Ensemble Pruning on Majority Voting ELM using PSO

Pranay Ranjan
pranaycode@gmail.com
Department of Computer Science, M.A.N.I.T.,
Bhopal, India

Abstract

Neural Network(NN) is one of the state of the art algorithms for classification problem. It is specifically used for classification problems because the number of neurons in the output layer can be finite and limited. To speed up computation in NN, many variants have been proposed. A kind of neural networks called extreme learning machines(ELM), is used to make the computation faster for real valued classification. But, it suffers from the problem of instability and over-fitting. Ensembling of ELM can increase the performance of a classifier. One of the ensembling techniques such as majority voting can be used here, but redundancy is introduced due to it. To solve this problem, pruning of ensemble can be used.

It is already stated in [1] that many is better than all. So, if we are given an ensemble, there exists a subset of ensemble that gives same or better result as compared to entire ensemble. The act of finding a subset for the aforementioned reason is called pruning. Certain optimisation algorithms can be used for pruning which actually give better solution compared to other pruning techniques. But there are overheads due to them [2] as these approaches are stochastic in nature.

Particle swarm optimisation(PSO) is one such approximate optimisation technique which can be for this purpose. PSO has already been exploited in past for ensemble pruning [3], but using it to prune an ensemble of ELM is a fresh and an interesting area of research. This algorithm is based on social interaction and is inspired from bird flocking, fish schooling and swarming theory. Thus

this paper analyses the result of pruning based on PSO and properly depicts the observations.

Keywords: Extreme Learning Machine, Ensemble Pruning, Optimisation-based pruning, Majority Voting, Genetic Algorithm, Particle Swarm Optimisation

1. Introduction

ELM is a modification of NN. ELM is a feed-forward neural network consisting of a single hidden layer. A NN uses a method such as back-propagation algorithm to compute the weights which are used to train it through feed-forward approach. ELM escapes this step and simply uses feed-forward algorithm. A lot of time is saved due to this. Actually, the weights between input and hidden layer are randomly initialized and the weights between hidden layer and output layer is compared analytically. Due to random initialization of weights, there is the problem of instability and over-fitting in ELM [4] [5].

One way to resolve this problem is to work with a set of ELM classifiers instead of using a single ELM and then combine the result using certain approaches called as ensembling techniques. Due to random initialization of input layer weights the performance of extreme learning machine fluctuates. The performance fluctuation due to any change in the parameters of the classification algorithm or training dataset composition is known as error due to variance. Ensembling approaches like majority voting, bagging, stacking can be used to reduce this variance in performance and also, increase the performance of the classifier. Majority voting is used to get combined output of the set of classifiers in an ensemble. The ensemble in that case is denoted by VELM. VELM improves the performance over ELM at the cost of redundancy. Pruning can be used to decrease redundancy in this case.

PSO was first introduced by James Kennedy and Russell Eberhart in 1995 [6]. PSO uses the concept of a swarm. A swarm is a disorganised or random collection(population) of moving individuals that tend to cluster together(meet

at global optimum) which each individual seems to be moving in a random direction. Individuals are called particles of the swarm. Each particle is assigned position and velocity randomly. Particles keep track of their best and also the best of all the particles in each step. So, each particle decides the direction of movement by taking the help of those particles who have already attained global optimum. Ultimately, all the particles reach the optimum by sharing information among each other. This approach can be employed in ELM ensemble by treating classifiers as particles. VELM_PSO denotes the ensemble which is pruned using PSO.

In the next section, the paper decsribes the related working of ELM, VELM and ensemble pruning techniques. After this, the proposed work is discussed,ie, voting based VELM_PSO. And then, the experimental setup and results are described. The last section includes conclusion and future work.

2. Related Work

This section discusses the fundamental topics that come across as a part of the work.

Extreme Learning Machine.

Voting based ELM. Out of several ensembling techniques, majority voting is one of them. In majority voting, that class which is predicted by most of the classifiers in the ensemble is the output class. The required class is obtained using mode operation which returns the class having maximum frequency. Also, votes of all classifiers of the ensemble have equal weightage. If the weights of the votes of the classifiers vary, then this variant is called weighed voting based ELM.

Lets say there is an ensemble having k classifiers and there are m instances in the data used for prediction. The predicted class of all instances and each classifier is stored in a matrix of size $(k * m)$ and lets represent it by term β. And, the combined output of the ensemble is denoted by α and has size $(1 * m)$.

$$\beta = [\ \beta^1;\ \beta^2;\ \beta^3;\ \ldots\ ;\ \beta^k\ ] \tag{1}$$

$$\beta^i = [\ \beta^i_1,\ \beta^i_2,\ \ldots\ ,\ \beta^i_m\ ]\ \ \forall\ i \in [1, k] \tag{2}$$

$$\alpha = mode(\beta) \tag{3}$$

Ensemble Pruning and PSO. Although performance is enhanced by using a set of classifiers instead of a single one, but more memory is used now to store the results. At the same time, additional time is consumed to generate new classifiers and obtain results. Nevertheless, there are some classifiers in an ensemble whose presence in the ensemble has no impact or negative impact on overall performance. These classifiers are called redundant classifiers or low-performing classifiers. Pruning the low-performing classifiers while maintaining a good diversity of the ensemble is typically considered as a proper recipe for a successful ensemble. This is the motivation for pruning.

Apart from optimisation based pruning, other categories of pruning are order based pruning and clustering based pruning [7]. After pruning, ensemble size decreases and computational overhead decreases. But, the process of pruning leads to some overhead in terms of time taken to execute the algorithm involved in it. In general, optimisation techniques give better results than other techniques, but they have more overheads involved [2]. Moreover, this overhead is different for different optimisation algorithms depending upon their complexity.

The advantage of PSO is that its convergence speed is very high [8] [9]. This makes it an efficient algorithm with less overheads. Besides this, PSO is versatile in nature. It is due of the fact that it can be applied to a host of optimisation problems. The reason for the same is that it does not use the gradient of the problem to be optimised. This is specifically useful when gradient is too cumbersome or even impossible to compute. Also, PSO is easy to understand and implement.

3. Proposed Work

The paper uses optimisation-based pruning to remove redundant classifiers in the ensemble of ELMs. Any optimisation is performed on the basis of some metric. In PSO, that metric is also called fitness function. Here, G-mean is the metric or fitness function. Data being unbalanced is a common observation. In that case, G-mean is a better measure as compared to overall accuracy. The pseudo-code of the proposed algorithm is as follows:

Algorithm for VELM_PSO.

3.1. Training Phase

Input : L = number of hidden neurons, N = number of classifiers in ensemble, M = number of training instances, m = number of testing instances, f = number of features in the dataset, C = number of class labels.

Size of training data is $(M * f)$ and that of testing data is $(m * f)$. ELM returns a matrix of size $(M * C)$ for training data and $(m * C)$ for testing data. The final output is of size $(M * 1)$ for training data and $(m * 1)$ for testing data.

3.2. Pruning Phase

Next step is to prune the ensemble and PSO algorithm [10] is used now.

Input : n = swarm size(population size), w = inertial weight, w_{max} = maximum inertial weight, w_{min} = minimum inertial weight, $c1$ and $c2$ are acceleration factors, $maxIter$ = maximum number of iterations, $maxStallIterations$ = maximum number of stall iterations, $FunctionTolerance$ = function tolerance.

A vector of size N is used which denotes the positions of the particles in the swarm and is represented by x. The lower bound(lb) and upper bound(ub) for each x_i ($i \in [0, N]$) are 0 and 1 respectively. w controls the balance between global search and local search [11]. Also, it is to be noted that $rand()$ is a random number generator. The following is the required algorithm :

- Initialise the positions of the particles using the formula

$$x_i = round(lb(x_i) + rand() * (ub(x_i) - lb(x_i))) \tag{4}$$

- Initialise the velocities of the particles which is denoted by v.

- Evaluate the fitness of particles at the current location. Find the position at which particle best occurs for each particle and it is denoted by matrix *pbest*. At the same time, find the position at which best among all the particles occurs and it is denoted by a vector *gbest*. Size of pbest is $(n*N)$ and that of gbest is $(1*N)$;

- for iteration $= 1 : maxIter$

  1. Update inertial weight by the formula

$$w = w_{max} - \frac{(iter + (w_{max} - w_{min}))}{maxIter} \tag{5}$$

  2. Update particle velocity using

$$v_{ij} = w*v_{ij}+c1*rand()*(pbest_{ij}-x_{ij})+c2*rand()*(gbest_j-x_{ij}) \tag{6}$$

  (i is for i^{th} particle and j is for j^{th} dimension of position or velocity).

  3. Update position by

$$x_{ij} = x_{ij} + v_{ij} \tag{7}$$

  4. Check boundary violations for lb and ub of x.
  5. Evaluate fitness function for the current location of the particles. If this fitness is greater than maximum fitness, then update the maximum fitness. Also update the positions *pbest* and *gbest* corresponding to this update.
  6. If $maxStallIterations$ is reached and change in maximum fitness is not greater than $FunctionTolerance$, then terminate and exit the loop.

- Now, *gbest* is obtained. But, it is a vector containing real values in [0,1]. So, it is converted to a bit vector by rounding it off. If $gbest_i = 1$, then include this classifier in the pruned ensemble and reject otherwise.

- In this way, PSO generates a pruned ensemble from the given ensemble by optimising the fitness function which is G-mean here. P denotes the size of pruned ensemble.

3.3. Testing Phase

4. Experimental Setup

Data Specification. The experimentation has been done on 17 binary and 3 multiclass datasets, downloaded from the Keel-data set Repository and the UCI repository. The data sets are available in 5 fold cross validation format i.e. for each dataset has 5 training and 5 testing sets. The details of the datasets, ie, the number of training instances, testing instances, attributes and classes are specified in Table 1.

Performance Metric and Evaluation. The predictions of any dataset can be used to construct a confusion matrix. It is a square matrix having size equal to the number of output classes. G-mean and overall accuracy are calculated from confusion matrix using these formulae for multiclass classification:

$$R_i = \frac{Number\ of\ correctly\ classified\ instances\ of\ i^{th}\ class}{Number\ of\ instances\ belonging\ to\ i^{th}\ class} \tag{8}$$

$$G - mean = (\ \prod_{i=1}^{C} R_i\ )^{\frac{1}{C}} \tag{9}$$

Here, R_i is the recall of i^{th} class and C is the number of classes of a dataset.

$$OverallAccuracy = \frac{Number\ of\ correctly\ classified\ instances}{Total\ number\ of\ instances} \tag{10}$$

Now, in case of of binary classification, the results can be categorized as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). True positive and True Negative are the correctly classified instances belonging to positive class and negative class respectively. False Negative is the number of instances belonging to positive class that are misclassified as negative class. FP is the number of instances belonging to negative class and classified as positive class. The overall accuracy and G-mean for binary classification is calculated by the following formulae :

$$OverallAccuracy = \frac{TP + TN}{Number\ of\ Samples} \tag{11}$$

Table 1: **Specifications of datasets**

DATASET	**Classes**	**Attributes**	**Training Instances**	**Testing Instances**
APPENDICITIS	2	7	84	22
BANANA	2	2	4240	1060
BUPA	2	6	276	69
CHESS	2	36	2556	640
HABERMAN	2	3	244	62
HAYES-ROTH	3	4	128	32
HEART	2	13	216	54
IONOSPHERE	2	33	280	71
IRIS0	2	4	120	30
MONK2	2	6	345	87
NEWTHYROID	3	5	172	43
PHONEME	2	5	4323	1081
PIMA	2	8	614	154
RING	2	20	5920	1480
SA_HEART	2	9	369	93
SONAR	2	60	166	42
SPAMBASE	2	57	3677	920
SPECTFHEART	2	44	213	54
TITANIC	2	3	1760	441
VEHICLE	4	18	676	170

$$G - mean = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}} \tag{12}$$

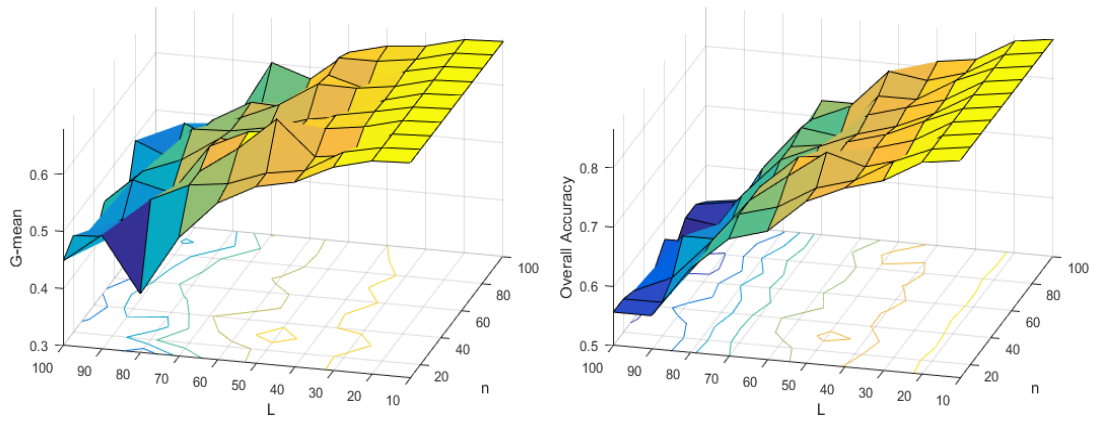
Parameter Settings. VELM is a special case of VELM_PSO when all the classifiers are used to find the output class. Here, the number of classifiers in VELM is taken to be 51, ie , N = 51. For both VELM and VELM_PSO, the results are calculated by varying the number of hidden neurons, denoted by L , on [10 ,20 , ... ,100]. Also, these results are averaged over 10 rounds. Out of results at all used values of L, the best one is the final result.

The tuning of the parameters used in PSO is an important aspect to obtain proper results. The characteristics that need to be considered are : number of iterations($maxIter$), population size(n), inertial weight(w) and acceleration factors($c1$ and $c2$). Previous works have suggested the values that can be used for these parameters [12] [13] [14] [11] [10]. $c1$ and $c1$ generally range in [2, 2.05]. Suggested range of w is [0.4, 0.9]. n is mainly used in [10, 100] and $maxIter$ value of 100 or above is sufficient for convergence.

But, these range of values do not always work and they actually depend on the type of problem under consideration. To analyse the effect of n and $maxIter$, different curves are plotted. The effect of n [10 ,20 , ... ,100] on metrics for a particular $maxIter$ by varying L is shown in figure 1 & figure 2 and by keeping L constant is shown in figure 3. The variation of overall accuracy with $maxIter$ by keeping n and L constant is shown in figure 4.

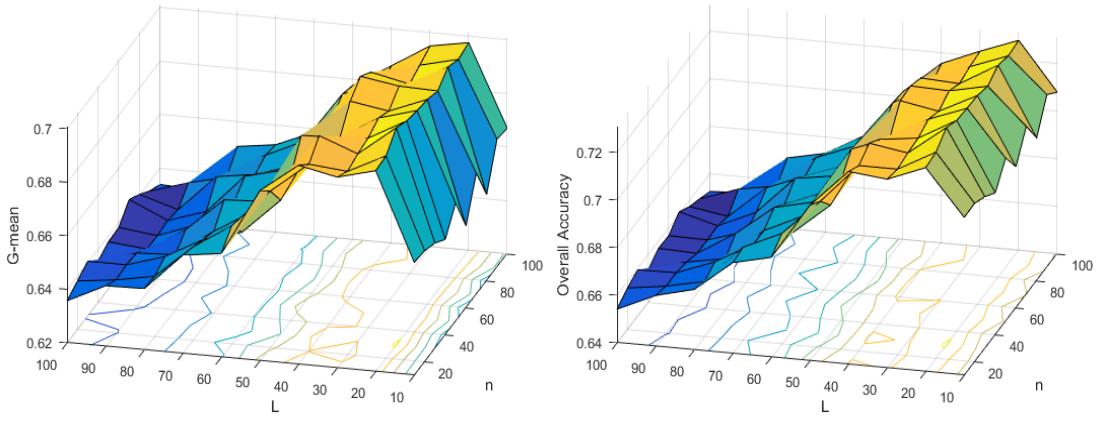
Experimental Results. Testing G-mean and overall accuracy of VELM_PSO and VELM for all datasets are shown in table 2 and table 3 respectively. As it can be observed from table 2 that VELM_PSO surmounts VELM in all 20 datasets. And in table 3, VELM_PSO surmounts VELM in 17 out of 20 datasets. The variation of G-mean and overall accuracy with respect to L for MONK2 dataset is shown in figure 5.

Statistical Test. To compare the results of VELM_PSO with that of VELM, wilcoxon signed rank test is conducted. This test returns a value called p-value.



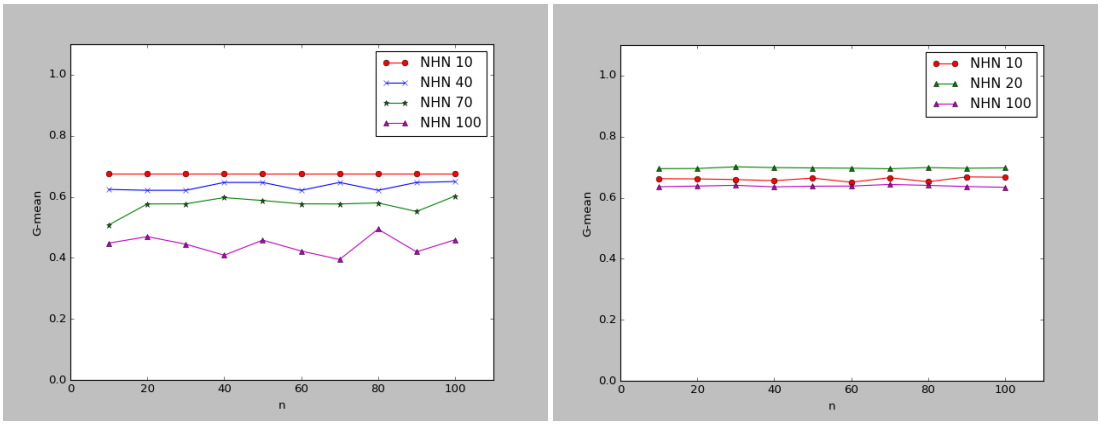
(a) G-mean (b) Overall Accuracy

Figure 1: Display of test G-mean and overall accuracy of APPENDICITIS dataset at $maxIter$ = 50 by varying L and n



(a) G-mean (b) Overall Accuracy

Figure 2: Display of test G-mean and overall accuracy of BUPA dataset at $maxIter = 50$ by varying L and n



(a) APPENDICITIS (b) BUPA

Figure 3: Display of test G-mean at $maxIter = 50$ by varying n and keeping L constant

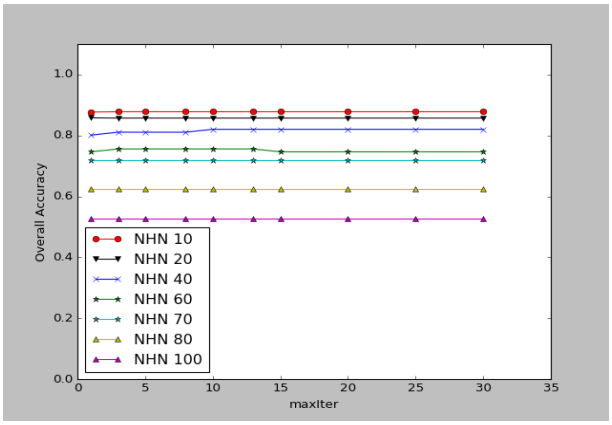


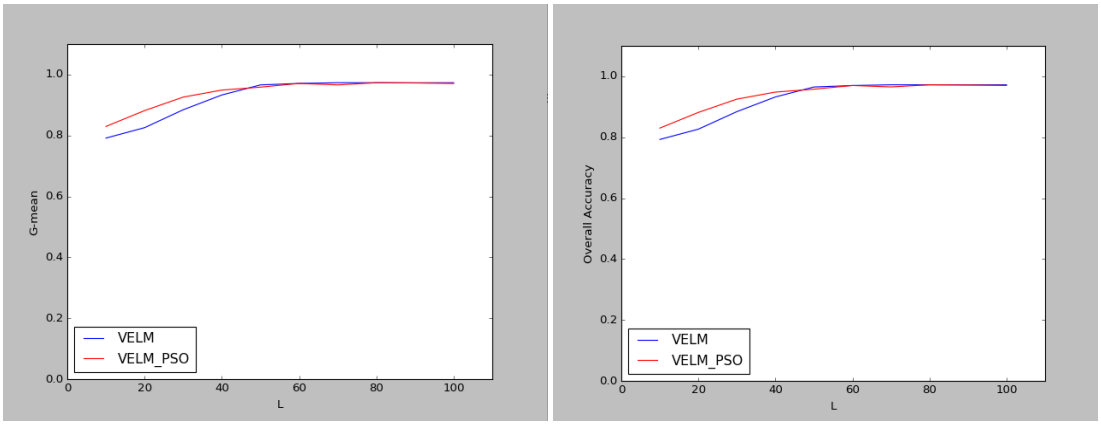
Figure 4: Display of test overall accuarcy of APPENDICITIS dataset at $n = 50$ by changing $maxIter$ and keeping L constant

Table 2: **Testing G-mean for PSO**

DATASET	**VELM**		**VELM_PSO**		
	L	G-mean%	P	L	G-mean%
APPENDICITIS	10	**67.70**	25	10	**67.70**
BANANA	100	89.90	23	90	**89.91**
BUPA	20	69.96	21	20	**69.99**
CHESS	100	95.36	17	100	**95.65**
HABERMAN	20	50.01	21	20	**50.24**
HAYES-ROTH	20	72.12	24	50	**74.48**
HEART	30	83.18	21	20	**83.50**
IONOSPHERE	70	90.35	24	90	**90.46**
IRIS0	10	**100**	31	10	**100**
MONK2	70	**97.32**	22	80	**97.32**
NEWTHYROID	20	86.74	31	60	**87.24**
PHONEME	100	80.44	19	100	**80.68**
PIMA	40	70.57	19	40	**70.70**
RING	100	94.31	21	100	**94.59**
SAHEART	30	64.62	20	20	**64.76**
SONAR	80	84.73	27	80	**86.56**
SPAMBASE	100	89.05	23	100	**89.34**
SPECTFHEART	100	39.83	19	100	**42.94**
TITANIC	10	**67.08**	31	10	**67.08**
VEHICLE	100	81.62	22	100	**82.05**

Table 3: **Testing Average Overall Accuracy (A) for PSO**

DATASET	**VELM**		**VELM_PSO**		
	L	A%	P	L	A%
APPENDICITIS	10	**86.75**	25	10	**86.75**
BANANA	100	90.26	23	90	**90.29**
BUPA	20	**73.10**	21	20	72.99
CHESS	100	95.41	17	100	**95.67**
HABERMAN	10	73.53	21	20	**74.51**
HAYES-ROTH	20	73.75	22	20	**75.00**
HEART	30	84.19	21	20	**84.30**
IONOSPHERE	70	**92.95**	24	90	92.89
IRIS0	10	**100**	31	10	**100**
MONK2	70	**97.22**	22	80	**97.22**
NEWTHYROID	40	94.42	25	30	**94.47**
PHONEME	100	84.30	19	100	**84.43**
PIMA	20	**77.47**	19	10	77.34
RING	100	94.52	21	100	**94.77**
SAHEART	20	73.43	20	20	**73.56**
SONAR	80	85.04	27	80	**86.97**
SPAMBASE	100	90.59	23	100	**90.82**
SPECTFHEART	10	**79.40**	31	10	**79.40**
TITANIC	20	**78.87**	31	20	**78.87**
VEHICLE	100	83.15	22	100	**83.52**



(a) G-mean (b) Overall Accuracy

Figure 5: Display of test G-mean and overall accuracy at different values of L [10,20,...,100] for MONK2 dataset

The threshold for p-value is set to 0.05. If p-value obtained is less than 0.05, then there is significant difference between VELM_PSO and VELM. The p-value for the two algorithms here is 4.3681e-04. The lesser the p-value, the more is significance of difference between two set of results.

5. Conclusion and Future Work

In this paper, ELM classifiers are ensembled using majority voting method and this ensemble is called VELM. And, a new classifier termed as VELM_PSO has been proposed. VELM_PSO prunes VELM to obtain a subset of ensemble which performs approximately same as entire ensemble. This result is that pruned ensemble gives approximately the same or more accurate results as compared to entire ensemble over testing data-set. Overall accuracy and g-mean of testing data in pruned ensemble are approximately same as that of ensemble without pruning. Also, PSO algorithm consumes time and adds to the computational overhead. But, this overhead is not large as compared to other optimisation techniques such as Genetic Algorithm(GA).

The future work is to analyse other optimisation techniques in pruning

VELM. The aim would be to compare testing G-mean, overall accuracy and time of computation of those algorithms with PSO.

References

[1] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: Many could be better than all, Artificial Intelligence 137 (12) (2002) 239 – 263. doi:http://dx.doi.org/10.1016/S0004-3702(02)00190-X.

[2] G. Martnez-Muoz, D. Hernndez-Lobato, A. Surez, An analysis of ensemble pruning techniques based on ordered aggregation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 245–259. doi:10.1109/TPAMI.2008.78.

[3] J. Zhang, K.-W. Chau, Multilayer ensemble pruning via novel multi-sub-swarm particle swarm optimization, j-jucs 15 (4) (2009) 840–858. doi:10.3217/jucs-015-04-0840.

[4] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (13) (2006) 489 – 501. doi:http://dx.doi.org/10.1016/j.neucom.2005.12.126.

[5] J. Cao, S. Kwong, R. Wang, X. Li, K. Li, X. Kong, Class-specific soft voting based multiple extreme learning machines ensemble, Neurocomputing 149, Part A (2015) 275 – 284. doi:http://dx.doi.org/10.1016/j.neucom.2014.02.072.

[6] J. Kennedy, R. Eberhart, Particle swarm optimization, Neural Networks, 1995. Proceedings., IEEE International Conference on 4 (1995) 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

[7] Z.-H. Zhou, Ensemble Methods Foundations and Algorithms, Chapman and Hall, 2012.

[8] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73. doi:10.1109/4235.985692.

[9] C.-H. Chen, Y.-p. Chen, Convergence time analysis of particle swarm optimization based on particle interaction, Adv. in Artif. Intell. 2011 (2011) 7:7–7:7. doi:10.1155/2011/204750.

[10] M. N. Alam, Particle swarm optimization: Algorithm and its codes in matlab, ResearchGate (2016) 1–10doi:10.13140/RG.2.1.4985.3206.

[11] X. Hu, Y. Shi, R. Eberhart, Recent advances in particle swarm, IEEE 1 (2004) 90–97 Vol.1. doi:10.1109/CEC.2004.1330842.

[12] Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, Proceedings of the 7th International Conference on Evolutionary Programming VII (1998) 591–600.

[13] M. E. H. Pedersen, Good parameters for particle swarm optimization, Technical Report HL1001, Hvass Laboratories.

[14] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (6) (2003) 317 – 325. doi:http://dx.doi.org/10.1016/S0020-0190(02)00447-7.