EXTREME LEARNING MACHINE

Extreme Learning Machine commonly referred to as ELM, is one of the machine learning algorithms introduced by Huang et al in 2006. This algorithm has gained widespread recognition in recent years, primarily due to its lightning-fast learning capabilities, exceptional generalization performance, and ease of implementation. This makes it awesome for businesses and researchers because they can get results fast and efficient way. It provides a significant contribution to fields like Image recognition, speech recognition, Natural language processing financial forecasting, medical diagnosis, social media analysis, and recommendation systems.

In this article, we will dive deep into the concept of an "Extreme learning machine" by explaining its architecture, training process, and application which are listed below in the table of contents.

What is ELM in Machine Learning?

In Deep learning, an Extreme Learning Machine (ELM) is a type of feedforward neural network utilized for tasks such as classifications and regression. ELM stands apart from traditional feedforward neural networks due to its unique training approach.

In ELM, the hidden layer's weights and biases are randomly initialized. However, these initial values are just starting points. The distinctive aspect of ELM lies in its ability to compute the output layer's weights using the Moore-Penrose generalized

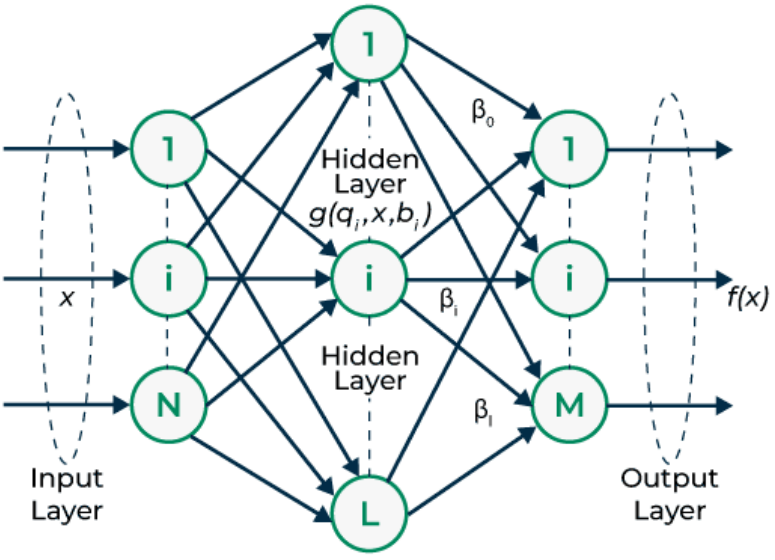
inverse of the hidden layer's output matrix. This approach enables ELM to learn from training data in a single step, setting it apart from traditional neural networks that often require iterative training procedures, such as backpropagation. It uses single-hidden layer feedforward neural networks (SLFN) instead of traditional feedforward neural networks. Thus it randomly selects hidden nodes and analytically finds their output weight. ELM's single-step training process makes it an efficient and versatile tool for a wide range of machine-learning applications.

Architecture of ELM

In this section, we are going to discuss the architecture of ELM which provides a detailed explanation of how ELM works in machine learning.

The architecture of ELM is very simple and straight forward which involves three segments which are listed below,

1. **Input layer**
2. **Hidden layer - Single hidden layer**
3. **Output layer**



1. Input layer

In ELM, the Input Layer is where the data enters the model. It's represented as a vector called X, which contains the input features.

$$X = [X[1], X[2], X[3], ..., X[N]]$$

In this representation, each **X[i]** corresponds to a specific feature or attribute of the data. N is the total number of features. The Input Layer is responsible for passing the data to the Hidden Layer for further processing.

Hidden layer-single hidden layer

The hidden layer of ELM is where random weights and biases are assigned. Let's denote the number of hidden neurons as L as per above Fig 1. The weights connecting the input features to the hidden neurons are represented by a weight matrix W of size (number of features, L). The value of N is a hyperparameter that needs to be set before training the neural network. The more hidden neurons there are, the more complex the neural network will be and the more accurate it will be at modeling complex functions. However, having too many neurons will lead to overfitting.

Each column in the weight matrix corresponds to the weights of a hidden neuron. The biases for the hidden neurons are represented by a bias vector b of size (L, 1). The second dimension of 1 is used to ensure that the bias vector is a column vector. This is because the dot product of the weight matrix W and input feature vector X results in a column vector, and adding a row vector (the bias vector) to a column vector requires that the bias vector be a column vector as well. The purpose of the bias term is to shift the activation function to the left or right, allowing it to model more complex functions.

The output of the hidden layer, often denoted as H, is calculated by applying the activation function g like linear regression concept by making element-wise to the dot product of the input features and the weights, adding the bias.

$$H = g(W * X + b)$$

2. Output layer

In ELM, the output layer weights are calculated using Moore-Penrose inverse of the hidden layer output matrix. This output weight matrix is denoted as beta. The output predictions, represented as f(x), are calculated by multiplying the hidden layer output H by the output weights beta:

$$f(x) = H * beta$$

To make predictions, we multiply the hidden layer output H by the output weights beta. Each row in f(x) represents the predictions for a corresponding data point. where, The output predictions f(x) is a matrix of size J x K, where J is the number of data points and K is the number of output variables.

H is a matrix of size J x L, where L is the number of hidden neurons that contains the transformed input data after applying the random weights and biases of the hidden layer. Each row represents to data points and each column represents to hidden neuron.

The output weights beta is a matrix of size L x K that constitutes a link between hidden layer output to output predictions. Each row corresponds to hidden neuron, and each column represents an output variable.

Since, f(x) is our training data target matrix, therefore it can written as:

$$f(x)=T=\begin{bmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_J^T \end{bmatrix}, \beta=\begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_L^T \end{bmatrix}$$

and

$$H=\begin{bmatrix} g(w_1 * x_1 + b_1) \cdots g(w_L * x_1 + b_L) \\ \vdots \\ g(w_1 * x_J + b_1) \cdots g(w_L * x_J + b_L) \end{bmatrix}$$

Moore-Penrose generalized inverse

The Moore-Penrose generalized inverse, sometimes known as the Moore-Penrose pseudoinverse, is a linear algebra topic. It is a non-square and singular (non-invertible) matrix inverse generalization. The Moore-Penrose pseudoinverse finds an approximate solution to a system of linear equations even if the matrix is neither square or of full rank.

The Moore-Penrose pseudoinverse is commonly indicated as (A^+), and it is calculated for a matrix (A). The Moore-Penrose pseudoinverse (A+) has the following characteristics for a given matrix (A):

1. If (A) is a square, invertible matrix, then (A^+) is the same as the ordinary matrix inverse, i.e., ($A^+ = A^{-1}$).
2. If (A) is a tall matrix (more rows than columns), then (A^+) is used to solve over-determined systems of linear equations, and it minimizes the sum of squared errors.
3. If (A) is a wide matrix (more columns than rows), then (A^+) is used to solve under-determined systems of linear equations. It provides the solution with the smallest norm.

The Moore-Penrose pseudoinverse can be computed using various methods, and one common approach is through the singular value decomposition (SVD). The formula to compute the pseudoinverse (A^+) for a matrix (A) is as follows:

If the SVD of A is $A = U\Sigma V^T$, where U and V are orthogonal matrices, and $\Sigma$is a diagonal matrix with singular values on the diagonal, then the pseudoinverse $A^+$is given by:

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

Where,

- Σ^+*is the pseudoinverse of* Σ*obtained by taking the reciprocal of the non-zero singular values and taking the transpose of the resulting matrix.*

How ELM trained?

In this section we are going to cover how ELM is get trained based on input training data in step by step procedure which are listed below,

1. Input Training Data: The first step is to gather the training data which include input features and target variable to feed into ELM machine learning algorithm.
2. Random Initialization: Next the weights and bias are randomly initialized in hidden layer thus this process eliminate iterative weight adjustment.
3. Feature Mapping: The input data is transformed into a high-dimensional feature space using randomly assigned weights. This process is known as feature mapping and helps to capture complex relationships between the input features.
4. Hidden Layer Processing: Then the transformed input data is processed by the hidden layer, resulting in an output matrix. This output is a key component in ELM's unique single-step learning process.
5. Output Weight Calculation: In the fifth step ELM leverage the output matrix by applying Moore-Penrose generalized inverse to calculate the output weights. This mathematical technique ensures robustness, even in the presence of noise or missing data.

6. Model Evaluation: Once the output weights are determined by **Moore-Penrose generalized inverse**, the model's training output is generated. This output is evaluated by specific metrics based on the problem use cases.
7. Fine-Tuning and Hyperparameters Adjustment: Depending on the assessment results, fine-tuning or hyperparameter adjustments can be applied to improve model performance.
8. Deployment: Finally training process is complete, the ELM model is ready for deployment in real-world applications, where it can make predictions and decisions based on the learned patterns.

   ELM offers a unique approach to machine learning by combining random initialization of weights, feature mapping, and the use of the Moore-Penrose generalized inverse. This allows for efficient training and robustness in handling noisy or incomplete data.

Application of ELM

Extreme learning machine is used in wide range of application in machine learning and artificial intelligence which are listed below,

- It is used in Image recognition and classification task, where applied to identify object in photos and perform analysis on medical image reports.
- ELM is used in Speech recognition task, by converting human speech to text which applied in voice assistants, transcription services.
- It is used in various Natural language processing tasks, including text classification, sentiment analysis, language translation, and chatbot development.
- ELM is used in finance sector to predict stock prices, exchange rates, and other financial data which beneficial for trader and investors.

- It is used in social media analysis by implementing sentiment analysis, trend detection, and user behavior understanding, which is beneficial for marketing and brand management.
- It is used in recommendation system that suggest products, content, or services to users based on their preferences and behavior which is beneficial to e-commerce website and content platform.
- It is used for predictive maintenance, helping to prevent equipment failures by analyzing sensor data which is beneficial to manufacturing industries.

Advantage of ELM

Extreme learning machine has number of advantage over other machine learning algorithm which are listed below,

- ELM is a relatively simple algorithm, which makes it easier to explain how the model makes decisions.
- ELM can learn from the training data in one step, without repeating the learning process in multiple steps. This makes much fast to train compared to other neural network methods like backpropagation.
- ELM is known for good generalization performance, even when the training data is limited. This means that ELM models are less overfit in training data, and they can still do a good job when dealing with new, unseen test data.
- ELM can handle noisy and incomplete training data effectively by using Moore-Penrose generalized inverse. This method calculate output weights, which is particularly avoid these types of errors.
- ELM is easy to implement in practice, and there many open source libraries to help with training and utilizing ELM models.