Faculty of Computer science and Artificial Intelligence

Dimensionality Reduction using Evolutionary Algorithms

Supervised By :
DR/ Amr Ghoneim & T.A./ Omar Tarek

abdallah wael
abdallah mohamed
abdallah faisal ragab
alhosein ali
hossam mohamed
youssef ali

Introduction

Before delving deeper into our project, it is important to first understand the concept of dimensionality reduction. This process plays a critical role in machine learning, as the dimensionality of a dataset can significantly impact the performance of learning algorithms . By selectively reducing the size of the data, the efficiency and accuracy of machine learning models can be greatly improved. As a result, dimensionality reduction has become an essential step in both supervised and unsupervised learning tasks.

The primary goal of dimensionality reduction is to transform high-dimensional data into a lower-dimensional representation that retains the most relevant and discriminative information. This transformation helps eliminate noise, remove redundant features, and highlight the most informative aspects of the data.

There are two main approaches to dimensionality reduction: feature extraction and feature selection. Feature extraction transforms the original data into a new, lower-dimensional feature space, aiming to preserve essential information while discarding irrelevant details . In contrast, feature selection focuses on identifying and selecting a subset of the original features that are most informative and relevant for the task at hand .

Each of these approaches encompasses various methods. Feature extraction techniques are generally divided into linear and non-linear methods. Feature selection methods, on the other hand, are typically categorized into three groups: filter methods, wrapper methods, and embedded methods.

Types of Dimensionality Reduction

Dimensionality reduction techniques fall into two broad categories:

1. Feature Selection

This involves selecting a subset of the original features that are most relevant to the predictive task, without transforming them. The goal is to retain important variables while discarding irrelevant or redundant ones.

- Filter Methods: Use statistical measures (e.g., correlation, information gain) to rank and select features independently of any model.
- Wrapper Methods: Use a predictive model to evaluate feature subsets and select the best performing combination.
- Embedded Methods: Perform feature selection during model training, such as LASSO regularization or feature importance from Random Forests.

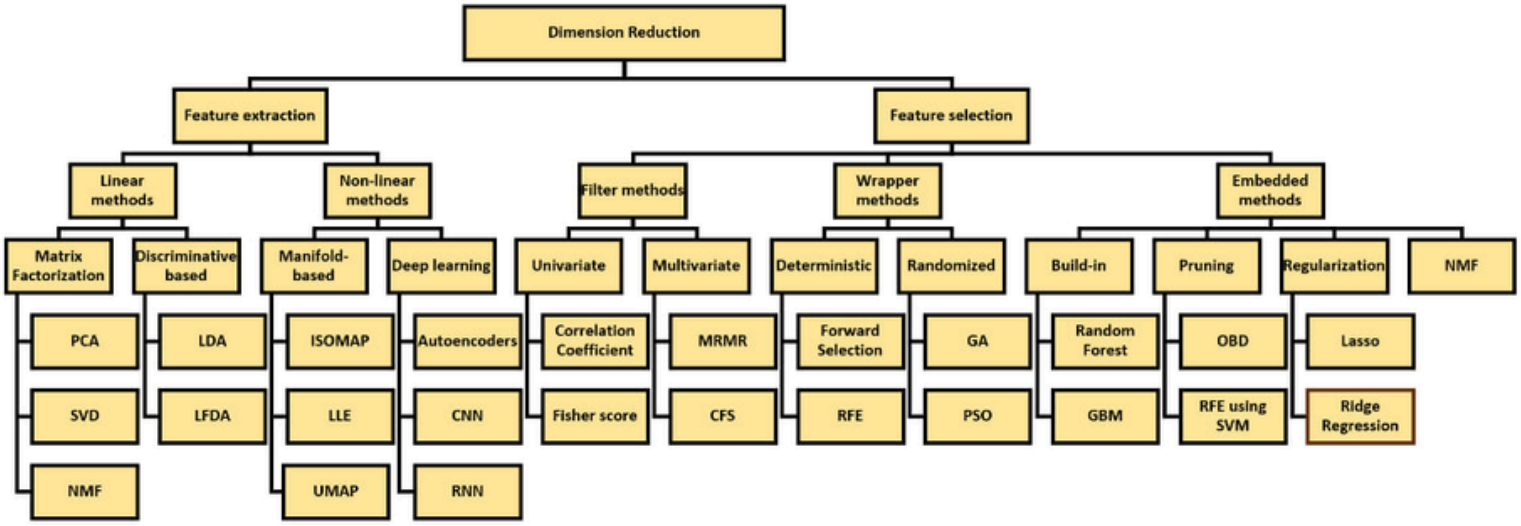
2. Feature Extraction (Feature Projection)

This transforms the original features into a new set of features in a lower-dimensional space, often by combining or projecting the data.

- Linear Methods: Principal Component Analysis (PCA) is the most common, projecting data onto directions of maximum variance to reduce dimensionality while preserving variance.
- Discriminant Analysis: Linear Discriminant Analysis (LDA) maximizes class separability.
- Nonlinear Methods: Techniques like t-SNE, UMAP, Kernel PCA, and autoencoders capture complex nonlinear structures in data for dimensionality reduction

Benefits of Dimensionality Reduction

- Reduces computational cost and storage requirements.
- Helps prevent overfitting by eliminating noise and irrelevant features.
- Improves visualization and interpretability of data.
- Enhances model generalizability and accuracy



Dimensionality Reduction algorithms

Dimensionality Reduction In Real Life Applications

Dimensionality reduction is widely applied in real-life scenarios across various fields to simplify complex, high-dimensional data while preserving essential information. Here are key real-world applications:

Image and Video Processing

- **Image Compression**: Techniques like Principal Component Analysis (PCA) reduce image size by compressing high-resolution images or video frames, improving storage efficiency and transmission speed without significant loss of quality. This is commonly used in social media platforms and mobile devices to handle large volumes of image data efficiently.
- **Face Recognition**: Dimensionality reduction helps extract meaningful facial features from high-dimensional image data, enabling faster and more accurate face recognition systems.
- **Video Summarization** and Retrieval: Methods such as Linear Discriminant Analysis (LDA) reduce video data dimensionality to facilitate efficient summarization, indexing, and retrieval

Text and Natural Language Processing (NLP)

- **Text Categorization and Email Filtering:** Dimensionality reduction simplifies large text datasets by reducing the number of features (e.g., words or topics), enabling efficient categorization and spam detection.
- **Topic Modeling and Sentiment Analysis:** Techniques like PCA, t-SNE, and UMAP help identify clusters and relationships in high-dimensional word embeddings, improving text classification and sentiment detection.
- **Language Understanding:** Reducing dimensionality in word vector spaces aids in better capturing semantic relationships and improving NLP model performance

---

Medical and Biological Research

- **Gene Expression Analysis:** Dimensionality reduction assists in identifying patterns and relationships in high-dimensional genomic data, helping researchers understand gene functions and disease mechanisms.
- **Medical Imaging:** Techniques like PCA and t-SNE reduce the complexity of MRI, CT, and other medical images, speeding up diagnosis by highlighting critical features while reducing data volume

---

Finance and Economics

- **Financial Data Analysis:** PCA and related methods analyze large-scale financial data (stock prices, interest rates) to uncover underlying trends, manage risks, and optimize investment portfolios.
- **Risk Management:** Reducing dimensionality helps in simplifying complex financial models, improving computational efficiency and decision-making

Recommender Systems

- **Personalized Recommendations:** Dimensionality reduction techniques such as Singular Value Decomposition (SVD) are used to simplify user-item interaction data, enabling faster and more accurate recommendations in platforms like Netflix and Amazon

---

Data Visualization and Exploratory Data Analysis (EDA)

- **High-dimensional data** is projected into 2D or 3D spaces using methods like PCA or t-SNE to facilitate visualization and pattern discovery, making complex datasets more interpretable for data scientists and decision-makers.
- **For example**, customer segmentation with dozens of features can be visualized as clusters in 2D space, revealing distinct user groups

---

In summary, dimensionality reduction is essential in real-life applications involving image and video processing, text analysis, medical research, finance, recommender systems, and data visualization. It improves computational efficiency, reduces noise, enhances interpretability, and enables better decision-making in handling complex, high-dimensional data

---

Dataset Used

Context : The Impact and Urgency of Addressing Diabetes

Diabetes is one of the most widespread chronic conditions in the United States, affecting over 34 million people and costing the economy nearly $400 billion annually. This serious disease impairs the body's ability to regulate blood sugar due to insufficient insulin production or ineffective insulin use. If left unmanaged, diabetes can lead to severe complications such as heart disease, vision loss, kidney failure, and amputations.

Despite its severity, diabetes is often undetected—1 in 5 individuals with diabetes and 8 in 10 with prediabetes are unaware of their condition. Type II diabetes, the most common form, is influenced by lifestyle and social factors like income, education, and access to care. While there is no cure, early diagnosis and proactive management through diet, exercise, and medical care can significantly reduce its impact.

Given the scale and consequences of diabetes, predictive models that help identify at-risk individuals early are vital tools for improving public health outcomes and targeting interventions where they are needed most.

Dataset Characteristics and Challenges for Dimensionality Reduction

The dataset used in this project primarily consists of binary and ordinal features, which introduces specific challenges when applying traditional dimensionality reduction techniques:

- Limited Variance: Binary variables can only take two values (typically 0 and 1), resulting in low variance across features. This limits their contribution to methods like PCA, which rely heavily on variance to identify meaningful directions in the data.
- Unclear Interpretability: When binary features are linearly combined into continuous components (as with PCA), the resulting features often lack intuitive interpretation. For example, a linear combination of yes/no variables does not easily translate into a concept that is meaningful or actionable.
- Nonlinear Relationships: The relationships between the features and the target (diabetes diagnosis) are often complex and nonlinear. Factors such as income, physical activity, and general health interact in ways that may not be captured well by linear methods.

As a result, while dimensionality reduction is still possible, classical linear techniques may not perform optimally. Instead, methods tailored for categorical data or those that can model nonlinear relationships—such as UMAP, or autoencoders—may provide better insight and performance

Dimensionality Reduction

1- Feature extraction :

Feature extraction is a dimensionality reduction technique that transforms the original high-dimensional dataset into a new, often lower-dimensional set of features. These newly generated features are designed to retain the most informative characteristics of the original data while reducing complexity. Unlike feature selection, which involves choosing a subset of existing features, feature extraction creates entirely new features by combining or projecting the original ones into a different mathematical space.

Key Aspects of Feature Extraction:

Transformation of Original Features

Feature extraction works by applying mathematical transformations to the original data to create new features. Common techniques include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Autoencoders (neural network-based)
- These methods aim to uncover hidden structure or patterns in the data that may not be obvious from the original features alone.

Dimensionality Reduction

The transformed feature set typically has fewer dimensions than the original dataset. This helps:

- Reduce computational complexity
- Minimize storage requirements
- Speed up model training and inference

Trade-off in Interpretability

One common drawback of feature extraction is the loss of interpretability. Since the new features are often mathematical combinations or projections of the original ones, they may lack clear physical or semantic meaning—making it harder to explain model decisions.

Enhanced Model Performance

By capturing the most important structure and variance in the data, feature extraction can lead to:

- Better predictive accuracy
- Improved generalization to unseen data
- Reduced risk of overfitting, especially in high-dimensional datasets

---

Used Algorithms

1- PCA

Principal component analysis, or PCA, reduces the number of dimensions in large datasets to principal components that retain most of the original information. It does this by transforming potentially correlated variables into a smaller set of variables, called principal components.
PCA is commonly used for data preprocessing for use with machine learning algorithms. It can extract the most informative features from large datasets while preserving the most relevant information from the initial dataset. This reduces model complexity as the addition of each new feature negatively impacts model performance, which is also commonly referred to as the “curse of dimensionality.”
By projecting a high-dimensional dataset into a smaller feature space, PCA also minimizes, or altogether eliminates, common issues such as multicollinearity and overfitting. Multicollinearity occurs when two or more independent variables are highly correlated with one another, which can be problematic for causal modeling. Overfit models will generalize poorly to new data, diminishing their value altogether. PCA is a commonly used approach within regression analysis but it is also leveraged for a variety of use cases, such as pattern recognition, signal processing, image processing, and more.

PCA Algorithm: Step-by-Step

Standardization
Scale the data so that each feature has zero mean and unit variance. This ensures that all variables contribute equally to the analysis, preventing features with larger ranges from dominating.

Covariance Matrix Computation
Calculate the covariance matrix of the standardized data to understand how variables vary together and identify correlations.

Compute Eigenvectors and Eigenvalues
Find the eigenvectors and eigenvalues of the covariance matrix.

- Eigenvectors represent directions (principal components) in the feature space.
- Eigenvalues indicate the amount of variance captured by each eigenvector.

Select Principal Components (Feature Vector)
Sort eigenvectors by their eigenvalues in descending order and select the top k eigenvectors that capture the majority of the variance. These form the new feature space.

Recast the Data
Transform the original data by projecting it onto the selected principal components, creating a reduced-dimensional representation.

PCA Performance On the Dataset :

Accuracy on Test Set Projection (using selected eval model): 0.7260
No. of Dimensions = 3

Drawbacks of PCA on the BRFSS Diabetes Dataset

Binary and Ordinal Nature of Features
Most features in your dataset are either binary (e.g., Smoker, HighBP, CholCheck) or ordinal (e.g., Education, Income), not continuous. PCA assumes continuous, normally distributed features. Applying PCA on binary/ordinal data may result in misleading principal components, since variance isn't a reliable measure of information for such features.

Limited Variance in Features
Binary features can only take two values (0 or 1), leading to low variance across the dataset. Since PCA relies on variance to identify informative components, this could cause PCA to underperform or produce components that do not reflect meaningful patterns.

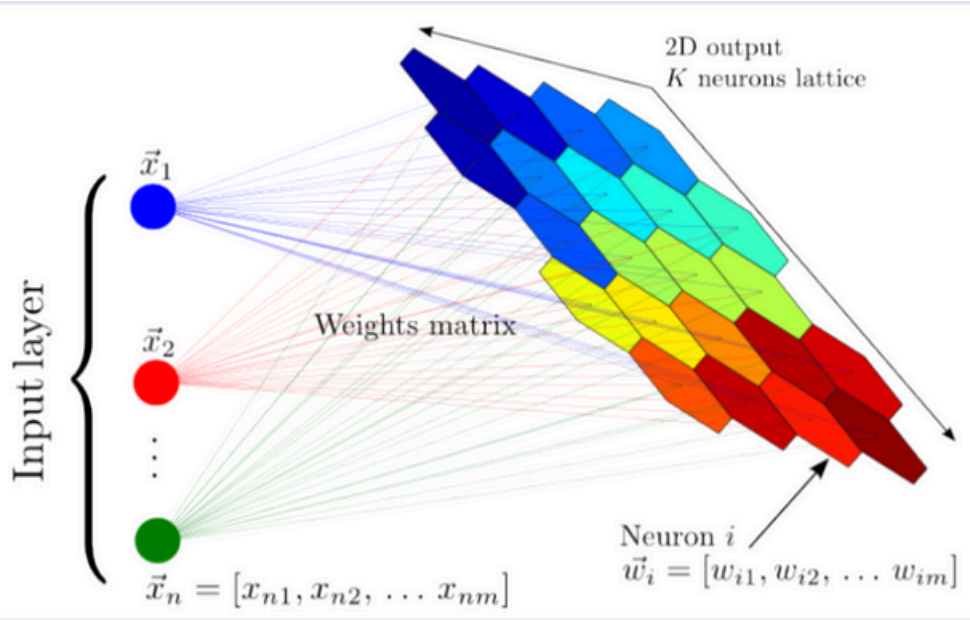
Loss of Interpretability
In medical and public health applications like diabetes prediction, interpretability is critical. PCA transforms original features into linear combinations, making it harder to understand how specific health factors (e.g., high blood pressure or physical activity) contribute to the prediction outcome.

Potential Violation of Linearity Assumption
PCA assumes that relationships among features are linear. However, health behavior and demographic data often exhibit complex nonlinear relationships, which PCA cannot capture effectively.

Risk of Information Loss
While PCA reduces dimensionality, it may discard components that, though low in variance, could be clinically significant. For instance, rare but critical features (e.g., Stroke or HeartDiseaseorAttack) might be underrepresented in the transformed space.

2-Self Organizing Map (SOM)

A SOM is an unsupervised learning algorithm trained using dimensionality reduction (typically two-dimensional), discretized representation of input space of the training samples, called a map. It differs from other ANN as they apply competitive learning and not the error-correction learning (like backpropagation with gradient descent). They use a neighborhood function to preserve the topological properties of the input space to reduce data by creating a spatially organized representation, and also helps to discover the correlation between data.

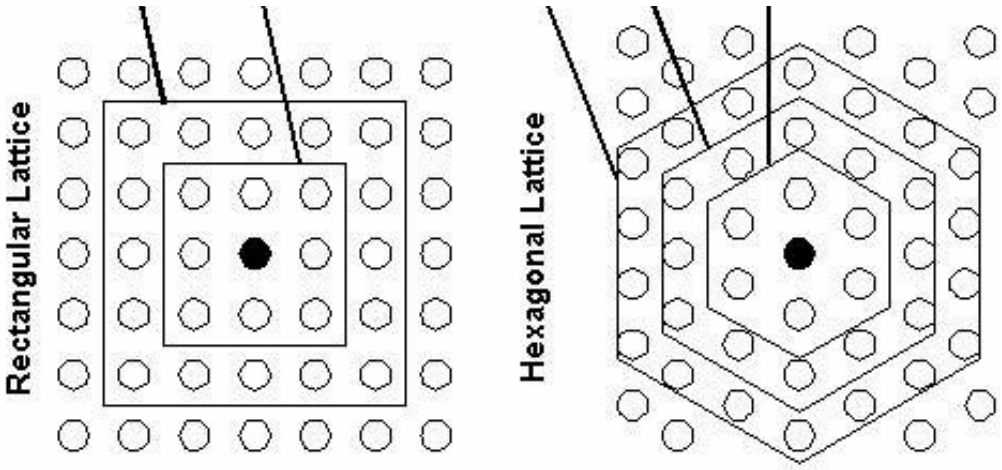


SOM

How SOM Works

Structure:

The SOM consists of a grid of nodes (neurons), typically arranged in a 2D hexagonal or rectangular lattice. Each node has an associated weight vector of the same dimension as the input data.



Training Process:

Initialization: Weights are initialized randomly or sampled from the data space (e.g., using principal components).
Competition: For each input vector, the algorithm finds the node whose weight vector is closest to the input (measured by Euclidean distance). This node is called the Best Matching Unit (BMU).
Cooperation: Nodes neighboring the BMU in the grid are identified within a neighborhood radius.
Adaptation: The BMU and its neighbors adjust their weight vectors to become more similar to the input vector. The amount of adjustment decreases with time and with distance from the BMU on the map.
Iteration: Steps 2–4 are repeated over many iterations, gradually shrinking the neighborhood radius and learning rate.

SOM Parameters

x and y (Grid Dimensions)

- Define the SOM grid size (e.g., 10x10 neurons).
- Determines output space dimension (e.g., 2D grid).
- Larger grids provide more granularity but require more training time and data

sigma (Neighborhood Radius)

- Controls the radius of the neighborhood function during training.
- Larger sigma means more neurons are updated per iteration → smoother, more global adaptation.
- Sigma typically decreases over training iterations

learning_rate

- Step size for weight updates during training.
- Higher values speed initial learning but can cause instability; typically decays over time.
- Influences how quickly neurons adjust to input data.

num_iteration (Training Iterations)

- Number of times the SOM weights are updated during training.
- More iterations typically improve convergence but increase training time.

SOM Performance On the Dataset :

Accuracy on Test Set Projection (using selected eval model): 0.5020
SIGMA= 1.0
LEARNING RATE =0.5
NO.ITER.=500
NO.GRIDS=(20,20,20)

Limitations and Drawbacks of SOM

Curse of Dimensionality

- SOM struggles with very high-dimensional data.
- The distance metrics used (e.g., Euclidean) become less meaningful as dimensions grow.
- **Feature scaling and dimensionality reduction before SOM is often needed.**

Choosing Grid Size and Topology

- Deciding the appropriate SOM grid size (number of neurons) and shape is not straightforward.
- Too small grids can oversimplify data structure; too large grids increase computation and may overfit.

Slow Training and Scalability Issues

- Training SOM can be slow for large datasets or very large grids because it updates many neurons per input.
- It doesn't scale well with big data without optimization or parallelization.

No Guarantee of Global Optimum

- SOM uses heuristic updates and random initialization; it may get stuck in local minima.
- Results can depend on initialization and parameter choices.

Interpretability and Visualization Limitations

- SOM results (mapping high-dimensional data to 2D/1D) may be hard to interpret meaningfully.
- The mapping can distort some distances or cluster relationships.

Sensitive to Parameter Settings

- Performance strongly depends on proper tuning of parameters like learning rate, sigma, iterations.
- Poor tuning can lead to poor clustering or convergence issues.

Differential Evolution

Differential Evolution (DE) is a population-based stochastic optimization algorithm used for solving complex optimization problems over continuous domains. It belongs to the family of evolutionary algorithms and is known for its simplicity, robustness, and ability to handle non-differentiable, nonlinear, and multimodal objective functions. DE operates through mechanisms inspired by natural evolution, including mutation, crossover, and selection, to iteratively improve candidate solutions towards an optimal or near-optimal solution.

SOM

How Differential Evolution Works

Structure:
DE maintains a population of candidate solutions, each represented as a vector in the search space. The size of the population is fixed throughout the process and typically depends on the problem dimension.

Training Process:

- **Initialization:**
- The initial population is generated randomly within the defined bounds of the problem's sea Each individual vector corresponds to a potential solution.
- **Mutation:**
- For each individual in the current population, a mutant vector is generated by adding the w difference between two randomly selected population vectors to a third vector. This step in variation and guides the search process.
- **Crossover:**
- A trial vector is created by mixing components of the mutant vector and the current individ a crossover probability. This operator increases diversity in the population.
- **Selection:**
- The trial vector is evaluated using the objective function. If the trial vector yields a better fi than the current individual, it replaces the individual in the next generation; otherwise, the c individual is retained.
- **Iteration:**
- Steps 2–4 are repeated over multiple generations until a termination criterion is met, such a maximum number of generations or satisfactory objective function value.

Differential Evolution Parameters

Population Size (NP):

- Defines the number of candidate solutions maintained. Larger populations enhance exploration but increase computational cost.

Mutation Factor (F):

- A scaling factor controlling the amplification of the differential variation. Typical values range from 0.4 to 1.0. A higher value encourages exploration; a lower value favors exploitation.

Crossover Probability (CR):

- Determines the probability with which components from the mutant vector are combined into the trial vector. Values range from 0 to 1, influencing the diversity of offspring.

Number of Generations (MaxGen):

- Specifies how many iterations the algorithm performs. More generations generally improve solution quality but require more computation time.

Limitations and Drawbacks of Differential Evolution

Parameter Sensitivity:

- DE's performance heavily depends on proper tuning of parameters such as population size, mutation factor, and crossover probability. Poor tuning may lead to slow convergence or premature stagnation.

Computational Cost:

- For complex problems requiring large populations or many generations, DE can be computationally expensive.

Lack of Guarantee for Global Optimum:

- As a heuristic method, DE does not guarantee finding the global optimum and may converge to local optima depending on the problem landscape.

Difficulty Handling Constraints:

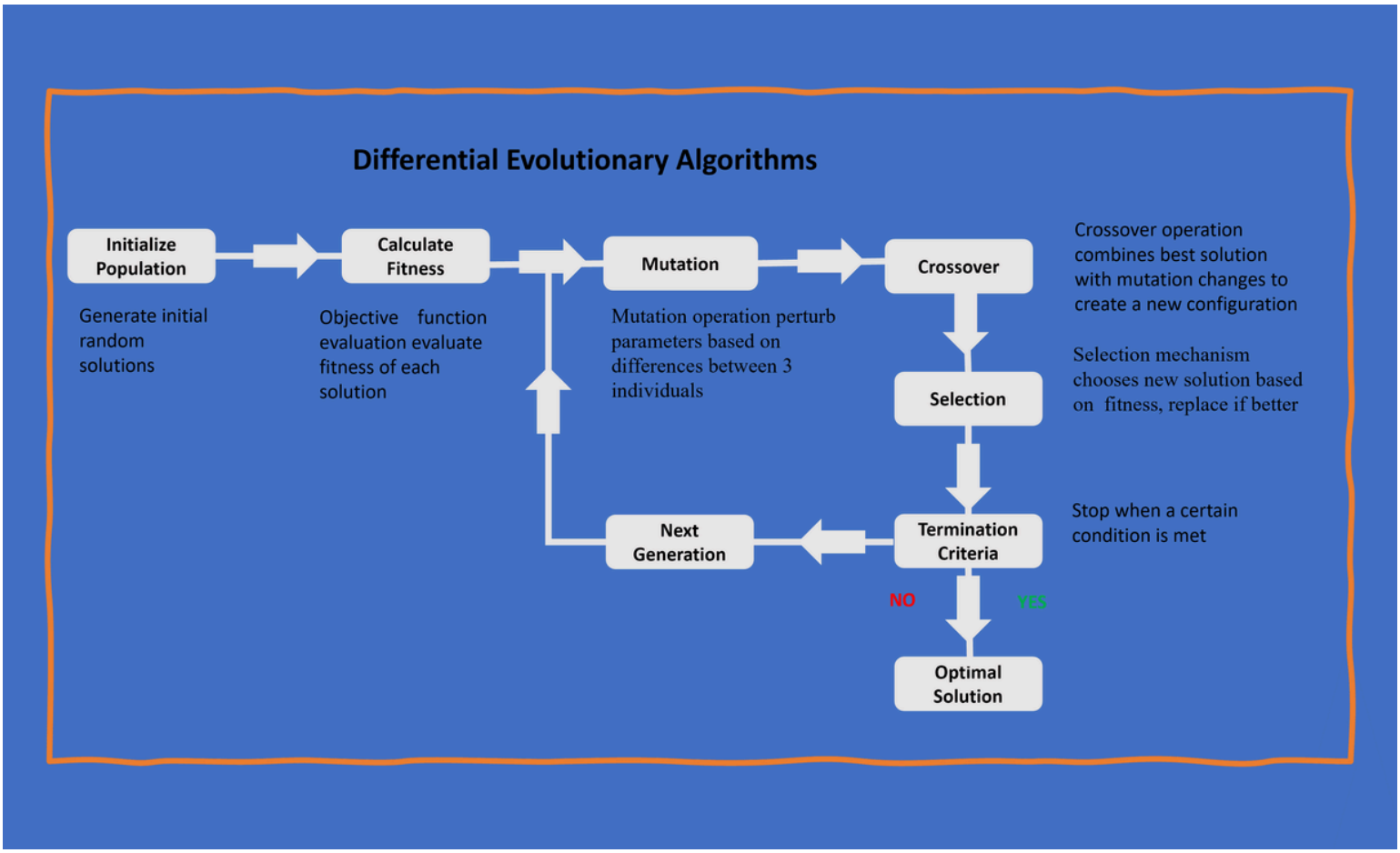
- DE requires additional mechanisms or adaptations to effectively handle constrained optimization problems.

P**opulation Diversity Maintenance:**

- Without proper parameter settings or diversity-enhancing techniques, the population may lose diversity over time, reducing the ability to explore the search space effectively.

Differential Evolution Performance On the Dataset :

Accuracy on Test Set Projection (using selected eval model): 0.7431
mutation rate=0.5
generations=30
crossover rate = 0.9



Memetic Algorithm

A Memetic Algorithm is a population-based metaheuristic optimization method that combines the global search capability of evolutionary algorithms with local refinement techniques. Inspired by the concept of memes and cultural evolution, MAs aim to enhance solution quality by integrating individual learning or improvement procedures within the evolutionary framework. This hybrid approach typically results in faster convergence and better solution accuracy for complex optimization problems.

How Memetic Algorithm Works

Structure:

Memetic Algorithm maintain a population of candidate solutions similar to other evolutionary algorithms. Each individual solution may undergo both evolutionary operations and local optimization to improve fitness.

Training Process:

Initialization:

- Generate an initial population of candidate solutions randomly or by heuristic methods within the problem's search space.

Selection:

- Select parent solutions from the population based on fitness to participate in reproduction.
- Genetic Operators (Crossover and Mutation):
- Apply crossover and mutation operators to parents to create offspring solutions, promoting diversity and exploration.

Local Search (Improvement):

- Each offspring solution undergoes a local search or refinement procedure, such as hill climbing or gradient-based optimization, to enhance solution quality before reinsertion.

Replacement:

- Offspring solutions, possibly improved by local search, replace less fit individuals in the population.

Iteration:

- Repeat steps 2–5 for a specified number of generations or until convergence criteria are met.

MA Parameters

Population Size:

- The number of candidate solutions maintained, affecting exploration and computational resources.

Crossover and Mutation Rates:

- Control the application frequency of genetic operators, influencing the balance between exploration and exploitation.

Local Search Frequency and Intensity:

- Defines how often and how intensively local optimization is applied to individuals, impacting convergence speed and computational cost.

Termination Criteria:

- Number of generations or fitness threshold to stop the algorithm..

Limitations and Drawbacks of Memetic Algorithm

Computational Complexity:

- The addition of local search increases computational cost significantly, especially if applied to many individuals or with complex local optimizers.

Parameter Tuning:

- The effectiveness of MA depends on careful tuning of evolutionary parameters and local search settings, which can be problem-specific and time-consuming.

Risk of Premature Convergence:

- Excessive local search or insufficient population diversity may lead to convergence to suboptimal solutions.

Design Complexity:

- Integrating appropriate local search methods with evolutionary operators requires expertise and can be non-trivial.

Scalability Issues:

- For very large or high-dimensional problems, the computational overhead of local search may limit scalability.

Memetic Algorithm Performance On the Dataset :

Accuracy on Test Set Projection (using selected eval model): 0.7404
mutation rate=0.17
generations=20
population size=20
crossover rate = 0.74
local search rate =0.3
tournament size = 5

Initialization and Evaluation Framework

In this framework, candidate solutions are represented as linear projection matrices used to reduce the original high-dimensional feature space to a lower-dimensional representation. The process begins by initializing these projection matrices with random values. Dimensionality reduction is achieved through linear projection, where the data is projected onto a subspace defined by each candidate matrix.

All input data is standardized using a standard scaler, which transforms each feature to have a mean of zero and a standard deviation of one. This normalization step ensures that all features contribute equally to the model training and projection process, preventing dominance by features with larger magnitudes.

During the optimization phase, each candidate projection matrix is evaluated by projecting the training and testing datasets into the reduced space and training a Logistic Regression model on the projected training data. Logistic Regression is chosen due to its simplicity, efficiency, and effectiveness in binary and multiclass classification tasks. The classification accuracy on the projected test data serves as the fitness score of each candidate, guiding the search process.

To ensure fairness and reproducibility, all optimization runs are conducted under identical computational environments and resource constraints.

Once the optimization process is complete and the best-performing projection matrix is selected, a final evaluation is carried out using a Random Forest classifier. Random Forest, known for its robustness and ability to capture complex, nonlinear relationships, provides a more comprehensive assessment of how well the learned projection generalizes to a more powerful ensemble model. This two-step evaluation—first with Logistic Regression during search and then with Random Forest post-optimization—ensures both efficient optimization and robust final validation.

References

https://www.ibm.com/think/topics/dimensionality-reduction
https://en.wikipedia.org/wiki/Dimensionality_reduction
https://nexocode.com/blog/posts/dimensionality-reduction-techniques-guide/
https://www.ibm.com/think/topics/principal-component-analysis
Multivariate Statistical Data Analysis- Principal Component Analysis (PCA)[Sidharth Prasad Mishra, Uttam Sarkar, Subhash Taraphder, Sanjay Datta, Devi Prasanna Swain, Reshma Saikhom, Sasmita Panda and Menalsh Laishram]
https://en.wikipedia.org/wiki/Self-organizing_map
https://www.latentview.com/blog/self-organizing-maps/
Application of self-organizing maps to genetic algorithms[S. Kan, Z. Fei & E. Kita]
https://www.turing.com/kb/guide-to-principal-component-analysis
https://en.wikipedia.org/wiki/Differential_evolution
https://en.wikipedia.org/wiki/Memetic_algorithm
Ant colony Optimization Algorithms : Introduction and Beyond[Anirudh Shekhawat Pratik Poddar Dinesh Boswal]
https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms
https://onlinelibrary.wiley.com/doi/10.1155/2021/5594267 Exploring and Implementing Non – Negative Matrix Factorization using Particle Swarm Optimization