ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

EXTREME LEARNING MACHINE

Rudranshu Sharma*, Ankur Singh Bist, Vikas Kumar
* U.P.T.U.
U.P.T.U.

ABSTRACT

Machine learning [1], a branch of artificial intelligence, that gives computers the ability to learn without being explicitly programmed, means it gives system the ability to learn from data. There are two types of learning techniques: supervised learning and unsupervised learning [2]. In this paper, we describe what Extreme Learning Machine is, their advantages and limitations followed by a study of genetic algorithm.

KEYWORDS: Support vector Machine & Evolutionary Extreme Learning Machine

INTRODUCTION

Extreme Learning Machine (ELM)

Initially, gradient-descendent based leaning algorithm such as, back propagation [10] was used to train feed-forward neural networks, but the learning speed of these networks was much slower than that was required by the applications. This was the major drawback of these algorithms and the major reason behind was that in these algorithms, all the parameters of network had to be tuned which increased the dependency between the parameters.

Extreme Learning Machine (ELM)[7] is based on least square solution and was originally proposed for single hidden layer feed-forward network. In ELM, rather than tuning of all the parameters here, the weights between input and hidden neurons and the bias for each hidden neuron are assigned randomly. Here, hidden nodes may be sigmoid additive nodes or Gaussian kernel nodes. It requires that the activation function (which provides mapping between input and hidden neurons) to be infinitely differentiable (such as sigmoidal, Radial Basis, exponential, Sin etc) because if activation function is infinitely differentiable then the learning parameter can be assigned randomly [7]. The classification rate of ELM is dependent on the Number of hidden neurons (NHN). For a particular hidden neuron value the accuracy of ELM is best, if NHN are increased or decreased further the accuracy of ELM is reduced due to overfitting problem. The complexity of ELM is directly proportional to NHN if NHN value are higher the complexity of the ELM will be higher. Hidden node parameters are randomly taken and output node parameter is calculated using Moore-Penrose inverse [8][9].This algorithm is thousand times faster than past learning methods such as, back propagation and also reaches the smallest training error. Due to these features, generalization performance of this algorithm is also good.

We have given training data in the form of (x, t). Now we consider N=Number of training samples, n=Number of Input neurons=Number of features, L= Number of hidden neurons (NHN) and m=Number of output neurons and two variable q and p where q= 1…. L and p=1…..N. The pth sample is (xp,,tp) where xp=[xp1,xp2,…….,xpn]^t ϵ R^n and tp=[tp1,tp2,…….,tpm] ϵ R^m. $b_q$is the bias for the q^{th} hidden neuron, w_q is the input weight connecting input neurons and q^{th} hidden neurons where wq= [wqi, wqi,…, wqn]^T, β_q the Output weight for layers connecting all hidden neurons and r^{th} output neuron where

$$\beta_r = [\beta_{1r}, \beta_{2r}, ..., \beta_{Lr}]^T \quad r=1,2,.....m.$$

H = hidden layer output matrix and $g(w_i . x + b_i)$ is the Activation function
Activation function g(x) with L hidden nodes is defined as

$$\sum_{q=1}^{L} \beta_q . g_q(x_q) = \sum_{q=1}^{L} \beta_q . g(w_q . x_p + b_q) = o_p \qquad (1)$$

for p=1,2,3,…….N

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

With Activation function g(x) and L hidden nodes SLF's can approximate these N samples with zero error

$$\text{i.e. } \sum_{p=1}^{L} ||o_p - t_p|| = 0 \qquad (2)$$

Thus, we can say that there exist w_q, b_q and β_q such that,

$$\sum_{q=1}^{L} \beta_q \cdot g(w_q \cdot x_p + b_q) = t_q \quad \text{for } p=1,2,3\ldots\ldots\ldots N$$

With the help of (1) and (2) we can also show the above equation as

$$H_{train}\beta = T_{train} \qquad (3)$$

$$\text{Where } H_{train} = \begin{bmatrix} g(w_1 x_1 + b_1) & \cdots & g(w_L x_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(w_1 x_N + b_1) & \cdots & g(w_L x_N + b_L) \end{bmatrix}_{NxL}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{Lxm} \quad \text{and } T_{train} = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{Nxm}$$

When the input weights (w_q) and hidden layer biases (b_q) are determined then output weight β_q is calculated as:

$$H^T_{train}\ H_{train}\ \beta = H^T_{train}\ T_{train}$$
$$\beta = (H^T_{train}\ / H^T_{train}\ H_{train}\ )\ T_{train}$$
$$\beta = \text{pinv}(H_{train}) T_{train}$$

or

$$\beta = H^{+}T$$

Where H^+ is the Moore-Penrose generalized inverse of matrix H.
According to the perspective of evaluation the samples are divided in the training and the testing sets. The training sets are used to first get the value of the output weight (β). The output weight calculated is used to classify the test patterns using the following equation

$$H_{test}\ \beta = Y_{test}$$

L_{test}(n x 1), output label of n testing instances is determined using this equation.

$$L_{test} = arg^{row}_{max}(Y_{test})$$

The arg function returns the index of the maximum value for each row of Y_{test}.

FEATURE MAPPING IN ELM

According to ELM theory any nonlinear piecewise continuous activation function satisfying some condition can be used for feature mapping. ELM can approximate any function and solve any classification problem. ELM hidden node is not limited to additive or radial basis function type rather it can be any non linear piecewise continuous hidden node. So ELM extends from SLFN to generalized SLFN which can implement any feature mapping function following above condition. Some examples of feature mapping function are: Sigmoid function, Gaussian functions, Hinging function, Ridge Polynomials etc.

Extreme learning machine is generalized single layer feed forward network. Here generalized means ELM can use large variety of feature mapping function (hidden output function). Almost all nonlinear piecewise continuous functions can be used as the hidden-node output functions and hence various types of hidden node output function can be used in ELM. In general hidden node output can be written as:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

$$h(x) = [G(a_1, b_1, x) \quad \cdots\cdots \quad G(a_L, b_L, x)]$$

where $G(a,b,x)$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems and $\{a_i, b_i\}_{i=1}^{L}$ are *randomly* generated according to any continuous probability distribution. Some example of feature mapping function satisfying above condition is given below.

1) Sigmoid function

$$G(a,b,x) = \frac{1}{1+\exp(-(a.x+b))}$$

2) Hard-limit function

$$G(a,b,x) = \begin{cases} 1 & if \quad ax-b>0 \\ 0 & otherwise \end{cases}$$

3) Gaussian function

$$G(a,b,x) = -b\|x-a\|^2$$

4) Multiquadric function

$$G(a,b,x) = (\|x-a\|^2 + b^2)^{1/2}$$

Limitation of Extreme Learning Machine

Although extreme learning machine is a faster algorithm in comparison with traditional learning algorithm, but disadvantage of ELM is that it takes into consideration only balanced data; means data where distribution of instances between classes is equal. But now a days, due to imbalanced datasets ELM is unable to make proper classification.

Regularized Extreme Learning Machine

Regularized extreme learning machine uses ridge regression theory, a positive value is added to the diagonal of $H^T H$ or HH^T. The positive value is called regularization parameter. This parameter is trade off between maximizing marginal distance and minimizing least square error. Solution obtained using this method is more stable and robust, also provide good generalization performance. After applying regularization parameter least square solution can be analytically derived using Moore-Penrose generalized inverse. Here C is added for better generalization performance. Optimization method can also be used to get above solution for β. Similar to SVM we can formulate optimization function for ELM in which our aim is to minimize cumulative error $(\sum_{i=1}^{N}\|\xi_i\|, \quad \xi_i = H\beta - T)$

where $\xi_i = [\xi_{i1} \quad \xi_{i2} \quad \cdots \quad \xi_{im}]^T$ error vector corresponding to i^{th} sample and to maximize marginal distance.

Maximizing marginal distance is equivalent to minimizing norm output weight β. Optimization function can be written as:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

$$Minimize: \quad f = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}C\sum_{i=1}^{N}\|\xi_i\|^2$$

$$Subject \quad to: \quad h(x_i)\beta = t_i^T - \xi_i^T, \quad i = 1 \cdots N$$

Here C is regularization parameter which acts as trade-off constant between maximizing marginal distance and minimizing cumulative error. Solution of above problem using KKT condition is:

$$\beta = H^T\left(\frac{I}{C} + HH^T\right)^{-1}T, \quad where \quad N < L$$

$$\beta = \left(\frac{I}{C} + H^TH\right)^{-1}H^TT, \quad where \quad N > L$$

Weighted Extreme Learning Machine (WELM)

Weighted extreme learning machine is proposed to deal with imbalanced data. Initially, popular classifiers assume that data is balanced, which degrades performance because equal weightage is given to both the classes. It provides proper classification for majority class but minority class samples become misclassified.

WELM determines what degree of re-balance is required and according to the determined re-balance, assigns different misclassification cost for each instance, but for simplicity these algorithms choose a weighing scheme which is automatically generated from the class information. In Weighted Extreme Learning Machine, a weight matrix is generated automatically according to the class distributions, which are inversely proportional to the number of instances in the training data. So weight assigned to majority class is always lesser than weight assigned to minority class. Weighted Extreme Learning Machine can be termed as extreme learning machine with weight matrix and regularization parameter. Regularization parameter is used to represent trade-off between maximizing marginal distance and minimizing misclassification error cost/value. WELM is also faster learning algorithm because it takes all the advantages of extreme learning machine.

Training samples are given in the form of (x_i, t_i) where, $x_i \in R^n$ and t_i is either -1 or +1 for i=1…….N and n= number of features in each training sample. Define a NxN diagonal weight matrix for each training sample x_i where, weight assigned to minority class is always greater than weight assigned to majority class. w_{pj} = Input weight (for p=1….L,j=1….n) between n input nodes and L hidden layer nodes and b_p =bias (for p=1…L) at each hidden layer node are randomly generated between 0 and 1. $h_p = w_{pj}x_p + b_p$ =Hidden layer node output. Output layer node output is $y_i = h_1(x_i)\beta_1 + h_2(x_i)\beta_2 + \ldots + h_L(x_i)\beta_L = \mathbf{h}(x_i)\boldsymbol{\beta}$ and output in vector form is $\mathbf{Y = H\beta}$. Target output is $\mathbf{T} = t_1, t_2, \ldots\ldots\ldots\ldots, t_N$. Since it is a binary classifier so there is only one output node. Error vector is $\boldsymbol{\xi} = \mathbf{Y-T = H\beta-T}$. Output weight should be such that the marginal distance is maximum for better generalization performance. So, for maximizing marginal distance and minimizing error vector we can model optimization function as

$$\text{Minimize:} \quad f = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}CW\sum_{i=1}^{N}\|\xi_i\|^2 \qquad (4)$$

$$\text{Subject to: } h(x_i)\beta = t_i^T - \xi_i^T \text{, for i=1,2,……N} \qquad (5)$$

According to Karush-Kuhn-Tucker theorem[14] the optimization problem is equivalent to eq.(5) is

$$f = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}CW\sum_{i=1}^{N}\|\xi_i\|^2 - \sum_{i=1}^{N}\alpha_i(h(\chi_i)\beta - t_i - \xi_i) \qquad (6)$$

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Where α_i is lagrange multiplier. For obtaining optimality condition for KKT Theorem, partial derivation is done with respect to variables β, ξ and α

$$\frac{\partial f}{\partial \beta} = 0 \Rightarrow \beta = \sum_{i=1}^{N} \alpha_i h(x_i)^T = H^T \alpha \qquad (7)$$

$$\frac{\partial f}{\partial \beta} = 0 \Rightarrow \alpha_i = CW\xi_i \quad , \text{i=1……..N} \qquad (8)$$

$$\frac{\partial f}{\partial \alpha} = 0 \Rightarrow h(x_i)\beta - t_i - \xi_i = 0 \qquad (9)$$

By equation (8) and (9) we get

$$\alpha = CW[T - H\beta] \qquad (10)$$

By eq.(10) we put value of α into eq.(7)

$$\beta = H^T CW[T - H\beta]$$
$$\beta = H^T CWT - H^T CWH\beta$$
$$\beta + H^T CWH\beta = H^T CWT$$
$$\beta[1 + H^T CWH] = H^T CWT$$
$$\beta = \frac{H^T CWT}{1 + H^T CWH}$$

After divide numerator and denominator by C, we get

$$\beta = H^T . \frac{1}{\left[\frac{I}{C} + H^T WH\right]} . WT$$

$$\beta = H^T \left[\frac{I}{C} + H^T WH\right]^{-1} WT \quad \text{When N is large} \qquad (11)$$

When N is small then value of β can be obtained like this

By eq.(8) we get

$$\alpha = CW\xi$$

Put value of ξ from eq. (9)

$$\alpha = CW[T - H\beta] \qquad (12)$$

put value of β in eq.(12)

$$\alpha = CW[T - HH^T \alpha]$$
$$\alpha = CWT - CWHH^T \alpha]$$
$$\alpha + CWHH^T \alpha = CWT$$

Take C common from both sides

$$\frac{\alpha}{C} + WHH^T \alpha = WT$$

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

$$\alpha\left[\frac{I}{C}+WHH^T\right]=WT$$

$$\alpha=\frac{WT}{\left[\frac{I}{C}+WHH^T\right]}$$

Put value of α in eq.(7)

$$\beta=H^T\left[\frac{I}{C}+WHH^T\right]^{-1}WT$$

Weight determination in WELM

In weighted ELM, two schemes for weighing misclassification are given. Minority misclassification weight is given as inverse of total minority samples and majority misclassification weight is given as inverse of total majority samples.

$$W_{Minority}=\frac{1}{\$(Minority)} \quad and \quad W_{Majority}=\frac{1}{\$(Majority)}$$

Where, $(minority) is the number of samples belongs to minority class and $(majority) is the number of sample belongs to majority class.

In other scheme, minority misclassification weight is given as inverse of number of minority samples multiplied by 1 and majority misclassification weight is given as inverse of majority samples multiplied by 0.618.

$$W_{Minority}=\frac{1}{\$(Minority)} \quad and \quad W_{Majority}=\frac{0.618}{\$(Majority)}$$

Here, minority misclassification weight is always greater than majority misclassification weight which causes proper positioning of classification boundary and hence determines good generalization performance.

Limitation of Weighted Extreme Learning Machine

In Weighted ELM weights assigned to binary class, generate according to the number of instances, which creates dependency of WELM on number of instances.

EVOLUTIONARY ALGORITHM

In an optimization algorithm, a number of possible solutions to a problem are available and the task is to find the best solution possible in a fixed amount of time. For a search space with only a small number of possible solutions, all the solutions can be examined in a reasonable amount of time and the optimal one found. This exhaustive search, however, quickly becomes impractical as the search space grows in size. Traditional search algorithms randomly sample (e.g., random walk) or heuristically sample (e.g., gradient descent) the search space one solution at a time in the hope of finding the optimal solution. The key aspect distinguishing an evolutionary search algorithm from such traditional algorithms is that it is population-based. Through the adaptation of successive generations of a large number of individuals, an evolutionary algorithm performs an efficient directed search. The advantages include the simplicity of the approach, its robust response to changing circumstances, and its flexibility. Process of EA is shown in Fig: 2.1.

Among the evolutionary techniques, the genetic algorithms (GAs) are the most extended group of methods representing the application of evolutionary tools. They rely on the use of a selection, crossover and mutation operators. Replacement is usually by generations of new individuals.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Comparison with Particle Swarm Optimization: In comparison with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gbest (or lbest) gives out the information to others. It is a one -way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases. The main difference between the PSO approach compared to EC and GA is that PSO does not have genetic operators such as crossover and mutation. Particles update themselves with the internal velocity; they also have a memory that is important to the algorithm. Compared to GAs, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust.

Genetic Algorithm

In nature, an individual in population competes with each other for virtual resources like food, shelter and so on. Also in the same species, individuals compete to attract mates for reproduction. Due to this selection, poorly performing individuals have less chance to survive, and the most adapted or "fit" individuals produce a relatively large number of offspring's. It can also be noted that during reproduction, a recombination of the good characteristics of each ancestor can produce "best fit" offspring whose fitness is greater than that of a parent. After a few generations, species evolve spontaneously to become more and more adapted to their environment.

Genetic algorithm is also based on the same concept. It is a well known probabilistic global search and optimization method which is based on principle of evolution. The strength of GAs is in the parallel nature of their search. Through genetic operators, even weak solutions may continue to be part of the makeup of future candidate solutions. The genetic operators used are central to the success of the search. All GAs require some form of recombination, as this allows the creation of new solutions that have, by virtue of their parent's success, a higher probability of exhibiting a good performance. In practice, crossover is the principal genetic operator, whereas mutation is used much less frequently. Crossover attempts to preserve the beneficial aspects of candidate solutions and to eliminate undesirable components, while the random nature of mutation is probably more likely to degrade a strong candidate solution than to improve it.

Search Space

Most often one is looking for the best solution in a specific set of solutions. The space of all feasible solutions (the set of solutions among which the desired solution resides) is called search space (also state space). Each and every point in the search space represents one possible solution. Therefore each possible solution can be "marked" by its fitness value, depending on the problem definition. With Genetic Algorithm one looks for the best solution among a number of possible solutions represented by one point in the search space i.e.; GAs are used to search the search space for the best solution e.g., minimum. The difficulties in this ease are the local minima and the starting point of the search.

Here, basic genetic algorithm is shown which is based on "survival of fittest" concept, which means, fittest individuals dominating over the weaker ones. We can show the optimization problem like this:

$$\max_{x \in X} \quad f(x) \qquad (13)$$

Where X is search space and f is objective function, $f : X \to R$. Genetic algorithm does not work with problem (13) directly, but with coded version of it. Search space X is mapped into set of string S.

Function $X \to S$ is called coding function, which have to be specified depending on the needs of the actual problem.

Usually, S is finite set of binary strings:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

$$S = \{0,1\}^m$$

Where m is length of string. Generally simple binary code is used.

In the process of evolution, Genetic Algorithm takes a number of binary strings of finite length as initial population and each individual is associated a fitness value corresponding to the fitness of the solution it represents. This fitness value is assigned by fitness function. Fitness value is the evaluation of how good the candidate solution is.

Genetic Algorithm ranks individuals according to their fitness value, then selection is carried out which select individuals with best fitness value and delete the not so good specimens. These individuals with high fitness can be termed as promising candidates. These promising candidates are kept and allowed to reproduce by crossover. From them multiple copies are made, but the copies are not perfect; random changes are introduced by mutation during the copying process. These offspring's then go on to the next generation, forming a new pool of candidate solutions, and those candidate solutions which were worsened, or made no better, by the changes to their code are again deleted.

These steps are repeating for each generation. Let m is space dimension and l is length of each binary string, pc is crossover probability, pm is mutation probability and n is population size. Here, stopping criteria is number of generations which is defined by the user. In each generation t, n binary strings are present which will be denoted by

$$B_t = (b_{1,t}, b_{2,t}, \ldots\ldots, b_{n,t})$$

Table 2.1 Basic Genetic Algorithm

Basic structure of Genetic algorithm is shown here :

t=1;
Compute initial population B_1;
While stopping criteria is not fulfilled DO Begin
for i=1 to n DO
Select best individual $b_{i,t+1}$ from B_t according to fitness value
for i=1 to n DO
With probability p_c perform crossover of $b_{i,t+1}$ and $b_{i+1,t+1}$
for i=1 to n DO
With probability p_m eventually mutate $b_{i,t+1}$
end
end
end
t=t+1;
end

GENETIC PARAMETERS

- **Selection**

Selection mechanism determines which and how many parents to select, how many offspring to create, and which individuals will survive into the next generation. In Selection, comparison of each individual in the population takes place on the basis of a fitness function. Each individual has an associated value corresponding to the fitness of the solution it represents. The fitness is the evaluation of how good the candidate solution is. The optimal solution is the one which maximizes the fitness function. Types of selection are as follows:

Remainder Selection

The Stochastic Remainder Sampling has identical concepts used in the deterministic sampling and the population must

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

be formed with the integer part of the expression result a_{pi}/a_{pavg}. In this case, free places were filled based on the Roulette method. In Remainder Selection Mechanism, expected number of copies of a string is calculated as $m_i = f_i/f$. It assigns parents deterministically from the integer part of each individual's scaled value (m_i), and then uses roulette selection on the remaining fractional part. For example, if the scaled value of an individual is 2.3, that individual is listed twice as a parent because the integer part is 2. After parents have been assigned according to the integer parts of the scaled values, the rest of the parents are chosen stochastically. The probability that a parent is chosen in this step is proportional to the fractional part of its scaled value.

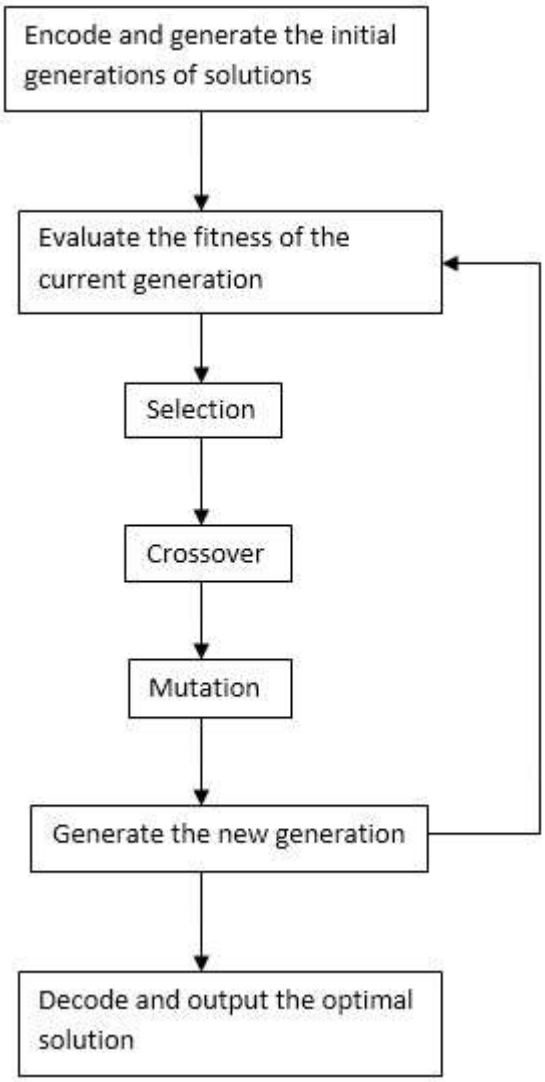

Figure2.2: Flowchart of Genetic Algorithm

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Roulette wheel Selection
In the Roulette wheel selection method [Holland, 1992]; the first step is to calculate the cumulative fitness of the whole population through the sum of the fitness of all individuals. After that, the probability of selection is calculated for each individual as being $psel_i = f_i/Pf_i$. Then, an array is built containing cumulative probabilities of the individuals. So, n random numbers are generated in the range 0 to Pf_i and for each random number an array element which can have higher value is searched for. Therefore, individuals are selected according to their probabilities of selection.

Stochastic Uniform Selection
Stochastic uniform lays out a line in which each parent corresponds to a section of the line of length proportional to its expectation. The algorithm moves along the line in steps of equal size, one step for each parent. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.

Tournament Selection
Tournament selects each parent by choosing individuals at random, the number of which you can specify by Tournament size, and then choosing the best individual out of that set to be a parent.

Uniform Selection
In Uniform selection parents are selected at random from a uniform distribution using the expectations and number of parents. This results in an undirected search.

Crossover
Crossover is a process of taking more than one parent solutions and producing a child solution from them with some crossover probability. It is applied to the mating pool with the hope that it creates a better offspring. The basic parameter in crossover technique is the crossover probability (P_c). Crossover probability is a parameter to describe how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. Types of crossover are:

Single Point Crossover
It chooses a random integer n between 1 and Number of variables, and selects the vector entries numbered less than or equal to n from the first parent, selects genes numbered greater than n from the second parent, and concatenates these entries to form the child.

Two Point Crossover
Two point selects two random integers *m* and *n* between 1 and Number of variables. The algorithm selects genes numbered less than or equal to m from the first parent, selects genes numbered from *m*+1 to *n* from the second parent, and selects genes numbered greater than *n* from the first parent. The algorithm then concatenates these genes to form a single gene.

Heuristic Crossover
Heuristic creates children that randomly lie on the line containing the two parents, a small distance away from the parent with the better fitness value, in the direction away from the parent with the worse fitness value.

Mutation
Mutation functions make small random changes in the individuals in the population, which provide genetic diversity and enable the genetic algorithm to search a broader space. It occurs according to user-defined mutation probability (p_m).

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

CONCLUSION

This paper gives a brief overview about Extreme Learning Machine. There are lots of advancements going on in this specific domain. Continuous evolution in this area has added various dimensions in base atoms of concerned area. This study will be helpful for those working in the area Extreme Learning Machine.

REFERENCES

1. Rumelhart D., Hinton G., "Williams R. Learning representations by back- propagation errors," Nature, pp. 533–536. 1986.
2. Huang G., Zhu Q., Siew C., "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. International Joint Conference on Neural Networks", pp. 985–990, 2004.
3. Huang G., Zhu Q., Siew C., "Extreme learning machine: Theory and applications. Neurocomputing 70," vol. 1 issue 3, pp. 489-501. 2006.
4. Huang G., Ding X., Zhou H., "Optimization method based extreme learning machine for classification. Neurocomputing" pp. 155-163, 2004.
5. Huang G., Chen L., Siew C.,"Universal approximation using incremental constructive feedforward networks with random hidden nodes," IEEE Transactions on Neural Networks, pp.879-892.2006.
6. Huang G., Zhou H., Ding X., Zhang R., "Extreme Learning Machine for Regression and Multiclass Classification", IEEE Transactions on Systems, Man, and Cybernetics, Part B, pp. 513-529.2012.
7. He H., Garcia E, "Learning from Imbalanced Data. IEEE Transactions on Knowledge And Data Engineering", pp. 1263-1284. 2009.
8. Deng W., Zheng Q., Chen L., "Regularized Extreme Learning Machine. IEEE Symposium on Computational Intelligence and Data Mining". pp. 389 - 395.2009.
9. Zong W., Huang G., Chen Y. , "Weighted extreme learning machine for imbalance learning. Neurocomputing", pp. 229-242. 2013.
10. Holland J., "Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan; re-issued by MIT Press". 1992.
11. Srinivas M., Patnaik L., "Genetic algorithms: A survey IEEE" pp. 17-26. 2009.
12. Rao C., Mitra S., "Generalized inverse of a matrix and its applications. In: Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics, Berkeley, Calif. Sixth Berkeley Symposium on Mathematical Statistics and Probability. University of California Press", pp 601-620.1972.
13. Serre D., "Matrices:Theory and Applications. Springer-Verlag, New York, Inc". 2002.
14. Fletcher R., "Practical Methods of Optimization: Constrained Optimization. Wiley, New York 2". 2002.
15. Reeves C., "Genetic Algorithms. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, vol 57. International Series in Operations Research & Management Science. Springer US," pp 55-82. doi:10.1007/0-306-48056-5_3. 2003.
16. Mitchell M., "An introduction to genetic algorithms. MIT Press".2003.
17. Sharapov R., "Genetic Algorithms: Basic Ideas, Variants and Analysis. In: Vision systems: segmentation and pattern recognition" pp 407-422.2007.
18. Sivaraj R., Ravichandran T., "A Review of selection methods in genetic algorithm. International Journal of Engineering Science & Technology" vol. 3 issue 5, pp. 3792-3797.2011.
19. Herrera F., Lozano M., "Heuristic crossovers for real-coded genetic algorithms based on fuzzy connectives. In: Voigt H-M, Ebeling W, Rechenberg I, Schwefel H-P (eds) Parallel Problem Solving from Nature — PPSN IV, vol 1141. Lecture Notes in Computer Science. Springer Berlin Heidelberg", pp 336-345. doi:10.1007/3-540-61723-X_998.1996.
20. Alcala-Fdez J., Fernandez A., Luengo J., Derrac J., Garcia S. , "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithm and Experimental Analysis Framework. Multiple-Valued Logic and Soft Computing 17" vol. 2 issue3, pp.255-287.2011.