. MARS is particularly effective for high-dimensional data and can capture complex patterns without requiring a pre-specified functional form.

Base Functions

- *Composition:* Piecewise linear "basis" or "base" functions.
- *Knot:* Point where the function changes slope.
- *Basis Function:* Defined as $h(x, c, s) = \max(0, \frac{x-c}{s})$ where x is the independent variable, c is the knot, and s is a scaling factor.
- *Predicted Value:* $\hat{y}(x) = \beta_0 + \sum_{j=1}^{J} \beta_j h(x, c_j, s_j)$ where β_0 is the intercept, β_j are coefficients, and J is the number of basis functions.

Forward Pass

- Begins by selecting base function pairs that minimize the residual sum of squares (RSS).
- Greedy addition of base functions.
- Stops when RSS reduction is minimal.

Backward Pass (Pruning)

- Addresses overfitting.
- Uses Generalized Cross Validation (GCV).
- GCV formula: $GCV = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}(x_i)}{1 - \frac{h}{N}} \right)^2$
  where y_i are the actual values, $\hat{y}(x_i)$ are the predicted values, N is the number of observations, and h is the effective number of parameters.

Stacking

Stacking, short for stacked generalization, is an ensemble machine learning algorithm. It involves combining multiple predictive models to generate a new model, typically resulting in improved prediction accuracy. In stacking, different algorithms are trained on the same dataset and their predictions are used as inputs to a final 'meta-model', which makes the ultimate prediction. This technique leverages the strengths of each individual model, thereby reducing the risk of choosing a suboptimal algorithm.

- **Base Model Training:**
  - Train base models $f_1, f_2, \ldots, f_M$ on training data.
- **Meta-Data Creation:**
  - Use base models to generate predictions $P_1, P_2, \ldots, P_M$ as meta-features.
- **Meta-Model Training:**
  - Train meta-model g on the meta-features.
- **Final Prediction:**
  - Final prediction by meta-model:

$$f_{\text{stacking}}(x) = g(P_1(x), P_2(x), \ldots, P_M(x))$$

Voting Ensemble

Voting Ensemble is a machine learning technique that combines the predictions from multiple models. It involves creating multiple different models on the same dataset and using a majority vote (for classification) or average (for regression) of their predictions as the final prediction. This approach is beneficial for improving model performance and robustness, as it reduces the likelihood of an unfortunate selection of a poorly performing model. Voting can be 'hard', using a strict majority vote, or 'soft', where probabilities are averaged.

- **Model Selection:**
  - Choose diverse machine learning models.
- **Voting Mechanism:**
  - **Hard Voting:** Majority vote from each model.

    $$\text{Prediction}_{\text{hard}} = \text{mode}\{\text{prediction}_1, \text{prediction}_2, \ldots, \text{prediction}_M\}$$

  - **Soft Voting:** Average of predicted probabilities.

    $$\text{Prediction}_{\text{soft}} = \frac{1}{M} \sum_{m=1}^{M} \text{probability}_m$$

- **Model Training:**
  - Train each model independently on the same dataset.
- **Aggregation of Predictions:**
  - Combine predictions using hard or soft voting.
- **Final Prediction:**
  - Final prediction based on aggregated votes or probabilities.

Random Subspace Method (RSM)

The Random Subspace Method (RSM) is a machine learning technique for improving model accuracy and robustness. It involves training each model on a different random subset of features of the dataset, rather than on the complete feature set. This approach, also known as feature bagging, helps in reducing the correlation among the models in an ensemble, leading to better generalization and reduced overfitting, especially in cases with high-dimensional data. RSM is particularly effective in combination with decision trees and other algorithms sensitive to feature selection.

- **Feature Subsampling:**
  - From a dataset with P features, randomly select k features ($k < P$) for each model.
- **Model Training:**
  - Train separate models on these feature subsets.
- **Aggregation:**
  - Aggregate predictions using averaging or majority voting.
- **Model formulation:**
  - *For classification:*
    $\text{Prediction}_{\text{RSM}} = \text{mode}\{\text{prediction}_1, \text{prediction}_2, \ldots, \text{prediction}_M\}$
  - *For regression:*
    Average predictions from all models.

Mixture of Experts (MoE)

Mixture of Experts (MoE) is an ensemble machine learning technique that divides a complex problem into simpler sub-problems, solved by specialized models called experts. Each expert is trained on a different segment of the data or task, and a gating network determines the weight or influence of each expert's output in the final prediction. MoE effectively combines the outputs of various models, making it well-suited for tasks where different regions of the input space require different types of modeling or expertise.

- **Division of Problem Space:**
  - Divide the problem space into regions for different expert models.
- **Training of Experts:**
  - Train each expert on data corresponding to its region.
- **Gating Network:**
  - Train a gating network to select the appropriate expert for each input.
- **Aggregation:**
  - Combine outputs from experts based on gating network weights.
- **Model formulation:**
  - $f_{\text{MoE}}(x) = \sum_{i=1}^{M} g_i(x) \cdot f_i(x)$ where $g_i(x)$ is the gating network's weight for the i-th expert.

5.9.2 Ensemble Models Comparison

5.9.3 AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble machine learning algorithm used primarily for classification tasks. It works by combining multiple weak learners, typically simple decision trees, to create a strong classifier. In AdaBoost, each subsequent model focuses more on the instances that were incorrectly predicted by previous models, as these receive increased weight. The final prediction is a weighted sum of the predictions from all learners. AdaBoost is known for its effectiveness in boosting the performance of simple models and its ease of implementation.

AdaBoost Classification

- **Initialization:**
  - Dataset: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ where x_i represents features and y_i represents the class label for the i-th observation.
  - Assign equal weights: $w_i = \frac{1}{N}$ for $i = 1, 2, \ldots, N$.
- **Repeat the process until a specified number of iterations or until classification error is minimal:**
  - **Fitting Weak Learners:**
    - In iteration t, train weak learner L_t on the dataset, weighted by w_i.
    - Use L_t to make predictions $\hat{y}_{it}$ on the training data.
  - **Calculating Error and Learner Weight:**
    - Calculate error: $e_t = \sum_{i=1}^{N} w_i \mathbb{1}(\hat{y}_{it} \neq y_i)$, where $\mathbb{1}$ is the indicator function.The indicator function 1(condition)1(condition) returns 1 if the specified condition is true, and 0 if it is false.
    - Determine learner weight: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-e_t}{e_t}\right)$.
  - **Updating Weights:**
    - Update weights: $w_i \leftarrow w_i \cdot \exp(\alpha_t \cdot \mathbb{1}(\hat{y}_{it} \neq y_i))$.
    - Normalize weights to sum to 1.
- **Final Model:**
  - The final prediction model is a weighted vote of the weak learners:

$$\hat{y}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \hat{y}_t(x)\right)$$

    where T is the total number of iterations and sign function returns the class label based on the sign of the summation.

AdaBoost Regression

- **Initialization:**
  - Dataset: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ where x_i represents features and y_i represents the target variable for the i-th observation.
  - Assign equal weights: $w_i = \frac{1}{N}$ for $i = 1, 2, \ldots, N$.
- **Repeat the process until a specified number of iterations or until classification error is minimal:**
  - **Fitting Weak Learners:**
    - In iteration t, train weak learner L_t on the dataset, weighted by w_i.
    - Make predictions $\hat{y}_{it}$ on the training data.
  - **Calculating Error and Learner Weight:**
    - Calculate error: $e_t = \sum_{i=1}^{N} w_i |y_i - \hat{y}_{it}|$.
    - Determine learner weight: $\alpha_t = \eta \cdot \log\left(\frac{1-e_t}{e_t}\right)$ where η is the learning rate.
  - **Updating Weights:**
    - Update weights: $w_i \leftarrow w_i \cdot \exp(\alpha_t \cdot |y_i - \hat{y}_{it}|)$.
    - Normalize weights to sum to 1.
- **Final Model:**
  - The final model: $\hat{y}(x) = \sum_{t=1}^{T} \alpha_t \hat{y}_t(x)$, where T is the total number of learners.

5.9.4 Gradient Boosting

Gradient Boosting is an ensemble machine learning technique used for both classification and regression tasks. It builds the model in a stage-wise fashion, with each new model being trained to correct the errors made by the previous ones. The method uses the gradient descent algorithm to minimize the loss when adding new models. Each tree in the ensemble is fit on a modified version of the original dataset. Gradient Boosting is known for its high effectiveness, particularly in situations where data is unbalanced and in predictive tasks involving complex datasets.

Initialization:

- Start with an initial model, $F_0(x)$, often a constant value such as the mean of the target values.

For each iteration m from 1 to M:

1. Compute the residuals r_{im} for each training instance i, which are the negative gradients of the loss function with respect to the prediction.
2. Fit a weak learner $h_m(x)$ to these residuals.
3. Find the multiplier γ_m that minimizes the loss when added to the current model:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Output the final model:

- The final model is the sum of the initial model and all the weak learners' contributions:

$$F(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x)$$

Aspect	Bagging	Boosting	Stacking	Random Subspace Method	MoE	Blending
Model Training	Independent, parallel training	Sequential, focuses on errors of previous models	Independent base models, followed by a meta-model	Independent, parallel training on feature subsets	Train multiple experts and a gating model	Train models independently, combine using a hold-out set
Data Handling	Bootstrap samples (subsets of data)	Full dataset with adjusted weights for training samples	Full dataset for base models, meta-model on outputs	Full dataset with subsets of features	Full dataset, gating model directs to experts	Full dataset, split into training and validation sets
Model Type	Similar types, e.g., decision trees	Typically similar types	Diverse model types	Similar types, different feature subsets	Diverse, specialized models	Typically diverse models
Impact on Bias	Reduces model-specific bias	Focuses on reducing bias through error correction	Varies based on model selection	Reduced by feature diversity	Depends on the expertise of individual models	Optimized during the blending process
Impact on Variance	Averaging reduces variance	Can increase if overfitting occurs	Depends on the variance of base and meta-models	Mitigated by diversity in feature subsets	Varies, complex models might increase variance	Controlled by validation set
Computational Complexity	Generally lower, parallelizable	Higher due to sequential training	Potentially high due to two levels of training	Similar to Bagging, manageable	Potentially high, complex training	Moderate, depending on complexity of models
Overfitting Risk	Lower due to averaging/majority voting	Higher if not carefully tuned	Depends on base and meta-model complexity	Reduced due to feature diversity	Varies, requires careful design	Lower, due to use of validation set for final model
Applicability	High-variance models, e.g., decision trees	Improving weak models, high-bias situations	Leveraging strengths of different models	High-dimensional feature spaces	Complex problems with diverse data characteristics	Problems where a simpler model can combine predictions

Updated Comparison of Ensemble Techniques in Machine Learning

Evaluation

6.1 Evaluation Approaches

6.1.1 K-fold Validation

Data Splitting

- Shuffle the Dataset Randomly, This ensures that the data splitting into folds is as unbiased as possible.
- Dataset D with N samples.
- Choose K for the number of folds.
- Split D into K subsets $(D_1, D_2, \ldots, D_K)$, each with approximately $\frac{N}{K}$ samples.

Cross-Validation Cycle

- For each fold i (where $i = 1, 2, \ldots, K$):
  - Validation set: D_i.
  - Training set: $D \setminus D_i$ (all data in D except D_i).
  - Train model on $D \setminus D_i$, test on D_i.
  - Record score: S_i.

Calculating Performance Metrics

- Aggregate scores: $S_1, S_2, \ldots, S_K$.
- Mean score: $\text{Mean} = \frac{\sum_{i=1}^{K} S_i}{K}$.
- Standard deviation (optional): $\text{SD} = \sqrt{\frac{\sum_{i=1}^{K} (S_i - \text{Mean})^2}{K}}$.

Interpretation

- The mean score represents the average performance across all folds.
- The standard deviation provides insight into the consistency of the model's performance across different subsets of data.
- The beauty of K-fold cross-validation lies in its ability to use every data point for both training and validation. This makes it a robust method for model evaluation, especially in cases where the dataset isn't large enough to afford a separate hold-out test set.
- By averaging the performance across different subsets, it offers a balanced view of how well the model might perform on unseen data.

6.1.2 The ROC Curve in Binary Classification

- The ROC curve graphically evaluates binary classification models.
- **True Positive Rate (TPR):**
  - Proportion of actual positives correctly identified.
  - Example: In a medical test, TPR indicates how many sick people are correctly diagnosed.
- **False Positive Rate (FPR):**
  - Proportion of actual negatives incorrectly identified as positives.
  - Example: In a medical test, FPR shows how many healthy people are wrongly diagnosed.
- **Threshold Levels:**
  - ROC curve plotted by changing the threshold, the cut-off point for class decisions.
  - Lowering the threshold increases both true and false positives.
  - Raising the threshold reduces false positives but may miss true positives.
- **Shape of the ROC Curve:**
  - A curve closer to the top left corner indicates good performance (high TPR, low FPR).
  - A curve near the diagonal line indicates less effective performance.
- **Area Under the Curve (AUC):**
  - A single number summarizing model performance.
  - A larger AUC indicates a better model. An AUC of 1 is perfect, while 0.5 suggests no discriminative ability.

6.1.3 Accuracy Metrics

- Prevalence: $\frac{\sum \text{Condition Positive}}{\sum \text{Total Population}}$
- Positive Predictive Value (PPV) or Precision: $\frac{\sum \text{True Positive}}{\sum \text{Predicted Condition Positive}}$
- False Omission Rate (FOR): $\frac{\sum \text{False Negative}}{\sum \text{Predicted Condition Negative}}$
- Accuracy (ACC): $\frac{\sum \text{True Positive} + \sum \text{True Negative}}{\sum \text{Total Population}}$
- False Discovery Rate (FDR): $\frac{\sum \text{False Positive}}{\sum \text{Predicted Condition Positive}}$
- Negative Predictive Value (NPV): $\frac{\sum \text{True Negative}}{\sum \text{Predicted Condition Negative}}$
- True Positive Rate (TPR) or Recall or Sensitivity: $\frac{\sum \text{True Positive}}{\sum \text{Condition Positive}}$
- False Positive Rate (FPR) or Fall-out: $\frac{\sum \text{False Positive}}{\sum \text{Condition Negative}}$
- False Negative Rate (FNR) or Miss Rate: $\frac{\sum \text{False Negative}}{\sum \text{Condition Positive}}$
- Specificity (SPC) or Selectivity or TNR: $\frac{\sum \text{True Negative}}{\sum \text{Condition Negative}}$
- Positive Likelihood Ratio (LR+): $\frac{\text{TPR}}{\text{FPR}}$
- Negative Likelihood Ratio (LR-): $\frac{\text{FNR}}{\text{TNR}}$
- Diagnostic Odds Ratio (DOR): $\frac{\text{LR+}}{\text{LR-}}$
- F1 Score: $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Confusion Matrix

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

6.1.4 Lift and Drift Charts

Lift Charts focus on quantifying the effectiveness of a model compared to a baseline, while Drift Charts are concerned with tracking changes in data distributions over time, which can impact the performance of predictive models.

Lift Chart

- Evaluates the performance of a predictive model against a baseline (often random selection).
- **Lift Calculation**:

$$\text{Lift} = \frac{\text{Results with Model}}{\text{Results without Model}}$$

- Example: If targeting 10% of the dataset by the model yields 30% of positives, while random targeting gives 10%, then Lift $= \frac{30\%}{10\%} = 3$. This indicates the model is 3 times as effective as random selection.

Drift Chart

- Identifies changes in data distribution over time, essential in machine learning.
- **Types of Drift**:
  - Data Drift: Changes in input data distribution.
  - Model Drift: Degradation in model performance due to changing data relationships.
- **Common Method for Detection**:
  - Statistical Measures: Kullback-Leibler divergence (KL divergence).
- **KL Divergence Equation**:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

KL Divergence for Data Drift Detection

- Compares the distribution of input features between training data and new, incoming data.

$$D_{KL}(P_{\text{train}} \parallel P_{\text{new}}) = \sum_{x \in X} P_{\text{train}}(x) \log\left(\frac{P_{\text{train}}(x)}{P_{\text{new}}(x)}\right)$$

- Identifies significant changes in input data, crucial for maintaining model accuracy as input patterns evolve.

KL Divergence for Model Drift Detection

- Compares predicted probability distributions from a historically trained model with those from a recently trained model.

$$D_{KL}(P_{\text{historic}} \parallel P_{\text{recent}}) = \sum_{x \in X} P_{\text{historic}}(x) \log\left(\frac{P_{\text{historic}}(x)}{P_{\text{recent}}(x)}\right)$$

- Helps understand shifts in model predictions over time, indicating changes in data patterns or relationships, crucial for adapting the model to current data trends.

Anomalies and Outliers

7.1 Anomalies and Outliers

Concept and Definition

- **Anomalies:**
  - Data points that deviate significantly from expected behavior.
  - Represent instances not conforming to general patterns or trends.
- **Outliers:**
  - Specific type of anomaly, significantly different and located far from the majority.
  - Affect statistical analysis and may indicate different mechanisms.

Characteristics

- **Anomalies:**
  - Unexpected, infrequent, significant deviations.
- **Outliers:**
  - Extreme values, statistically rare, can skew results.

Detection Techniques

- **Anomalies:**
  - Include supervised, semi-supervised, and unsupervised methods.
- **Outliers:**
  - Identified using statistical measures, such as Z-scores, IQR.

Applications

- **Anomalies:**
  - Used in cybersecurity, medicine, machine vision, financial fraud.
- **Outliers:**
  - Significant in statistical analyses, quality control, market analysis.

Types

- **Anomalies:**
  - Categorized based on domain or detection method.
- **Outliers:**
  - Types include Global Outliers, Contextual Outliers, Collective Outliers.

7.2 Isolation Forest Algorithm

The Isolation Forest algorithm isolates each data point by randomly selecting features and split values, creating an ensemble of iTrees. It then calculates the path length for each point in these trees. Anomalies are identified based on their shorter path lengths, which are used to compute an anomaly score indicating the likelihood of a point being an outlier in the dataset.

1. Construction of Isolation Trees (iTrees)

Isolation Forest constructs iTrees for a given dataset. An iTree is similar to a binary search tree, but it's constructed by randomly selecting an attribute and a split value for that attribute to partition the data. The process continues recursively until a termination condition is met.

2. Path Length $h(x)$

The path length $h(x)$ is a measure of the number of edges a point x traverses in the iTree from the root node to an external node. Anomalous points typically have shorter path lengths because they are easier to isolate.

3. Anomaly Score Calculation

The anomaly score for a data point is calculated using the path length. The average path length $E(h(x))$ across all iTrees for a point x is used in the calculation. The formula for the anomaly score $s(x, m)$ of a data point x in a sample of size m is:

$$s(x, m) = 2^{-\frac{E(h(x))}{c(m)}}$$

Here, $c(m)$ is a normalization factor given by:

$$c(m) = \begin{cases} 2H(m-1) - \frac{2(m-1)}{n} & \text{for } m > 2 \\ 1 & \text{for } m = 2 \\ 0 & \text{otherwise} \end{cases}$$

where $H(i)$ is the i-th harmonic number and n is the size of the testing dataset. γ is the Euler-Mascheroni constant, approximately 0.5772156649.

4. Interpreting the Anomaly Score

The value of $s(x, m)$ indicates the likelihood of a point being an anomaly:

- If $s(x, m)$ is close to 1, the point x is highly likely to be an anomaly.
- If $s(x, m)$ is much smaller than 0.5, the point x is likely to be a normal point.
- A score around 0.5 indicates uncertainty.

7.3 Cook's Distance

Cook's Distance is a measure used in statistics to identify influential observations in a dataset, particularly in the context of linear regression. It estimates the influence of a data point in a least-squares regression analysis.

Mathematical Expression

The formula for Cook's Distance is:

$$D_i = \frac{\sum_{j=1}^{n} (\hat{Y}_j - \hat{Y}_j^{(i)})^2}{p\, MSE}$$

Where:

- D_i is Cook's Distance for the i-th observation.
- $\sum_{j=1}^{n} (\hat{Y}_j - \hat{Y}_j^{(i)})^2$ is the sum of squared differences between the predicted values $\hat{Y}_j$ from the full model and the predicted values $\hat{Y}_j^{(i)}$ from the model without the i-th observation.
- p is the number of predictors in the model.
- MSE is the mean squared error of the full model.

Key Components

- **Residuals and Leverage:** Considers the contribution of an observation's residual to the overall prediction error, and the observation's leverage on the regression line.
- **Sum of Squared Differences:** The numerator calculates the sum of squared differences in predicted values with and without the particular observation.
- **Normalization Factor:** The denominator normalizes this sum by the number of predictors and the mean squared error.

An observation with a high Cook's Distance indicates significant influence on the model's parameters. A threshold, such as $4/n$ (where n is the number of observations), is often used to identify observations with a substantially high Cook's Distance as influential.

7.4 Quartiles and Interquartile Range (IQR)

Quartiles

- In a dataset:
  - The first quartile, $Q1$, is the median of the lower half of the data.
  - The third quartile, $Q3$, is the median of the upper half of the data.
  - $Q2$, or the second quartile, is the median of the entire dataset, but it's not used in calculating the IQR.

Calculating the IQR

- The IQR is the difference between the third and first quartiles:

$$\text{IQR} = Q3 - Q1$$

- This value represents the spread of the middle 50% of the data.

Identifying Outliers Using IQR

- Outliers are identified using the IQR:
  - Lower Bound for Outliers (data points less than this value are considered lower outliers):

$$\text{Lower Bound} = Q1 - 1.5 \times \text{IQR}$$

  - Upper Bound for Outliers (data points greater than this value are considered upper outliers):

$$\text{Upper Bound} = Q3 + 1.5 \times \text{IQR}$$

- The choice of 1.5 as the multiplier is conventional but effective in distinguishing typical data points from those that are significantly different.

7.5 Local Outlier Factor

The Local Outlier Factor (LOF) algorithm is a method for detecting outliers in a dataset by examining the local density deviation of each data point compared to its neighbors.

Step 1: Determining the k-Distance

The k-Distance of a point P is defined as the distance of P to its k^{th} nearest neighbor. This is mathematically represented as:

$$k\text{-distance}(P) = \text{dist}(P, O_k)$$

where O_k is the k^{th} nearest neighbor of P and $\text{dist}(P, O_k)$ is the distance between P and O_k.

Step 2: Calculating the Reachability Distance

The Reachability Distance between two points P and O is given by:

$$\text{Reachability-Distance}_k(P, O) = \max\{k\text{-distance}(O), \text{dist}(P, O)\}$$

This represents the maximum of the k-distance of O and the actual distance between P and O.

Step 3: Computing the Local Reachability Density (LRD)

The LRD of a point P is calculated as:

$$\text{LRD}_k(P) = \left(\frac{\sum_{O \in N_k(P)} \text{Reachability-Distance}_k(P, O)}{|N_k(P)|} \right)^{-1}$$

where $N_k(P)$ is the set of k nearest neighbors of P. The LRD is the inverse of the average reachability distance from P to its neighbors.

Step 4: Determining the Local Outlier Factor (LOF)

The LOF of a point P is determined as:

$$\text{LOF}_k(P) = \frac{\sum_{O \in N_k(P)} \frac{\text{LRD}_k(O)}{\text{LRD}_k(P)}}{|N_k(P)|}$$

The LOF is the average of the ratio of the LRD of P to the LRD of its neighbors. A high LOF value, significantly greater than 1, indicates that P is an outlier.

Overview

The LOF algorithm is particularly effective in identifying outliers due to several key reasons:

- **Focus on Local Spatial Properties:** It emphasizes the local spatial characteristics of data points, rather than their global distribution in the dataset.
- **Useful for Varying Densities:** The algorithm is especially useful in datasets where density varies significantly, accommodating clusters that are either more sparse or dense than others.
- **Identification of Notably Different Points:** The LOF score quantifies the extent to which an object deviates from its neighboring points in terms of density. This helps in identifying points that are significantly different or isolated from their local surroundings.

7.6 Mahalanobis Distance

Measures the distance between a point and a distribution, particularly in a multivariate context.

$$D^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

- D^2: Square of the Mahalanobis distance.
- x: Vector of the observation.
- μ: Mean vector of independent variables.
- Σ^{-1}: Inverse covariance matrix of independent variables.

Effective in multivariate anomaly detection and classification.

- Sensitivity to Outliers: Calculations can be significantly affected by outliers, potentially leading to misleading results.
- Assumption of Gaussian Distribution: Works best when the data distribution is Gaussian, which might not be the case in all datasets.

7.7 Minimum Covariance Determinant (MCD) Method

The goal of the MCD method is to find a subset of the dataset with the smallest covariance determinant, representing the "normal" observations and reducing the influence of outliers.

Process

- **Dataset and Subset Selection:**
  - Dataset $X = \{x_1, x_2, \ldots, x_n\}$, where each x_i is in a p-dimensional space.
  - Find subset $X_h \subset X$ with h observations $(n/2 < h \leq n)$.
- **Calculating Mean and Covariance for the Subset:**
  - Mean μ_h of X_h:
$$\mu_h = \frac{1}{h} \sum_{i \in X_h} x_i$$
  - Covariance matrix Σ_h of X_h:
$$\Sigma_h = \frac{1}{h-1} \sum_{i \in X_h} (x_i - \mu_h)(x_i - \mu_h)^T$$
- **Minimizing the Determinant:**
  - Select X_h to minimize the determinant of Σ_h.
  - Optimization problem:
$$\min_{X_h \subset X, |X_h| = h} \det(\Sigma_h)$$
- **Final Estimation:**
  - The subset X_h minimizing the determinant provides robust MCD estimates of mean and covariance.

Solving the MCD Method

- **Computational Challenge:** Evaluating many subsets for the minimum determinant covariance matrix is intensive, especially for large datasets.
- **Heuristic Approaches:** Approximations, like the Fast-MCD algorithm, are used for efficiency.
- **Fast-MCD Algorithm:** An iterative algorithm that refines the subset selection to minimize the determinant.
- **Iterative Process:** Each iteration updates the subset to better approximate the minimum determinant.
- **Random Sampling:** Initial subset selection involves random sampling to cover a broad range of possibilities.

Significance of the MCD Method

- **Outlier Resistance:** MCD is robust against outliers, reducing their impact on covariance estimation.
- **Improved Analysis:** Provides more accurate covariance estimates in datasets with outliers.
- **Wide Application:** Useful in fields like finance and economics where outliers are common.
- **Better Decision-Making:** Leads to more reliable decision-making in statistics-based models.
- **Enables Advanced Techniques:** Essential for complex statistical methods in datasets with outliers.

7.8 Single-Class SVM

- Single-class SVM is used for anomaly detection.
- It focuses on a single class, unlike standard SVMs which handle two or more classes.
- The goal is to establish a decision boundary that separates the data points of a single class from the origin in high-dimensional space.
- The technique involves mapping data points into high-dimensional space for separation.
- Useful in scenarios with significant data from one class to detect anomalies or outliers.

1. Tax and Duin's Formulation (Using a Hypersphere)

- Enclosing data in a hypersphere in high-dimensional space.
- **Center and Radius:** Hypersphere characterized by center **a** and radius R.
- **Objective:** Minimize the radius while keeping all data points inside or on its surface.
- **Optimization Problem:**

$$\begin{aligned} \text{Minimize:} \quad & R^2 + C \sum_i \xi_i \\ \text{Subject to:} \quad & |\mathbf{x}_i - \mathbf{a}|^2 \ \leq \ R^2 + \xi_i \ \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

- $\mathbf{x}_i$ are data points, ξ_i are slack variables, and C is a regularization parameter.

2. Schölkopf's Formulation (Using a Hyperplane)

- Using a hyperplane instead of a hypersphere.
- **Objective:** Maximize the distance of the hyperplane from the origin.
- **Optimization Problem:**

$$\begin{aligned} \text{Maximize:} \quad & \frac{1}{|w|} \\ \text{Subject to:} \quad & (\mathbf{w} \cdot \mathbf{x}_i) + b \geq \rho - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \\ & \sum_i \alpha_i = 1 \end{aligned}$$

- $\mathbf{w}$ is the normal to the hyperplane, b is the bias, ρ is the margin.

Information Theory

8.1 Shannon Uncertainty Formula

The Shannon Uncertainty Formula, also known as Shannon's entropy, is a fundamental concept in information theory, introduced by Claude Shannon. It quantifies the uncertainty or the amount of information contained in a random variable or a system. The formula is expressed as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

Where:

- $H(X)$ is the entropy of the random variable X.
- $\mathcal{X}$ represents the set of all possible outcomes of X.
- $p(x)$ is the probability of an outcome x.
- The logarithm base is chosen depending on the context (base 2 for bits, base e for natural units, and base 10 for dits).

The entropy $H(X)$ measures the average level of "information", "surprise", or "uncertainty" inherent in a random variable's possible outcomes. It represents the average amount of information conveyed by identifying the outcome of a random trial.

Key Properties of Shannon Uncertainty Formula

- **Non-negativity**: Entropy is always non-negative, implying that the average amount of information or uncertainty in a system cannot be a negative value.
- **Maximum entropy with uniform distribution**: Entropy is maximized when the distribution of the random variable is uniform. This is because uniform distribution indicates the highest level of uncertainty or lack of specific information about the variable.
- **Additivity for independent events**: For two independent random variables X and Y, the entropy of their joint distribution is the sum of their individual entropies:

$$H(X, Y) = H(X) + H(Y)$$

8.2 Boltzmann's Entropy Formula

Boltzmann's entropy formula is a cornerstone in statistical mechanics and thermodynamics, offering a mathematical expression for the concept of entropy.

Basic Formula

The basic formula is given as:

$$S = k_B \ln(\Omega)$$

where:

- S is the Entropy of the system.
- k_B is the Boltzmann constant, valued at $1.38 \times 10^{-23}\,\text{J/K}$.
- Ω represents the number of possible microscopic configurations (microstates) of the system.

Interpretation

- Entropy, S, measures the degree of disorder or randomness within a system.
- A higher number of microstates, Ω, implies greater entropy.
- This formula is applicable in systems where all microstates are equally probable.

Relation with Probability

- The original Boltzmann formula connects entropy with the probability, W, of the system being in a particular microstate:

$$S = k_B \ln(W)$$

- This emphasizes the probabilistic nature of entropy.

Applications

This formula finds extensive applications in fields such as physics, chemistry, and information theory.

Entropy in Systems with Multiple Particles

- Microstates for N Particles
  - Consider a system with N identical particles, each capable of being in one of K states.
  - The total number of microstates, W, is calculated as:

$$W = \frac{(N + K - 1)!}{N!(K - 1)!}$$

- Entropy Calculation
  - Using Stirling's approximation, the logarithm of W is approximated as:

$$\ln(W) \approx N \ln(K) - N \ln(N) - (K - N) \ln(K - N)$$

- Boltzmann Entropy for the System
  - The entropy for a system with N particles is given by:

$$S = k_B \left(N \ln(K) - N \ln(N) - (K - N) \ln(K - N)\right)$$

  - Here, N signifies the number of particles in a particular state, and $K - N$ the number in other states.

- Links convexity or concavity of functions to expected values.
- Established by Johan Jensen in 1906.
- Influential in statistics, optimization, economics, and various other mathematical fields.

8.3 Jensen's Inequality

For Convex Functions

- **Inequality Statement:** $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$
- **Meaning:** When f is a convex function and X is a random variable, the function value at the expected value of X is less than or equal to the expected value of the function of X.
- **Interpretation:** For convex functions, the "average" output is at least as large as the output at the "average" input.

For Concave Functions

- **Inequality Statement:** $f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$
- **Meaning:** For a concave function f and a random variable X, the function value at the expected value of X is greater than or equal to the expected value of the function of X.
- **Interpretation:** For concave functions, the "average" output is at most as large as the output at the "average" input.

8.4 Fisher's Score and Fisher's Information

In essence, Fisher's Score provides a mechanism to locate the most probable parameter values in a likelihood function, while Fisher's Information quantifies how much certainty there is in these estimates.

Fisher's Score

Fisher's Score is the derivative (gradient) of the log-likelihood function with respect to the parameter.

$$s(\theta) = \frac{\partial}{\partial \theta} \log \mathcal{L}(\theta)$$

where:

- $s(\theta)$ is the Fisher's Score,
- θ is the parameter,
- $\mathcal{L}(\theta)$ is the likelihood function.

The score function is used to find the point where the likelihood function reaches its maximum with respect to the parameter. Setting the score to zero leads to the maximum likelihood estimate.

Fisher's Information

Fisher's Information measures the amount of information an observable random variable carries about an unknown parameter of a distribution that models the variable.

- Variance of the Score:
$$I(\theta) = \text{Var}[s(\theta)]$$
where $I(\theta)$ is the Fisher's Information.
- Expected Value of Second Derivative of Log-Likelihood:
$$I(\theta) = -E\left[\frac{\partial^2}{\partial \theta^2} \log \mathcal{L}(\theta)\right]$$
- As Variance of the Score:
$$I(\theta) = E\left[\left(\frac{\partial}{\partial \theta} \log f(X;\theta)\right)^2\right]$$
where $f(X;\theta)$ is the probability density or mass function of the random variable X.

Higher Fisher Information suggests less variance in the parameter estimate, indicating more certainty in the estimate.

8.5 Kullback-Leibler Divergence

- The Kullback-Leibler (KL) divergence, denoted as $D_{\mathrm{KL}}(P \parallel Q)$, is a statistical measure used to quantify how one probability distribution P differs from a second, reference distribution Q.
- It's not a symmetric measure and doesn't satisfy the triangle inequality, setting it apart from a traditional metric.

For Discrete Distributions

- **Distributions:** P and Q.
- $D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$.
- **General Interpretation:** Sums over all possible events x in the sample space $\mathcal{X}$, each term being the product of the probability of event x under P and the logarithm of the ratio of probabilities under P and Q.
- **Information Theory Perspective:** Quantifies the additional bits required for encoding each event x using a model Q instead of P.

For Continuous Distributions

- **Density Functions:** $p(x)$ and $q(x)$.
- $D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$.
- **General Interpretation:** Integral over the entire range of the random variable, involving the product of the probability density under P and the logarithm of the ratio of densities under P and Q.
- **Information Theory Perspective:** Measures the continuous "information cost" or extra bits required when outcomes are coded using Q instead of P.

8.6 Mutual Information

Mutual Information (MI) is a measure used in statistics to quantify the amount of information obtained about one random variable by observing another. It is intimately linked to the concept of entropy in information theory. The definition and computation of MI depend on whether the random variables involved are discrete or continuous.

General Definition

For a pair of random variables X and Y with joint distribution $P_{(X,Y)}$ and marginal distributions P_X and P_Y, the mutual information is defined as:

$$I(X;Y) = D_{\mathrm{KL}}(P_{(X,Y)} \| P_X \otimes P_Y)$$

Here, D_{KL} represents the Kullback–Leibler divergence, and $P_X \otimes P_Y$ is the outer product distribution assigning probability $P_X(x) \cdot P_Y(y)$ to each pair (x, y).

For Discrete Distributions

When X and Y are discrete, MI is calculated as a double sum:

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y)}(x,y) \log \left(\frac{P_{(X,Y)}(x,y)}{P_X(x) P_Y(y)} \right)$$

Here, $P_{(X,Y)}$ is the joint probability mass function, and P_X and P_Y are the marginal probability mass functions of X and Y, respectively.

For Continuous Distributions

For continuous variables, the double sum is replaced by a double integral:

$$I(X;Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{(X,Y)}(x,y) \log \left(\frac{P_{(X,Y)}(x,y)}{P_X(x) P_Y(y)} \right) dx dy$$

In this case, $P_{(X,Y)}$ is the joint probability density function.

Equivalent Expressions

Mutual information can also be expressed in terms of entropy:

$$I(X;Y) \equiv H(X) - H(X|Y) \equiv H(Y) - H(Y|X) \equiv$$

$$H(X) + H(Y) - H(X,Y) \equiv H(X,Y) - H(X|Y) - H(Y|X)$$

where $H(X)$ and $H(Y)$ are the marginal entropies, $H(X|Y)$ and $H(Y|X)$ are the conditional entropies, and $H(X,Y)$ is the joint entropy of X and Y.

Time Series Forecasting

9.1 Common Components of a Time Series

- **Trend Component:**
  - Indicates persistent increases or decreases over time.
  - Represents the long-term progression of the series.
  - Can be linear or non-linear.
  - *Example:* Steady increase in average global temperatures over decades.
- **Seasonal Component:**
  - Repeats over a fixed period.
  - Reflects seasonality or periodic fluctuations.
  - *Example:* Higher hotel bookings during summer vacation seasons.
- **Cyclical Component:**
  - Fluctuations in a non-fixed, irregular pattern.
  - Usually tied to economic conditions, like boom or recession.
  - Can span several years.
  - *Example:* Business cycle phases impacting employment rates.
- **Irregular (or Random) Component:**
  - Consists of random fluctuations not attributed to other components.
  - Also known as the ”residual” or ”error” component.
  - Caused by unpredictable or random events.
  - Usually of short duration.
  - *Example:* Sudden stock market fluctuations due to an unexpected political event.

9.2 Autocorrelation Function (ACF)

- Autocorrelation measures the correlation of a signal or time series with a delayed version of itself, It is calculated for various time lags.
- The autocorrelation function is used to find repeating patterns or periodic signals within a dataset.
- For instance, it can identify if there's a regular, cyclical behavior in temperature readings over days or in stock market prices over weeks.

Basic Definition

For a time series X_t, where t represents time, the autocorrelation function (ACF) assesses the correlation between X_t and X_{t-h} for various values of h (lag). The formula is:

$$\text{ACF}(h) = \frac{\text{Cov}(X_t, X_{t-h})}{\text{Var}(X_t)}$$

where:

- $\text{Cov}(X_t, X_{t-h})$ is the covariance between X_t and X_{t-h}.
- $\text{Var}(X_t)$ is the variance of X_t.

Wide-Sense Stationary Process

In a wide-sense stationary (WSS) process, where the mean and variance are constant over time, the autocovariance and autocorrelation depend only on the lag, not on the specific time t. For such processes, the ACF is defined as:

$$R_{XX}(\tau) = E[X_{t+\tau}\overline{X}_t]$$

where:

- τ is the lag.
- $E[\cdot]$ is the expectation operator.
- $\overline{X}_t$ denotes the complex conjugate of X_t.

Continuous and Discrete Signals

Continuous Signal: For a continuous signal $f(t)$, the autocorrelation is defined as an integral:

$$R_{ff}(\tau) = \int_{-\infty}^{\infty} f(t+\tau)\overline{f(t)}\,dt$$

Discrete Signal: For a discrete-time signal $y(n)$, the autocorrelation at lag ℓ is given by the sum:

$$R_{yy}(\ell) = \sum_{n\in\mathbb{Z}} y(n)\overline{y(n-\ell)}$$

Estimating Autocorrelation Coefficient

For a discrete process with known mean μ and variance σ^2, and n observations, the estimated autocorrelation coefficient at lag k is:

$$\hat{R}(k) = \frac{1}{(n-k)\sigma^2}\sum_{t=1}^{n-k}(X_t - \mu)(X_{t+k} - \mu)$$

Practical Significance

In time series analysis, ACF is used to detect non-randomness, identify seasonality or periodicity, and to inform the choice of model (e.g., ARIMA models in forecasting). In signal processing, it helps in identifying signal properties, such as the presence of a periodic signal.

Normalization

In statistics and time series analysis, it's common to normalize the autocovariance function to get a time-dependent Pearson correlation coefficient. The auto-correlation coefficient for a stochastic process is:

$$\rho_{XX}(t_1, t_2) = \frac{K_{XX}(t_1, t_2)}{\sigma_{t_1}\sigma_{t_2}}$$

9.3 Partial Autocorrelation Function (PACF)

Definition and Basic Concept:

- PACF is the partial correlation of a stationary time series with its own lagged values, regressed against the values of the time series at all shorter lags.
- In simpler terms, it tells you the direct relationship between an observation and its lag, removing the influence of intermediate lags.
- The PACF of order k can be defined as the last element in the matrix R_k divided by r_0, where R_k is a $k \times k$ matrix and C_k is a $k \times 1$ column vector.

Calculation:

- The 1st order PACF is defined to be equal to the 1st order autocorrelation.
- For higher orders, the 2nd order (lag) PACF is given by the equation:

$$\text{PACF}_2 = \frac{\text{Covariance}(x_t, x_{t-2} \mid x_{t-1})}{\sqrt{\text{Variance}(x_t \mid x_{t-1})\text{Variance}(x_{t-2} \mid x_{t-1})}}$$

  This represents the correlation between values two time periods apart, conditional on the knowledge of the value in between.

- Similarly, the 3rd order (lag) PACF is calculated as:

$$\text{PACF}_3 = \frac{\text{Covariance}(x_t, x_{t-3} \mid x_{t-1}, x_{t-2})}{\sqrt{\text{Variance}(x_t \mid x_{t-1}, x_{t-2})\text{Variance}(x_{t-3} \mid x_{t-1}, x_{t-2})}}$$

  This continues for higher lags.

Application in Time Series Analysis:

- PACF is particularly useful in identifying the order of an autoregressive (AR) model in time series analysis.
- The theoretical ACF and PACF for AR, MA, and ARMA conditional mean models are known and are different for each model, aiding in model selection.

9.4 Autoregressive Integrated Moving Average (ARIMA)

Key Parameters

- **p (Autoregressive Part)**:
  - Order of the autoregressive component.
  - Number of lags of the dependent variable used as predictors.
  - Captures the influence of prior values on current values.
- **d (Differencing)**:
  - Degree of differencing to make the series stationary.
  - Transformation to stabilize the mean and variance over time.
  - Removes trends and seasonal effects.
- **q (Moving Average Part)**:
  - Order of the moving average component.
  - Number of lagged forecast errors included in the model.
  - Addresses the influence of random shocks from previous points.

General Equation for Non-Seasonal ARIMA

$$(1 - \sum_{i=1}^{p} \varphi_i L^i)(1 - L)^d X_t = (1 + \sum_{i=1}^{q} \theta_i L^i)\varepsilon_t$$

- L: Lag operator.
- X_t: Time series value at time t.
- φ_i: Coefficients for the autoregressive part.
- θ_i: Coefficients for the moving average part.
- ε_t: Error terms.

Equation with Drift Component

$$(1 - \sum_{i=1}^{p} \varphi_i L^i)(1 - L)^d X_t = \delta + (1 + \sum_{i=1}^{q} \theta_i L^i)\varepsilon_t \tag{9.1}$$

- δ: Represents the drift component, indicating a linear trend.

Application and Fitting

- Identifying appropriate values for p, d, and q based on time series data characteristics.
- Once parameters are determined, the model is used for forecasting future values.

9.5 SARIMAX Model

ARIMAX Model

- Extends the ARIMA model by including exogenous inputs.
- Incorporates independent variables influencing the time series but not autoregressed on.
- Models the time series using both the series itself and other independent variables.

SARIMA Model

- Stands for Seasonal ARIMA, used for time series with evident seasonality.
- Combines two ARIMA models: one for non-seasonal and one for seasonal parts.
- Includes extra seasonal parameters: P (Seasonal autoregressive order), D (Seasonal differencing order), Q (Seasonal moving average order), S (Length of the seasonal cycle).

SARIMAX Model

- Combines SARIMA and ARIMAX, incorporating both seasonal components and exogenous inputs.
- Enhances ARIMA by adding seasonality and external data for improved forecasting.
- Equation for SARIMAX(p,d,q)(P,D,Q,s):

$$\Theta(L)^p\theta(L^s)^P\Delta^d\Delta_s^D y_t = \Phi(L)^q\phi(L^s)^Q\Delta^d\Delta_s^D\epsilon_t + \sum_{i=1}^{n}\beta_i x_t^i$$

  - $\Theta(L)^p$ and $\Phi(L)^q$ are the non-seasonal components.
  - $\theta(L^s)^P$ and $\phi(L^s)^Q$ are the seasonal components.
  - Δ^d and Δ_s^D represent differencing operations.
  - y_t is the time series, ϵ_t is the error term.
  - x_t^i with coefficients β_i are the exogenous variables.

9.6 Simple Exponential Smoothing (SES)

Simple exponential smoothing (SES) is a method suitable for forecasting time series data that does not display any clear trend or seasonal pattern. It is part of the family of exponential smoothing methods and works particularly well for data that is essentially random with no evident seasonality or trend.

Core Principles of Simple Exponential Smoothing:

- **Weighted Averages:** SES forecasts are calculated using weighted averages where the weights decrease exponentially as observations come from further in the past. The smallest weights are associated with the oldest observations. This can be represented by the equation:

  $$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

  Here, (0 ¡ α ¡ 1) is the smoothing parameter, y_T is the last observed value, and $\hat{y}_{T+1|T}$ is the forecast for the next period.

- **Influence of α:** The value of α determines the weights assigned to observations. A small α gives more weight to observations from the distant past, while a large α gives more weight to recent observations. If α equals 1, the SES forecast is the same as the last observed value, akin to the naïve forecast.

- **Forecast :** The forecast at time $T+1$ is a weighted average between the most recent observation y_T and the previous forecast $\hat{y}_{T|T-1}$:

  $$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}$$

  This formula is used iteratively to calculate forecasts for each time period.

- **Component Form:** In simple exponential smoothing, the only component is the level ℓ_t. The component form comprises a forecast equation and a smoothing equation:

  $$\hat{y}_{t+h|t} = \ell_t$$

  $$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$

  The forecast value at time $t+1$ is the estimated level at time t.

9.7 Damped Trend Method

The damped trend method in time series forecasting enhances traditional models by incorporating a damping factor, ϕ, to account for the diminishing impact of trends over time.

Key Advantages

- By incorporating the damping factor ϕ, this method modifies the trend component to account for a diminishing impact over time.
- The reduced impact of trends over time makes the damped trend method particularly suitable for long-term forecasting where trends are expected to level off.
- It provides more realistic forecasts, especially for data with trends that are likely to decelerate, offering a conservative yet often more accurate long-term outlook compared to models assuming a continuous linear trend.

Level

$$\ell_t = \alpha y_t + (1 - \alpha)(\ell_{t-1} + \phi b_{t-1})$$

Similar to Holt's model, it updates the level of the series. The presence of ϕ in the equation dampens the influence of past trends on the current level.

Trend

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1 - \beta^*)\phi b_{t-1}$$

The trend component is adjusted for damping. The factor ϕ dampens the trend, effectively reducing the influence of past trends over time.

Forecast

$$\hat{y}_{t+h|t} = \ell_t + (\phi + \phi^2 + \ldots + \phi^h)b_t$$

This equation projects the future values, taking into account the dampening effect on the trend. The term $(\phi + \phi^2 + \ldots + \phi^h)$ represents the cumulative dampening effect over h periods.

9.8 Holt's linear trend method

Holt's linear trend method is an extension of exponential smoothing used for time series forecasting, particularly effective when the data exhibits trends. This method involves two primary equations: the level equation and the trend equation.

Level

The level equation updates the series' current level, which is a smoothed estimate of the series' value.

$\ell_t = \alpha y_t + (1 - \alpha)(\ell_{t-1} + b_{t-1})$

- ℓ_t is the estimated level at time t.
- α is the smoothing parameter for the level, between 0 and 1.
- y_t is the actual observed value at time t.
- ℓ_{t-1} and b_{t-1} are the estimated level and trend, respectively, from the previous time step.

Trend

The trend equation updates the trend component, reflecting changes in the level.

$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1 - \beta^*)b_{t-1}$

- b_t is the estimated trend at time t.
- β^* is the smoothing parameter for the trend, also between 0 and 1.
- The trend component is the estimated change in the level component from one period to the next.

Forecast

Combines the level and trend components to forecast future values.

$\hat{y}_{t+h|t} = \ell_t + hb_t$

- $\hat{y}_{t+h|t}$ is the forecast for h periods ahead.
- The equation suggests that the future value is a function of the current estimated level and the current trend, projected h steps into the future.

Key Characteristics of Holt's Method

- **Effectiveness for Linear Trends:** Tailored for data exhibiting a linear trend, providing accurate forecasting in such scenarios.
- **Dynamic Adjustment:** Dynamically updates both the level and trend of the series, offering adaptability to changes in the data.
- **Flexibility and Robustness:** Adjusts to changing trends, making it a flexible and robust forecasting tool.
- **Importance of Smoothing Parameters (α and β^*):** Critical in determining the model's responsiveness to changes in the level and trend of the data. These parameters balance between the historical data's relevance and recent observations.

9.9 Exponential Smoothing Methods

Exponential Smoothing Methods

Trend	Seasonal	Forecast Equation	Level Equation	Seasonality Equation
N	N	$\hat{y}_{t+h\|t} = l_t$	$l_t = \alpha y_t + (1-\alpha) l_{t-1}$	N/A
N	A	$\hat{y}_{t+h\|t} = l_t + s_{t+h-m(k+1)}$	$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha) l_{t-1}$	$s_t = \gamma(y_t - l_{t-1}) + (1-\gamma) s_{t-m}$
N	M	$\hat{y}_{t+h\|t} = l_t s_{t+h-m(k+1)}$	$l_t = \alpha(y_t / s_{t-m}) + (1-\alpha) l_{t-1}$	$s_t = \gamma(y_t / l_{t-1}) + (1-\gamma) s_{t-m}$
A	N	$\hat{y}_{t+h\|t} = l_t + h b_t$	$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$	N/A
A	A	$\hat{y}_{t+h\|t} = (l_t + h b_t) + s_{t+h-m(k+1)}$	$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$	$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma) s_{t-m}$
A	M	$\hat{y}_{t+h\|t} = (l_t + h b_t) s_{t+h-m(k+1)}$	$l_t = \alpha(y_t / s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$	$s_t = \gamma(y_t / (l_{t-1} + b_{t-1})) + (1-\gamma) s_{t-m}$
A_d	N	$\hat{y}_{t+h\|t} = l_t + \phi h b_t$	$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + \phi b_{t-1})$	N/A
A_d	A	$\hat{y}_{t+h\|t} = (l_t + \phi h b_t) + s_{t+h-m(k+1)}$	$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + \phi b_{t-1})$	$s_t = \gamma(y_t - l_{t-1} - \phi b_{t-1}) + (1-\gamma) s_{t-m}$
A_d	M	$\hat{y}_{t+h\|t} = (l_t + \phi h b_t) s_{t+h-m(k+1)}$	$l_t = \alpha(y_t / s_{t-m}) + (1-\alpha)(l_{t-1} + \phi b_{t-1})$	$s_t = \gamma(y_t / (l_{t-1} + \phi b_{t-1})) + (1-\gamma) s_{t-m}$