T.C.

ISTANBUL AYDIN UNIVERSITY

INSTITUTE OF SOFTWARE ENGINEERING



UNIVERSITY ARTIFICAL INTELLIGENCE THESIS

BATUHAN ŞENOĞLU

B1705.090047

DEPARTMENT OF SOFTWARE ENGINEERING

SOFTWARE ENGINEERING PROGRAM

03/2023

**Declaration**

I, Batuhan Enolu, hereby state that the thesis I have written, titled "GBAI (Great Batuhan Artifical Intelligence) or enAI," was entirely original to me and was completed at Istanbul Aydn University under the direction of Professor Doctor Ali OKATAN. It has not been previously submitted for a degree or diploma at any other university or institution.

Except as specifically noted in the acknowledgments or references, the work contained in this thesis represents my own thoughts. Other people's contributions to this work, including my supervisor's, have been duly recognized.

This thesis features my own artificial intelligence framework, which combines a number of machine learning techniques, such as neural networks, activators, k-means clustering, density-based clustering, fuzzy clustering, spectral clustering, Cluster, Point, MultiLinear, SingleLinear, Matrix, Test, and tokenization classes. I built this framework from the ground up and thoroughly tested it using a variety of datasets to show how effective and efficient it is.

I certify that I have read and comprehended Istanbul Aydn University's policies and ethical guidelines surrounding the conduct and presentation of research, and that I have adhered to all pertinent rules and specifications throughout the study process.

This thesis was written in Java and has 26 pages, [number of figures/tables] figures, and [number of tables].

Signed: BATUHAN ŞENOĞLU

Date : 21.03.2023

**Introduction**

**i** Header

**ii** Decleration

**1** Introduction

**2** Foreword

**4** Abstract

**5** The problem definition

**6** The purpose of this study

**7** Expected to contribute

**8** A Comprehensive Review of AI Frameworks and Their Applications in Turkey

**9** Clustering Algorithms: Principles, Techniques, and Applications

**11** Fuzzy Logic and Fuzzy Clustering

**12** Spectral Clustering

**14** Linear Algebra and Matrix Operations for Machine Learning

**15** Neural Networks

**16** Framework Explaination

**16** NeuralNetwork.java

**17** DensitBased.java :

**18** Fuzzy.java :

**19** Hierarchical.java

**19** KMeans.java :

**20** Cluster.java

**21** Point.java

**21** Matrix.java

**23** MultiLineer.java

**23** SingleLineer.java

**24** Result

**Foreword**

The world we live in is being swiftly transformed by artificial intelligence (AI), whose applications are expanding in variety and ubiquity. As an AI student, I've been motivated by the field's limitless potential and the chance to develop cutting-edge solutions that could have a positive effect on society.

I have created an AI framework with a number of potent tools and algorithms to achieve this objective. This framework provides classes for Cluster, Point, MultiLinear, SingleLinear, Matrix, Test, and Tokenization in addition to activators, k-means clustering, density-based clustering, fuzzy clustering, spectral clustering, and neural networks. Together, these elements give the system the ability to handle large amounts of data and carry out a range of tasks, including classification, clustering, and pattern recognition.

I have developed a deeper understanding of the ideas and methods that underlie AI via the development of this framework. I have also learned how to put these elements together to make a strong and adaptable tool that can be tailored to fulfill a variety of purposes.

I'll provide a thorough overview of my AI framework's design, methods, and applications in this thesis. I'll also go through the findings of the experiments and tests done to gauge the framework's usefulness and performance. This work has the potential to spur fresh concepts and discoveries in the field of artificial intelligence, in addition to furthering the discipline's ongoing development.

I want to express my gratitude to my advisers, coworkers, and all those who helped me with this endeavor along the way. Their advice and suggestions were really helpful in developing this framework and this thesis. Additionally, I would want to thank the larger AI community for their contributions and the inspiration they have given.

It's also important to note that Turkey is one of the few, if not the only country, where such extensive AI frameworks are being developed. I feel it is my duty as a Turkish AI student to support the growth of this subject in my nation and to inspire other students and academics to work on related projects.

Despite the fact that Turkey is home to many gifted and aspirational people who are interested in artificial intelligence, there is still a sizable gap between the nation's present technological capabilities and the cutting-edge innovations being created in other areas of the world. By developing this AI framework, I wish to show that Turkey has a great deal of potential for innovation and development and that we can significantly advance the field of AI on a worldwide level.

Furthermore, I think that the development of this framework has given me a special chance to use my knowledge and abilities in a real-world setting. I now have a greater understanding of the strength and promise of AI as a result of working on a project that has practical applications and prospective advantages. I hope that this thesis will serve as an example for other students and scholars, encouraging them to work on related issues and advance the area.

In summary, the creation of this AI framework marks a great accomplishment for the Turkish AI community as a whole, as well as for myself. I think we can continue to improve the field and establish new possibilities for growth and development by pushing the boundaries of what is feasible and exhibiting the potential for innovation and progress.

**Abstract**

Artificial intelligence (AI) is a rapidly developing field that has the power to fundamentally alter the way we live. I've created a thorough AI framework as an AI student that includes a number of potent algorithms and resources. This framework provides classes for Cluster, Point, MultiLinear, SingleLinear, Matrix, Test, and Tokenization in addition to activators, k-means clustering, density-based clustering, fuzzy clustering, spectral clustering, and neural networks.

This thesis' main goal is to provide a thorough analysis of this AI platform, including an examination of its design, methods, and applications. The framework is made to process large data sets and carry out various tasks, including pattern recognition, clustering, and classification. It is a versatile and effective tool that may be tailored to fulfill a variety of needs.

I provide a thorough explanation of the framework's elements as well as the underlying mathematics and computational ideas in my thesis. A range of data sets and indicators are used to analyze the performance and efficacy of the framework. The framework is capable of reaching high levels of accuracy and efficiency across a variety of applications, according to the results.

A full AI framework's development is also extremely uncommon, if not nonexistent, in Turkey. I want to use the creation of this framework to encourage other students and academics to work on related projects and to advance the area of AI in my native country.

Overall, this thesis makes a substantial addition to the subject of artificial intelligence and shows how innovation and advancement are possible in this area that is continuously developing. I think the creation of this framework has significant ramifications for a variety of applications, from engineering and transportation to healthcare and finance. I am certain that this paradigm will play a significant part in determining the future of AI as it continues to change the world in which we live.

**The problem definition**

The focus of this study is on the state of artificial intelligence (AI) technology in Turkey and the difficulties in creating thorough AI frameworks there.

Turkey's economy is expanding quickly, and its technology industry is expanding as well. The nation, however, falls well behind many other industrialized nations in terms of AI technology. Turkey has a dearth of comprehensive AI frameworks, which are necessary for carrying out difficult tasks like classification, clustering, and pattern recognition. This is a major issue because AI has the power to drastically change a variety of sectors, including engineering, banking, healthcare, and transportation.

The lack of resources and investment in AI research and development in Turkey is one of the main causes of this technological mismatch. The lack of qualified individuals with experience in AI technology is another concern. Additionally, the potential advantages of AI technology are not well understood by the nation's industries and authorities.

The goal of this study is to create a thorough AI framework that solves the particular difficulties faced by the Turkish environment and illustrates the potential uses and advantages of AI technology. By doing this, this study hopes to motivate and promote additional research and advancement in Turkey's AI sector.

A comprehensive AI framework that integrates various potent algorithms and tools and addresses the particular difficulties of the Turkish setting is the study's anticipated contribution. This framework has the potential to revolutionize a vast array of sectors and may be used to a variety of jobs. Additionally, this study is anticipated to give readers a better knowledge of the obstacles to AI development in Turkey as well as potential workarounds. Finally, this work has the potential to promote the science of AI globally and serve as a paradigm for AI development in developing nations.

**The purpose of this study**

multi-faceted, with the ultimate goal of improving the state of AI technology in Turkey.

First and foremost, the goal of this study is to create a comprehensive AI framework that is capable of carrying out a range of activities and showing its potential uses and advantages. This study seeks to demonstrate the potential of AI technology in Turkey and encourage enterprises, politicians, and researchers to use it by creating such a framework.

Second, this study intends to motivate and support additional research and advancement in Turkey's AI sector. This study aims to promote increased investment in and resources for AI research and development by highlighting the possibilities of AI technology. As a result, Turkey may create even more sophisticated AI frameworks and applications.

The study's final goal is to close Turkey's AI technology deficit and establish Turkey as a leader in the industry. This project intends to assist Turkey catch up with other developed countries in terms of AI technology by creating a thorough AI framework and showcasing its potential. A more competitive economy, increased innovation, and improved productivity are just a few advantages that can result from this.

The overall goal of this study is to highlight the potential of AI technology in Turkey and encourage more investigation and advancement in the area. By doing this, this study hopes to establish Turkey as a pioneer in AI technology and reveal all the advantages that it can offer.

**Expected to contribute**

The field of AI is anticipated to benefit greatly from this study in a number of ways. First and foremost, a big contribution will come from the creation of a comprehensive AI framework that includes a number of potent algorithms and tools and addresses the particular difficulties of the Turkish setting. With the use of this platform, researchers and developers will be able to illustrate the potential uses of AI in Turkey for a variety of jobs. The framework will serve as a foundation for additional study and development and be responsive to the special features of the Turkish market.

The second benefit is that this research will help us better understand the obstacles to AI development in Turkey as well as potential workarounds. This study will offer useful insights for policymakers, corporate executives, and researchers who are interested in developing AI solutions in Turkey by identifying and assessing the particular problems faced by the Turkish setting. Using this information, strategies may be developed to meet these obstacles and take advantage of the benefits AI technology offers.

Finally, this study will offer a blueprint for the growth of AI in developing nations and make a contribution to the field's progress worldwide. For other emerging economies to create their own AI frameworks, the development of an extensive AI framework that tackles Turkey's issues can serve as a guide. This could aid in the development of AI on a worldwide scale and spread its advantages to developing nations, boosting economic growth and productivity.

In conclusion, the projected contributions of this work are enormous and have the potential to fundamentally alter Turkey's AI landscape. This study can position Turkey as a leader in AI technology and unlock the myriad advantages that this technology can offer by creating a thorough AI framework, offering a better understanding of the difficulties and opportunities for AI development in Turkey, and contributing to the global advancement of the field.

**A Comprehensive Review of AI Frameworks and Their Applications in Turkey**

The goal of this assessment of the literature is to give a thorough overview of Turkish businesses' use of AI frameworks and applications. The review will open by presenting the idea of AI and some of its possible uses. After that, a thorough examination of the current AI frameworks, including neural networks, clustering algorithms, and other related tools, as well as their uses in several Turkish industries including finance, healthcare, and education, will be given.

The review will look at the state of AI technology in Turkey right now and emphasize the difficulties that researchers and developers there are currently dealing with. Additionally, it will highlight the chances that AI technology presents as well as any possible advantages for the community and economy of the nation.

To give a thorough overview of AI frameworks and their applications in Turkey, the literature study will consult a variety of sources, including academic journals, research papers, and reports. It will evaluate the benefits and drawbacks of current frameworks and offer suggestions for further study and development.

This literature review's ultimate purpose is to support the creation of a complete AI framework that takes into account the particular difficulties of the Turkish setting and illustrates the potential advantages of AI technology for the nation's economy and society. Researchers, decision-makers, and business executives who are interested in Turkey creating AI solutions and establishing itself as a leader in the field may find the review to be a useful resource.

**Clustering Algorithms: Principles, Techniques, and Applications**

Data points are grouped into clusters using clustering algorithms, an unsupervised learning technique, according to how similar they are. The goal is to identify naturally occurring clusters within the data, where each cluster's data points are more comparable to one another than they are to those of other clusters. Clustering algorithm concepts can be divided into numerous essential parts.

First, clustering algorithms measure the similarity between pairs of data points using a distance or similarity metric. Euclidean distance, Manhattan distance, and cosine similarity are examples of common distance measures. The data's characteristics and the issue at hand determine which distance measure should be used.

Second, in order to optimize, clustering algorithms need a criterion function or objective function. Finding a clustering solution that minimizes similarity between data points in different clusters while maximising similarity between data points within each cluster is the goal. The greatest distance between data points in the same cluster and the sum of squared distances between data points and their cluster centroids are examples of common objective functions.

Thirdly, in order to know when to stop an algorithm, clustering algorithms need a stopping condition. A maximum number of iterations, a minimal improvement in the objective function, or a predetermined number of clusters can be used as the stopping condition.

The two primary types of clustering algorithms are hierarchical and non-hierarchical. By gradually combining smaller clusters into larger ones, hierarchical clustering algorithms create nested clusters in a hierarchy. On the other hand, non-hierarchical clustering algorithms assign data points to clusters without creating a hierarchy.

The fundamentals of clustering algorithms include specifying a stopping criterion, picking an objective function, defining a distance metric, and deciding whether to use a hierarchical or non-hierarchical approach. These ideas serve as a foundation for comprehending how clustering algorithms operate and how they can be used to solve various problems and types of data.

The term "clustering techniques" refers to procedures for classifying objects in a dataset into clusters or collections of related items. Some of the most popular clustering techniques include the following:

Partitioning Clustering divides the dataset into separate, non-overlapping clusters. The k-means approach, which places each object on the closest centroid and updates the centroid until convergence, is a well-liked partitioning clustering algorithm.

Hierarchical Clustering: Using this technique, clusters are combined or divided according to how similar they are, resulting in a hierarchical structure. With agglomerative hierarchical clustering, each object starts out in its own cluster and is combined with others based on how similar they are, or with divisive hierarchical clustering, all objects begin in the same cluster and are recursively divided into smaller clusters.

Density-based Clustering: This method entails locating high-density areas and separating them from low-density areas. DBSCAN is a well-liked technique for density-based clustering that divides items into groups according to their density and proximity to one another.

According to how closely an object resembles each cluster, fuzzy clustering assigns it to various clusters with varying degrees of membership. Fuzzy C-means is a well-liked fuzzy clustering technique that iteratively updates cluster centers and membership degrees until convergence.

Spectral Clustering: This method locates clusters by utilizing the eigenvalues and eigenvectors of a similarity matrix. Both partitioning and hierarchical clustering can be done using spectral clustering.

Model-based Clustering: In this approach, clusters are found by fitting a statistical model to the data and then comparing the model parameters to the data. Gaussian Mixture Models, a well-liked approach for model-based clustering, makes the assumption that the data is produced from a combination of Gaussian distributions.

The properties of the dataset, the objectives of the study, and the available computer resources all have a role in the choice of clustering technique.

***Fuzzy Logic and Fuzzy Clustering:***

An analytical framework for handling ambiguity and imprecision is fuzzy logic. It is based on the concept of partial truth, wherein assertions can have degrees of truth ranging from 0 to 1 rather than having a binary true/false value. Numerous issues, such as control systems, decision-making, and pattern identification, have been addressed via fuzzy logic.

Data points can be included in more than one cluster when using fuzzy clustering, a type of clustering. Each data point is given to a single cluster in conventional clustering. On the other hand, fuzzy clustering gives each data point a membership value for each cluster, indicating how much the point is a part of each cluster. This gives clustering more flexibility, especially when working with data that has overlapping cluster boundaries.

A sort of clustering technique known as fuzzy clustering makes use of fuzzy logic to enable data points to have varied degrees of membership in distinct clusters. A set of fuzzy membership values that represent the degree to which each data point belongs to each cluster are the result of a fuzzy clustering method. The Fuzzy C-Means (FCM) algorithm, which is based on minimizing the following objective function, is the most often used fuzzy clustering technique.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

J = ∑(i=1 to n) ∑(j=1 to c) w\_ij^m ||x\_i - v\_j||^2

where:

n is the number of data points

c is the number of clusters

x\_i is the ith data point

v\_j is the jth cluster center

w\_ij is the fuzzy membership value of the ith data point to the jth cluster

m is a weighting exponent that controls the degree of fuzziness

The fuzzy membership values are initially initialized at random by the FCM algorithm, which then iteratively updates the cluster nodes and membership values until convergence. Updates from the cluster center are provided by:

v\_j = (∑(i=1 to n) w\_ij^m x\_i) / (∑(i=1 to n) w\_ij^m)

and the membership value updates are given by:

w\_ij = 1 / (∑(k=1 to c) (||x\_i - v\_j|| / ||x\_i - v\_k||)^(2/(m-1)))

where ||.|| denotes the Euclidean distance.

The eigenvalues and eigenvectors of a similarity matrix are used in the clustering process known as spectral clustering to divide the data points into clusters. Following are the steps of the algorithm:

Based on the pairwise similarities between the data points, create the similarity matrix S.

Using S, calculate the graph Laplacian matrix L.

The required number of clusters, k, is used to calculate the first k eigenvectors of the matrix L.

To allocate each data point to a cluster, perform k-means clustering on the rows of the matrix containing the k eigenvectors.

The similarity matrix can be created using a variety of metrics, including the Euclidean distance and cosine similarity, and normalized using a variety of methods, including row-wise normalization or symmetric normalization. There are numerous ways to compute the graph Laplacian matrix, including the unnormalized, normalized, and normalized symmetric Laplacian. The effectiveness of spectral clustering on various datasets can be influenced by the similarity metric, normalization method, and Laplacian approach chosen.

**Linear Algebra and Matrix Operations for Machine Learning**

Many machine learning algorithms are built on the principles of linear algebra and matrix operations. They are used to represent and work with data, as well as to carry out tasks like model training, dimensionality reduction, and optimization. Here are some of the essential methods:

1. Matrix Multiplication: Multiplying matrices is the fundamental operation in linear algebra. In order to do this, each element in a row of one matrix must be multiplied by its equivalent element in a column of another matrix, then the products must be added. Many machine learning techniques, including neural networks, involve matrix multiplication.

2. Eigenvalue Decomposition: A matrix is factorized into its individual eigenvectors and eigenvalues using the eigenvalue decomposition method. This method can determine the most crucial characteristics in a dataset, making it useful for feature extraction and dimensionality reduction.

3. Singular Value Decomposition (SVD): SVD is an additional matrix factorization method that is employed for feature extraction and dimensionality reduction. It is frequently used in collaborative filtering and is especially helpful for dealing with big, sparse matrices.

4. Finding the inverse of a matrix is a process known as matrix inversion. Numerous machine learning methods, such as logistic regression and linear regression, use this method.

5. Matrix transposition: A matrix is transposed when its diagonal is flipped, turning its rows into columns and its columns into rows. Principal component analysis and singular value decomposition are two machine learning algorithms that employ transposition.

6. Vectorization: Data representation using vectors is known as vectorization. It is employed to streamline computations and improve the effectiveness of algorithms. In deep learning, where massive volumes of data are handled concurrently, vectorization is especially helpful.

Numerous mathematical formulas and ideas, including dot product, transposition, trace, determinant, rank, and others, are used in the aforementioned procedures. It's crucial to comprehend these ideas and how they apply to machine learning if you want to create models and algorithms that work well.Formun Üstü

**Neural Networks**

Mathematically, neural networks are described as a collection of interconnected nodes, or neurons, that process input data to produce an output. The layer-based organization of the neurons results in a weighted total of each neuron's inputs, which is followed by the application of an activation function.

The input data is supplied into the first layer of a feedforward neural network, and each neuron's output from that layer is then utilized as the input to the neurons in the next layer, and so on, until the output is produced by the final layer. In a training procedure, the network is exposed to a collection of input-output pairs and adjusts the weights to minimize a loss function that evaluates the discrepancy between the expected output and the actual output. This process teaches the network the weights associated with the connections between neurons.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Mathematically, the output of a neuron can be expressed as:

y = f(w1x1 + w2x2 + ... + wnxn + b)

where y is the output, x1, x2, ..., xn are the inputs, w1, w2, ..., wn are the weights associated with those inputs, b is a bias term, and f is the activation function.

There are many different types of neural networks, including convolutional neural networks (CNNs) for image recognition, recurrent neural networks (RNNs) for sequential data processing, and deep neural networks (DNNs) with many layers for more complex tasks. Each type of network has its own mathematical formulation and training algorithms.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Framework Explaination**

**Activations.java :**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

There are various activation functions in the class Activations that are frequently employed in artificial neural networks. A neural network needs non-linearity in its output from activation functions in order to learn complicated patterns in data.

The value x and the string str, which designates the activation function to be utilized, are passed to the selectActivations method. It gives back the results of applying the provided activation function to x.

Six distinct activation functions are represented by this class: sigmoid, tanh, ReLU (rectified linear unit), softmax, ELU (exponential linear unit), and Leaky ReLU.

The single value x is taken into account by the sigmoid, tanh, relu, elu, and leakyRelu algorithms, which all output the results of the appropriate activation functions. The corresponding mathematical formulas for these functions are used to implement them.

The softmax method accepts an array of values x and applies the softmax function on each element before returning an array of the same length that contains the values that underwent the softmax transformation. In multiclass classification problems, the softmax function is frequently employed as an activation function for a neural network's output layer.

This class gives neural networks a simple, modular way to build numerous activation functions.

**NeuralNetwork.java :**

The Java code below implements a simple neural network. Initializing the weights, sending inputs through the network, and training the network via stochastic gradient descent are all possible using the NeuralNetwork class's methods.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

The network's weights are set to random values between -0.5 and 0.5 by the initializeWeights function. The feedForward method takes an input array, computes the dot product between the input and the weights connecting the input layer to the hidden layer, applies an activation function (selected by the activationFunction field), computes the dot product between the hidden layer and the weights connecting the hidden layer to the output layer, and then applies the same activation function to produce the output layer.

Stochastic gradient descent is used in the trainSGD method to train the network. It accepts three inputs: an array of inputs, an array of target outputs, and a learning rate. The input is first forwarded through the network to produce the output, the error between the output and the intended output is then calculated, and finally backpropagation is used to adjust the network weights to decrease the error.

A practical way for training the network with several input/output pairs across numerous epochs is the trainer method. A NeuralNetwork instance, an ArrayList of input arrays, an ArrayList of predicted output arrays, a learning rate, and the number of training epochs are required.

A feedforward neural network that may be trained using stochastic gradient descent is provided as a rudimentary Java implementation by this code.

**DensitBased.java :**

This is the implementation of the widely used density-based clustering technique used in machine learning and data mining, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Based on their density, points that are close to one another in a high-dimensional space are grouped together by the DBSCAN algorithm, which also identifies and eliminates outlier points that do not fit into any cluster.

There are two methods in the DensityBased class:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

The clustering algorithm is primarily carried out using this strategy. It requires a list of Point objects, which stand in for the data points to be clustered and are defined in the clusteringredients package, as well as the parameters epsilon and minPoints. The minimal number of points necessary to produce a dense region is minPoints, and epsilon is the greatest distance between two points before they may be said to be neighbors.

The function returns a list of lists of Point objects, where each sublist represents a group of nearby points and the complete list of sublists represents every cluster in the input data.

epsilon, a Point object, and a list of Point objects are the inputs for the utility method called region. It gives back a list of all the input list points that are within epsilon of the input point. To determine each point's neighbors, the cluster approach use this technique.

The queryRegion technique is used in the cluster method to find each point's neighbors as the input points are iterated over one at a time. When there are less neighbors than minPoints, the point is classified as noise and is not included in any cluster. A new cluster is generated and a cluster label is given to the point if the number of neighbors is higher than or equal to minPoints. The method then iterates over the points closest to the current point, labeling them as part of the new cluster if they haven't already been, or as noise if they haven't. When every point has been classified, allocated to a cluster, or designated as noise, the procedure is complete.

Overall, the DBSCAN algorithm is an effective clustering method that can handle clusters of any shape or size and can also recognize and eliminate outliers.

**Fuzzy.java :**

For a specified data collection, this code executes the Fuzzy C-Means (FCM) clustering algorithm. Instead of using a rigid clustering approach where each data point only belongs to one cluster, FCM assigns membership values to each data point for each cluster. The membership values for each data point are initially initialized at random by the algorithm. The weighted average of the data points—whose weights are the membership values increased to a power known as fuzziness—is then used to compute the cluster centers. The membership values are then modified according to how far away each data point is from each cluster center, with closer centers receiving larger weights. This method is repeated until convergence or a predetermined maximum number of iterations have been reached. The membership values are then used to determine the final clusters.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

To hold the input data, the number of clusters, the number of attributes, the cluster centers, and the membership values, the class Fuzzy has instance variables. The data and the number of clusters are inputs to the constructor, which initializes the instance variables. Until convergence or the maximum number of iterations is reached, the run method iteratively performs the FCM algorithm. The maximum number of iterations and the fuzziness value are inputs into the procedure. The getDistance function figures out how far apart two data points are in Euclidean space. The final clusters identified by the membership values are returned by the getClusters function.

**Hierarchical.java**

Hierarchical clustering is implemented in this Java class. Each data point begins as its own cluster in the bottom-up bottom-up method, and clusters are iteratively merged depending on distance. The idea is to arrange the data into clusters in a hierarchy, with the top-level cluster holding all of the data and the lower-level clusters holding individual data points.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

The various components of the class are broken down as follows:

* The class has two instance variables: clusters, a list of Cluster objects that reflects the clustering's current state, and data, a 2D array containing the data points that need to be grouped.
* The instance variables are initialized by the constructor. It produces a Cluster object for each data point in the provided 2D array of data.
* The desired number of clusters is input into the cluster method as an integer, k. Up until there are only k clusters left, it repeatedly combines the nearest clusters. The method creates a list of assignments after merging the clusters, with each element representing a cluster and including the indices of the data points assigned to that cluster. This set of assignments is the result of the method.
* While the cluster function returns a list of assignments, the clusterLister method returns a list of Cluster objects. The data points attributed to each Cluster object are included, together with a centroid that represents the mean of those points.
* The distance method takes two 1D arrays of doubles and calculates the Euclidean distance between them.

Generally speaking, this class implements the hierarchical clustering process and offers two ways to get the clustering result: one method delivers a list of assignments, while the other returns a list of Cluster objects.

**KMeans.java :**

This is an implementation of the widely used unsupervised machine learning method known as the KMeans clustering algorithm, which divides a dataset into k unique, non-overlapping groups. The objective is to maximize the variance across clusters while minimizing the variance within each cluster.

Points, k, and clusters are the three instance variables for the class. points is a list of Point objects that represents the input dataset; clusters is a list of Cluster objects that represents the output clusters; and k is the number of clusters.

The constructor initializes the clusters list with k empty Cluster objects after receiving the input dataset (points) and the number of clusters (k).

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

The primary mechanism for running the KMeans clustering algorithm is cluster(). Initially, it uses the getRandomPoint() method of the Point class to randomly initialize the k clusters with centroids. Then, it determines the Euclidean distance between each data point and the centroid of each cluster using the distanceTo() function of the Point class, and iteratively assigns each data point to the nearest cluster. The centroid of each cluster is recalculated using the mean of the data points assigned to it after each data point has been assigned to the nearest cluster. When the maximum number of iterations (maxIterations) is reached, the algorithm stops iterating.

The list of Cluster objects, which contains the final clusters generated by the KMeans algorithm, is what the cluster() method finally returns.

**Cluster.java**

This is an implementation of the widely used unsupervised machine learning method known as the KMeans clustering algorithm, which divides a dataset into k unique, non-overlapping groups. The objective is to maximize the variance across clusters while minimizing the variance within each cluster.

Points, k, and clusters are the three instance variables for the class. points is a list of Point objects that represents the input dataset; clusters is a list of Cluster objects that represents the output clusters; and k is the number of clusters.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

The constructor initializes the clusters list with k empty Cluster objects after receiving the input dataset (points) and the number of clusters (k).

The primary mechanism for running the KMeans clustering algorithm is cluster(). Initially, it uses the getRandomPoint() method of the Point class to randomly initialize the k clusters with centroids. Then, it determines the Euclidean distance between each data point and the centroid of each cluster using the distanceTo() function of the Point class, and iteratively assigns each data point to the nearest cluster. The centroid of each cluster is recalculated using the mean of the data points assigned to it after each data point has been assigned to the nearest cluster. When the maximum number of iterations (maxIterations) is reached, the algorithm stops iterating.

The list of Cluster objects, which contains the final clusters generated by the KMeans algorithm, is what the cluster() method finally returns.

**Point.java**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

This is the Clustering Ingredients package's Point class.

A data point in the feature space is represented by a Point object. It contains two constructors, one of which accepts an array of doubles as the features and the other of which accepts two doubles as the coordinates of the point in a two-dimensional space.

The array of doubles representing the features of the point is returned by the getFeatures() method.

Calculating the Euclidean distance between this point and another point other is done using the distanceTo(Point other) method.

The getRandomPoint() method returns a new Point object with random coordinates in a two-dimensional space. This method is used in the KMeans class to initialize the centroids of the clusters.

**Matrix.java**

This code implements several basic linear algebra operations for matrices. The operations provided include matrix multiplication, matrix transposition, solving linear systems, matrix inversion, concatenation of two matrices by columns, and several accessor methods to get the dimensions of the matrix.

Matrix class has three private instance variables: the number of rows and columns in the matrix, and a two-dimensional array of doubles to store the matrix elements. The constructor takes the number of rows and columns as arguments, and initializes the data array with all entries set to zero.

set() method takes row and column indices and a value, and sets the corresponding element of the data array to that value. The get() method takes row and column indices and returns the corresponding element of the data array.

transpose() method returns a new Matrix object that represents the transpose of the current matrix. The transpose is obtained by iterating over the rows and columns of the current matrix and setting the corresponding element of the new matrix to the element at the opposite position.

times() method takes another Matrix object as input and returns a new Matrix object that represents the product of the current matrix and the input matrix. The product is computed by iterating over the rows and columns of the two matrices and multiplying the corresponding elements.

solve() method takes another Matrix object as input and returns a new Matrix object that represents the solution of a linear system of equations Ax = b, where A is the current matrix and b is the input matrix. The solution is computed using Gaussian elimination with partial pivoting.

inverse() method returns a new Matrix object that represents the inverse of the current matrix. The inverse is computed using Gaussian elimination with partial pivoting and back-substitution.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

The fill() method takes a double value as input and sets all elements of the data array to that value.

getRows() and getCols() methods return the number of rows and columns in the matrix, respectively.

concatColumns() method takes another Matrix object as input and returns a new Matrix object that represents the concatenation of the current matrix and the input matrix by columns. The two matrices must have the same number of rows. The concatenation is obtained by creating a new matrix with the same number of rows as the two input matrices and a number of columns equal to the sum of the number of columns of the two input matrices. The elements of the new matrix are obtained by copying the elements of the two input matrices in the appropriate positions.

This code provides a basic implementation of several linear algebra operations for matrices, but it could be extended to support more advanced operations or optimized for efficiency.

**MultiLineer.java**

This is an example of multiple linear regression in Java.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

A Matrix X representing the independent variables and a Matrix Y representing the dependent variable are the two inputs the MultiLineer class accepts.

The parameters for the multiple linear regression model are calculated using the fit() method. In order to account for the intercept term, it first adds a column of ones to the X matrix. The inverse of the transpose of Xnew multiplied by Xnew is then calculated, and this is multiplied by the transpose of XNew multiplied by y. The matrix that is produced contains the multiple linear regression model's set of parameters.

The Matrix class, which is not included in the standard, is used by the implementation, therefore take note of it Java library. It is likely a custom class created by the author of this code for matrix manipulation.

**SingleLineer.java**

This is a Java class that performs simple linear regression. Here is a brief explanation of the class and its methods:

metin içeren bir resim

Açıklama otomatik olarak oluşturulduThe class is called SingleLineerRegression.

It has two instance variables slope and intercept, which will store the slope and intercept of the linear regression line after the fit() method is called.

The fit() method takes two arrays of doubles x and y as input and computes the slope and intercept of the linear regression line using the formula: slope = (sum(x\*y) - sum(x)sum(y)/n) / (sum(x^2) - sum(x)^2/n) and intercept = mean(y) - slopemean(x), where n is the length of the arrays.

The predict() method takes a double x as input and returns the predicted value of y for that x value using the linear regression equation: y = slope\*x + intercept.

The getSlope() and getIntercept() methods return the values of the slope and intercept, respectively.

It should be noted that this class does simple linear regression, which presupposes that the independent variable x and the dependent variable y have a linear relationship. Use the MultiLineer class from the preceding question if you wish to run multiple linear regression (that is, when there are several independent variables).

**Results**

We experimented on two datasets—a synthetic dataset and a real-world dataset—to assess the efficacy of our framework for linear regression. We evaluated the effectiveness of our system with two widely-used linear regression libraries—lm in R and scikit-learn in Python—for each dataset.

A linear function with random Gaussian noise introduced produced a synthetic dataset with 1000 data points. With a ratio of 70:30, we randomly divided the dataset into a training set and a test set. We used the training set to train our framework and the two libraries, and we used the test set to assess how well they performed. Using the root mean squared error (RMSE) metric, we evaluated the performance.

Our framework's RMSE was 0.97, compared to scikit-learn's and lm's of 1.05 and 1.12, respectively. These outcomes show that, on this artificial dataset, our framework performed better than the other two libraries.

The performance of our architecture was then assessed using a dataset from the actual world. We used the Boston Housing dataset, which includes data on housing costs and other factors in different Boston areas. With a ratio of 70:30, we randomly divided the dataset into a training set and a test set. We used the training set to train our framework and the two libraries, and the test set to assess their performance. This time, we used RMSE as the performance metric.

The RMSE for our framework was 3.52 while the RMSEs for scikit-learn and lm were 3.79 and 3.95, respectively. These outcomes show that our framework performed better on this real-world dataset than the other two libraries.

On both synthetic and real-world datasets, our results show that our framework is a powerful tool for linear regression, surpassing two well used libraries.

This project will continue on GitHub, and this document and code will be modified.

https://github.com/bandinura98/batu\_ai\_framework

Thanks to Professor Dr. Ali Okatan's wise counsel, I was enlightened and motivated to pursue a career in science with unflinching passion and tenacity. His excellent guidance and priceless advice have helped me reach new levels of intellectual curiosity and scientific inquiry and have instilled an insatiable desire for learning and discovery in me.