Neural Networks and Particle Swarm Optimization for Function Approximation

Output Data Range Considerations for Efficient Learning

Rachana R. Patel
Heriot-Watt University
Edinburgh ,
EH14 4AS, UK
rrp3@hw.ac.uk

Robert Stone
Heriot-Watt University
Edinburgh ,
EH14 4AS, UK
rws1@hw.ac.uk

Abstract—**The optimization algorithms are an active area of research to solve complex real-world optimization problems. It is essential to identify parameters that govern the optimization to enable decoding the complex optimization problems. Here, using function approximation as the problem domain, Artificial Neural Networks (ANN) for learning and conventional Particle Swarm Optimization (PSO) as the optimization algorithm, we have identified parameters in data, ANN architecture and PSO which influence the function approximation. Among PSO parameters, the ratio of cognitive and social components with respect to the number of informants and relative velocity update play a crucial role in minimizing error and reaching global optima during function approximation. ANN architecture, especially the depth defined by the number of hidden layers, along with the choice of activation functions, govern the detection of global optima and swarm convergence to the detected global optima. Interestingly, we find that output data-range transformation to match the activation function of choice, especially the sigmoid activation function, dramatically improves learning and minimizes error for most of the functions approximated in this study.**

Keywords—ANN, PSO, sigmoid activation function, output data range, function approximation, relative velocity update, ANN architecture complexity, ANN depth, cognitive and social components, informants.

I. Introduction

Learning and optimization are two critical steps towards solving complex real-world problems. Bioinspired computation enables optimized learning to solve high dimensional problems by offering a range of computational technologies that model natural systems [1]. Biology inspired learning, and optimization algorithms are active areas of research in the field of artificial intelligence with significant applications spanning from computer vision to operational management [2].

The biological neural network in human brain, comprises of approximately 100 billion neurons which governs human learning from complex thought processes to pattern recognition based on efficient calculations. Artificial neural networks (ANNs) are learning algorithms modeled based on biological neural networks. ANNs acquire knowledge by extracting useful patterns from data and model input-output relationships by applying a nonlinear function to weighted inputs [3]. The architecture of ANN consists of many interconnected computing units, called neurons. In a simple ANN, the connections between neurons span between the input, hidden, and output layers. The number of input neurons are proportional to the number of input features in the data. Similarly, the number of output neurons is dependent on the type of anticipated output [4]. The complexity of ANN dictated by the number of hidden layers and neurons within each hidden layer may vary based on the learning task and can be considered as hyperparameters. The nonlinear functions used by ANN to model input-output relationships may also vary based on the problem domain. Each neuron has little intrinsic convergence towards the optimum solution. However, a combination of multiple neurons performing the same task has improved ability to model nonlinear relationships between inputs and outputs in parallel, which dramatically enhances the overall learning [5].

Learning algorithms such as ANN can learn by sequentially updating the initial input-output relationship model to minimize the output prediction error [5]. There are several methods to ensure the required information update. Methods such as backpropagation are popular for ANN learning [6]. In recent years, due to their effectiveness, nature-inspired optimization algorithms such as Particle Swarm Optimization (PSO) have gained popularity in optimizing ANN-based learning and problems solving [7]. PSO is a stochastic optimization algorithm modeled based on the collective behavior of a flock of the birds moving towards the destination [8]. PSO is based on how each entity of a large group collaborate with other entities based on same but simple rules to create diverse systems to achieve the ultimate goal collectively. For example, birds in large groups have a better survival advantage due to the higher likelihood of detecting food sources and predators. The key concept of PSO is based on following such natural processes to solve optimization problems. The basic version of canonical PSO consists of a collection of entities or particles performing the same task. Each particle has its velocity and position within the swarm. After each iteration, each particle updates their position based on the velocity update to converge towards the global optimum solution. Every particle updates its personal best and the swarm's best based on predefined criteria. Although stochastic, the PSO based optimization is significantly governed by all of its parameters such as swarm size, number of informants and variables within the algorithm.

Like any modelling algorithms, the hyperparameters of ANN-based learning and PSO based optimization algorithms have to be fine-tuned to reach an ideal combination to solve the assigned task [9]. Here, we have investigated hyperparameters that govern the PSO based optimization of ANN learning for six different function approximations. We find that ANN complexity, choice of nonlinear ANN functions, PSO algorithm variables and data output range influence the efficiency of ANN-PSO to approximate all the functions tested.

II. Program development rationale

The ANN-PSO program for function approximation problem domain consists of ANN, PSO and data as three major components which were developed based on following rationale:

A. ANN

I. ANN Architecture: ANN algorithm was developed in an iterative, incremental manner. For ANN architecture design, a simple multi-layered ANN was first created with one hidden layer and configurable number of neurons per layer. The number of nodes or neurons in each layer were configurable. After the basic functionality testing for correct layer design and subsequent check for calculations, the number of layers was also made configurable. The ANN design was configured to suit each of the six functions tested. The basic ANN architecture of the input and output layers with one node each along with flexible hidden layers and nodes was similar for four functions (linear, cubic, sine and tanh) with single feature data input (figure 1A). ANN architecture for complex and xor functions with two input data features have two nodes in the input layer (figure 1B). For function approximation experiments, the configurable ANN nodes and layers enabled investigations on ANN complexity, i.e., ANN width dictated by the number of nodes in each hidden layer and depth dictated by the number of layers as one of the hyperparameters. Since, one hidden layer is considered to be sufficient for simple function approximations, a comparative study of one versus more hidden layers was also conducted.

II. ANN feed forward and activation functions: The main purpose of ANN is to model input-output relationships using nonlinear functions and weighted inputs. For this, the feed-forward function was created where the weighted input values are passed through ANN layers. The output values from each layer are transformed using an activation function to ensure they fall within an anticipated range as dictated by the function. The error of prediction from the final output layer of ANN is calculated in the form of the mean squared error (MSE) of predicted and actual output. The feed-forward function returns the MSE after each forward pass of input values. The goal of the program is to converge at the lowest possible MSE. Predominantly, linear and nonlinear activation functions such as sigmoid and tanh are used and investigated as one of the hyperparameters of the program. A linear function was used as a baseline which will limit the overall learning and optimization, whereas the sigmoid and tanh functions were chosen as they are the most widely used activation functions [10]. Certain function approximation experiments were also carried out using cosine activation function to gauge the comparative PSO performance. In most experiments, same activation function was chosen for all ANN layers. However, for certain functions that failed to converge efficiently, a combination of two activation functions for different layers was also investigated.

B. PSO

I. PSO structure and algorithm: PSO structure was defined as two objects: Particle class (single ANN) and Swarm class (collection of ANNs). The purpose of the Particle class is to (i) assign initial random position and velocity based on a configurable range and (ii) to initialize the personal best position. The purpose of the Swarm class is to create a swarm object with a configurable number of particles. The Swarm class holds the PSO optimization function which holds a configurable number of informants and fitness update information based on the particle's best and global best (based in informants' best) position. Throughout all experiments, informants were randomly selected from a swarm for the information on the global best position. The key algorithm that updates the particle position has two steps: 1. the particle's velocity update based on velocity weight (alpha), cognitive and social components and 2. the particle's current position update based on the relative updated velocity dictated by the relativity factor epsilon:

step 1: $v_{i} = \alpha * v_i + c0_{(\text{cognitive component})} * (p.best_{(\text{particles best position})} - pi) + c1_{(\text{social component})} * (p.best_{(\text{global best position})} - pi)$

step 2: $p_{i(\text{updated position})} = p_{i(\text{current position})} + \varepsilon_{(\text{relativity factor})} * v_{i(\text{updated velocity from step 1})}$

All the PSO parameters and variables were investigated and fine-tuned for each of the function approximations.

C. Data

For function approximations with single input features (linear, cubic, sine and tanh), the input data (column 0) was read into the program. For expected output, the input values were transformed based on the function approximated. For complex and xor approximation with two input features, both input (column 0-1) and output data (column 2) were read into the program. For exploratory investigations, the expected output data-range was transformed to match the activation function used. For example for cubic approximation using sigmoid activation function the following expected outputs were defined: (i) Output data (non-transformed): *Y_output= cubic(X_input)* and (ii) Output data-range transformed: *Y_output= sigmoid(cubic(X_input)).* Comparative studies were carried out using the transformed and non-transformed output data.

D. PSO-ANN Integration

For the program execution, the swarm optimization function was used where ANN's forward pass function was utilized to define and update the position and velocity of the particles in the swarm. Thus, the fitness of particles is determined based on MSE generated during ANN's forward pass.

III. Methods

The program is coded in python using Jupyter notebooks based on the program development rationale. Publicly available code structures were used as references [11]–[13]. Briefly, forward pass function in the program defines the ANN's learning algorithm for feed-forward information transfer between layers. Forward pass function returns MSE calculated based on the specified activation function and expected output. Optimize function in Swarm class defines the PSO optimization algorithm where each particle's best and swarm's global best position (based on randomly selected informants from the specified range) is updated based on the fitness which is determined by the MSE calculated after each forward pass. Based on the personal and global best and other PSO parameters, the particle velocity update is calculated which is used to change the particle position based on the defined relativity factor. The weight and learning rate range determines the initial particle position and velocity randomly. Each function approximation is carried out in a separate notebook. The experiments were conducted as follows: (1) The value ranges for various PSO parameters were determined through multiple trials for each function. For detailed experiments, function-specific PSO parameters remained unchanged. (2) Detailed comparative studies on the activation functions and output data-range transformation were performed for all the function approximations using single or multiple hidden layers. These investigations are presented as individual notebooks. Performance of sigmoid versus tanh activation function was investigated and linear activation function was used as a base line. (3) Functions showing poor approximation were further investigated for activation function combinations. MSE and convergence of the program to a detected global minimum were the key experimental readouts. MSE below one was considered acceptable, and a stable MSE over iterations was considered as convergence (figure 1C). Maximum of 100 iterations was used to determine PSO convergence. Average MSE and convergence iterations from ten independent experiments were considered for the comparative studies.

IV. Results

A. PSO parameters

Using the ANN with a single hidden layer, values for various PSO parameters were explored for each function. The PSO parameters were fine tuned to minimize premature convergence. Based on the convergence of MSE to the global minimum, swarm size of 10, informant number of 6 and relativity factor (epsilon) of 0.2 was found to be optimum. Approximately 3:1 ratio of social to cognitive components and the velocity weight value close to 0.9 assisted in convergence. A narrow value range for learning rate (particle velocity) and relatively wider range for weights (particle position) also enabled better convergence.

B. ANN complexity, Activation function and Output data range transformation

A three way comparative study was carried out to investigated the effects of ANN complexity, activation function and output data-range transformation on function approximations. As reported before, ANN width determined by number of nodes per hidden layer led to premature convergence or inability to detect global minima for most of the functions approximated.

I. Linear Function

For ANN with a single layer, only sigmoid activation function enabled convergence after 40 iterations with an MSE of ~0.38 using the non-transformed output data (figure 2A). For ANN with multiple layers, only the program using transformed output data was convergent with sigmoid function. Highlighting the stochasticity of PSO optimization, TanH function showed rare convergence. For linear approximation, sigmoid function with transformed data reached the lowest observed MSE of 0.018 within 60 iterations (figure 3).

II. Cubic Function

In ANN with single hidden layer, only a combination of sigmoid function with transformed output data ensured convergence after 80 iterations with a low MSE of 0.008 (figure 2B). Dense ANN improved the convergence with tanh function (figure 4). Overall, for cubic approximation, only tanh function with multilayered ANN lead to convergence with low MSE of ~0.1. A combination of tanh with output data-range transformation enabled earlier convergence compared to non-transformed data.

III. Sine Function

The program very rarely stably converged with tanh function. In both sparse and dense ANNs, the sigmoid function performed better at convergence and MSE scores. Interestingly, for single hidden layered ANN with sigmoid function, non-transformed data yielded convergence after just ~20 iterations (figure 2C). For a denser ANN with sigmoid function, both data types converged after ~40-60 iterations. However, the use of transformed output data yielded the lowest MSE of ~0.015 (Figure 5).

IV. TanH Function

Only sigmoid function enabled convergence irrespective of the ANN complexity or output data transformation (figures 2D, 6). Interestingly, tanh activation function performed poorly for tanh function approximation. As observed in other function approximations, use of output data-range transformation was capable of reaching the lowest MSE of ~0.015. Overall, a single hidden layered ANN with the sigmoid function was sufficient for tanh function approximation independent of output data-range transformation.

V. Complex Function

In ANN with single hidden layer, only sigmoid function enabled convergence (figure 7A). Here, while the output data transformation reached a very low MSE of ~0.006, use of non-transformed data caused convergence as early as ~20 iterations with lowest MSE of ~0.17. For ANN with increased complexity, a combination of sigmoid with transformed data led to convergence at ~40-60 iterations with low MSE of ~0.006 (figure 8). Interestingly, tanh function also enabled early convergence at ~20-40 iterations with MSE of ~0.127 in denser ANNs, but only with non-transformed data.

VI. XOR Function

Irrespective of ANN complexity and data transformation, only tanh function was able to achieve stable convergence after ~40 iterations (figures 7C-D, 9,10). Although the sigmoid function could reach low MSE of ~0.013, it rarely achieved convergence. Overall, irrespective of ANN complexity, the tanh function converged with a lower MSE of ~0.14 with data transformation compared to MSE of ~0.24 with the use of non-transformed data.

C. Combination of Activation functions

Overall, we observed that in most function approximations, sigmoid function achieved the lowest MSE only with transformed output data-range. We also observed that for certain functions, dense ANN with tanh could enable convergence at earlier iterations. Hence, we hypothesized that a combination of these two functions in a multilayered ANN would be able to achieve low MSE with stable convergence even with non-transformed data. We tested this hypothesis using cubic approximation as the problem domain. As shown in figure 11, dense ANNs with single function did not show a stable convergence within 100 iterations. However, by using the sigmoid function for hidden layers and tanh for output layer, convergence was achieved after ~50-60 iterations with MSE below 0.15 (figure 11D). Interestingly, the reverse combination of using tanh for hidden and sigmoid for output neither achieved low MSE or detectable convergence.

V. Discussion and Conclusion

PSO mediated optimization is influenced by several parameters within its optimization algorithm as well as ANN architecture and associated hyperparameters [14]. As previously reported, we find that PSO parameters significantly affect PSO convergence. The primary purpose of identifying the correct values or ranges for PSO parameters was to minimize any premature convergence during function approximation. The gold standard to gauge the program's performance includes a stable convergence to the lowest detectable MSE within few (10 -100) iterations. Practically, such an ideal combination is rare, and in such cases, as a trade-off, PSO convergence was considered more important when the lowest detected MSE was acceptable (below 1). Our studies corroborate with previous reports on the number of informants, the ratio of social to cognitive components and velocity weight values[15]. Most of these were common across all function approximations tested. Thus, suggesting that these values may govern the general optimization algorithms across various problem domains. Overall, wider spread for particle position and limited particle velocity yielded optimum convergence highlighting the importance of appropriately initiating the particle spread and movement.

As previously reported, simple ANN with single hidden layer used in combination with appropriate activation functions and output data range was sufficient for most of the function approximations. However, use of deeper ANN enabled a stable convergence at earlier iterations for most function approximations, except Sine approximation which was best with simple ANN using a sigmoid function. These observations may suggest that ANN complexity may not always confer better solutions. Consistent with observations made by others, we confirm that wider ANNs lead to poor performances [16]. Thus, in most instances, narrow but deeper ANNs may serve as better learning architectures.

The choice of activation functions in ANN play a crucial role in modelling input-output relationships which govern the learning. Sigmoid and tanh are two widely used activation functions to model nonlinear input-output relationships. In our experimental setup, sigmoid offered a better or comparable performance to tanh for most of the approximations, expect for xor where only the tanh was capable of stable convergence. These observations may suggest the importance of difference in ranges between sigmoid (0 to1) and tanh (-1 to 1) values. Multiple activation functions, when used within an ANN, can enhance learning performance [17]. Confirming this, our studies suggest that a combination of sigmoid and tanh provided a superior performance of stable convergence with low MSE, especially for functions such as cubic that do not approximate easily. We also investigated the cosine activation function, and its performance was comparable to sigmoid and tanh.

Activation functions change the range of input data to model the input-output relationships. Since the programs learning and optimization are judges based on MSE defined by the difference between predict and actual output, we hypothesized that transforming the data output range to match that of the activation function may enable better approximations. Our experimental analyses, confirm that indeed output data-range transformation significantly improved the program's performance by enabling stable convergence at the lowest detectable MSE in the entire study. This was particularly true for sigmoid function. Thus, for certain problems, such output data-range transformations may offer a superior solution.

Thus, based on our investigations presented here, we find that: (i) The PSO mediated optimization is highly stochastic. Fine-tuning of PSO parameters is essential to avoid premature convergence. (ii) ANN architecture and complexity influence learning. Simple ANNs with one hidden layer can perform well with most general approximation task. For complex tasks, a deeper ANN with more layers is superior in performance to wider ANN with more nodes per layer. (iii) Activation functions are crucial hyperparameters, and a combination of activation functions enables better performance in deep ANNs. (iv) Both learning and approximation are improved by transforming the output data-range to match the activation function used in ANN.

APPENDIX

Appendix includes figures that have been referenced in the report. Github link: https://github.com/rws1/BIC_RP-RS

REFERENCES

[1] A. Darwish, "Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications," *Futur. Comput. Informatics J.*, vol. 3, no. 2, pp. 231–246, 2018.

[2] Y. Wang *et al.*, "Cancer genotypes prediction and associations analysis from imaging phenotypes: a survey on radiogenomics," *Biomark. Med.*, vol. 14, pp. 1151–1164, Aug. 2020.

[3] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6. American Psychological Association, US, pp. 386–408, 1958.

[4] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 2017.

[5] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, no. 1, pp. 17–61, 1988.

[6] F.-. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Control Syst. Mag.*, vol. 10, no. 3, pp. 44–48, 1990.

[7] M. A. Lones, "Mitigating Metaphors: A Comprehensible Guide to Recent Nature-Inspired Algorithms," *SN Comput. Sci.*, vol. 1, no. 1, pp. 1–12, 2020.

[8] M.-P. Song and G.-C. Gu, "Research on particle swarm optimization: a review," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, 2004, vol. 4, pp. 2236–2241 vol.4.

[9] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part I: background and development," *Nat. Comput.*, vol. 6, no. 4, pp. 467–484, 2007.

[10] H. Zhang, T. W. Weng, P. Y. Chen, C. J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS, pp. 4939–4948, 2018.

[11] Z. Ahmad, "No Title," *Train Neural Network (Numpy)— Particle Swarm Optimization(PSO). Available at:* <https://medium.com/@zeeshanahmad10809/train-neural-network-numpy-particle-swarm-optimization-pso-93f289fc8a8e>[Accessed 22 November 2020] .

[12] D. Erdeljan, "No Title," *2020. Dusanerdeljan/Pso. Available at: <https//github.com/dusanerdeljan/pso>* [Accessed 22 November 2020].

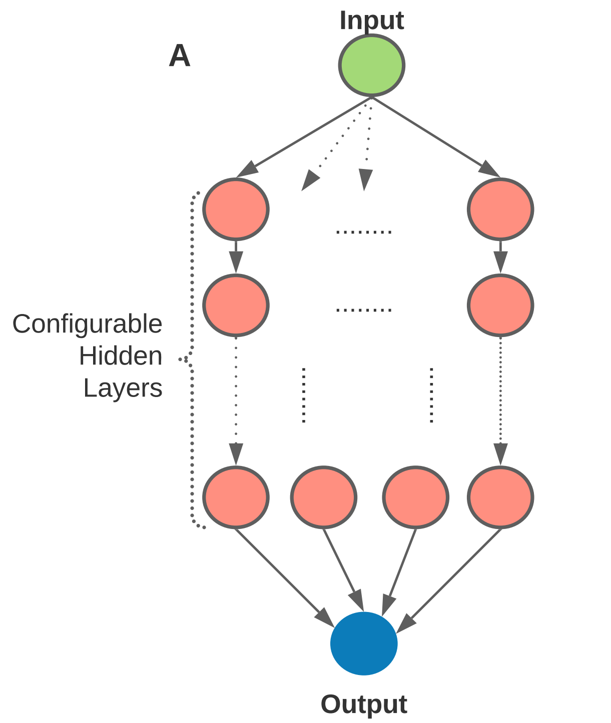
[13] J. Ideami, "No Title," *Coding A 2 Layer Neural Netw. From Scratch Python. Available at:* <https//towardsdatascience.com/coding-a-2-layer-neural-network-from-scratch-in-python-4dd022d19fd2> [Accessed 22 November 2020].

[14] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation, CEC 2000*, 2000, vol. 1, pp. 84–88.

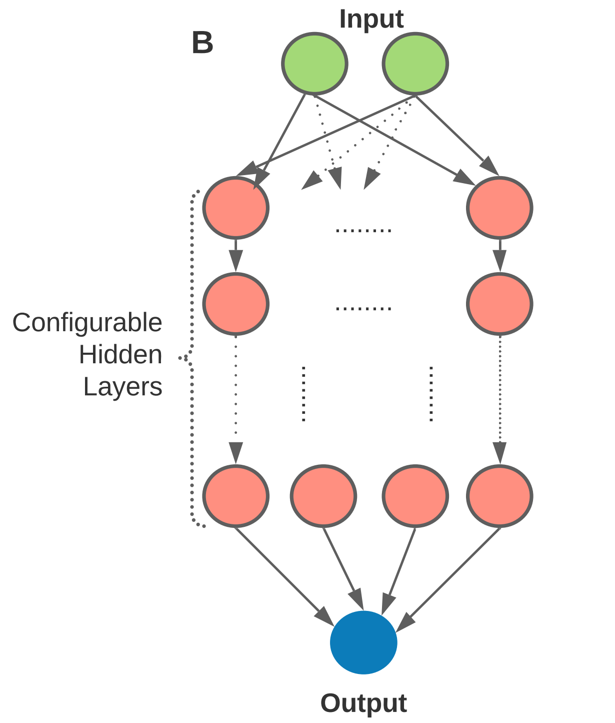
[15] J. Garcia-Nieto and E. Alba, "Why Six Informants is Optimal in PSO," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, 2012, pp. 25–32.

[16] J. Lee *et al.*, "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent," in *Advances in Neural Information Processing Systems*, 2019, vol. 32, pp. 8572–8583.

[17] F. Manessi and A. Rozza, "Learning Combinations of Activation Functions," *Proc. - Int. Conf. Pattern Recognit.*, vol. 2018-August, pp. 61–66, 2018.



ANN architecture used for linear, cubic, sine and tanh function approximations



ANN architecture used for complex and XOR function approximations

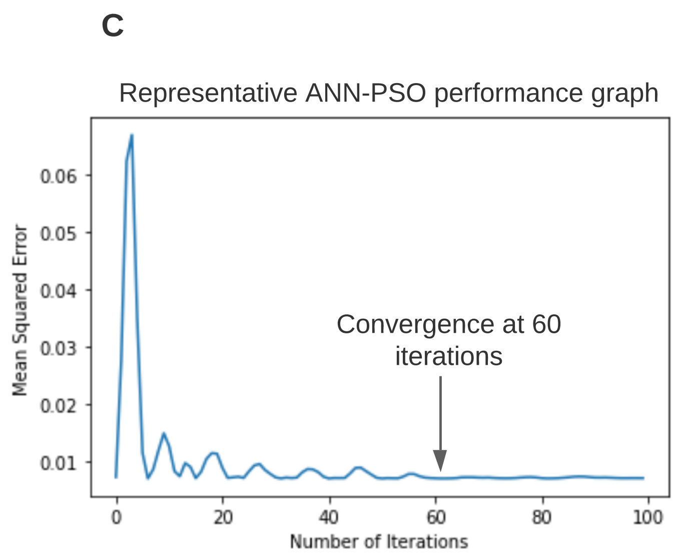


Figure 1

A

Linear function approximation for single hidden layer:

PSO parameters	**Sigmoid activation function**		**Hyperbolic Tangent activation function**		**Linear activation function**
	Output Data Range Transformed	**No Transformation**	**Output Data Range Transformed**	**No Transformation**	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.15
Alpha range	0.9	0.9	0.89 to 0.9	0.89 to 0.9	0.89
Position range	-0.1 to 0.1	-0.1 to 0.1	0.04 to 0.08	0.04 to 0.08	-0.01 to 0.01
Velocity range	0.05 to 0.25	0.05 to 0.25	0.6 to 0.9	0.6 to 0.9	0.05 to 0.15
Cognitive component	1	1	1.1	1.1	1
Social component	3	3	3.9	3.9	3
Hidden Layer nodes	15	15	5	5	6
Number of Hidden Layers	1	1	1	1	1
Convergence Iterations	~50-80 when convergent	~40-60	Non-convergent	Non-convergent	Non-convergent
Least MSE (approx)	~0.018	~0.38	~0.19	~0.3	~0.32
Global Best (lowest MSE)	~0.018	~0.38	~0.18	~0.27	~0.32

B

Cubic function approximation with single hidden layer:

PSO parameters	**Sigmoid activation function**		**Hyperbolic Tangent activation function**		**Linear activation function**
	Output Data Range Transformed	**No Transformation**	**Output Data Range Transformed**	**No Transformation**	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.9	0.9	0.9
Position range	-0.1 to 0.1	-0.1 to 0.1	-0.1 to 0.1	-0.1 to 0.1	-0.01 to 0.09
Velocity range	0.15 to 0.25	0.15 to 0.25	0.15 to 0.25	0.15 to 0.25	0.05 to 0.25
Cognitive component	1	1	1	1	1
Social component	3	3	2.7	2.7	3
Hidden Layer nodes	5	5	10	10	15
Number of Hidden Layers	1	1	1	1	1
Convergence Iteration	~80-100	Rare converge	Non-convergent	Non-convergent	Non-convergent
Least MSE (approx)	~0.009	~0.39	~0.08	~0.09	~0.14
Global Best (lowest MSE)	~0.008	~0.38	~0.07	~0.08	~0.12

C

Sine function approximation with single hidden layer:

PSO parameters	**Sigmoid activation function**		**Hyperbolic Tangent activation function**		**Linear activation function**
	Output Data Range Transformed	**No Transformation**	**Output Data Transformed**	**No Transformation**	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.9	0.9	0.9
Position range	-0.1 to 0.09	-0.1 to 0.09	-0.1 to 0.09	-0.1 to 0.09	-0.1 to 0.09
Velocity range	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25
Cognitive component	1	1	1.4	1.4	1.7
Social component	3	3	2.9	2.9	3.2
Hidden Layer nodes	20	20	6	6	20
Number of Hidden Layers	1	1	1	1	1
Convergence Iterations	~80	~20	Non-convergent	Non-convergent	Non-convergent
Least MSE (approx)	~0.015	~0.26	~0.2	~0.16-0.24	~0.2-0.24
Global Best (lowest MSE)	~0.015	~0.26	~0.2	~0.16-0.24	~0.2-0.24

D

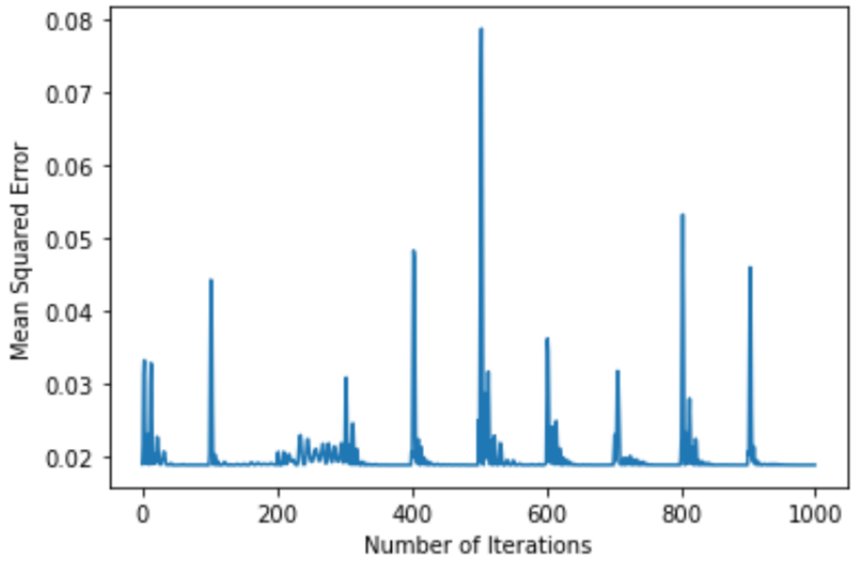
TanH function approximation with single hidden layer:

PSO parameters	**Sigmoid activation function**	**Sigmoid activation function (Without Output Data Range Transformation)**	**Hyperbolic Tangent activation function**	**Linear activation function**
Swarm size	10	10	10	10
Informants	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.9 to 0.91	0.9
Position range	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.01
Velocity range	0.05 to 0.25	0.05 to 0.25	0.15 to 0.2	0.15 to 0.25
Cognitive component	1.2	1.2	1.2	1.5
Social component	3.7	3.7	3.9	4
Hidden Layer nodes	15	15	15	15
Number of Hidden Layers	1	1	1	1
Convergence Iterations	~50	~40	Non-convergent	Non-convergent
Least MSE (approx)	~0.013	~0.26	~0.21	~0.2
Global Best (lowest MSE)	~0.013	~0.26	~0.21	~0.19

Figure 2

Linear Function Approximation with Sigmoid Activation

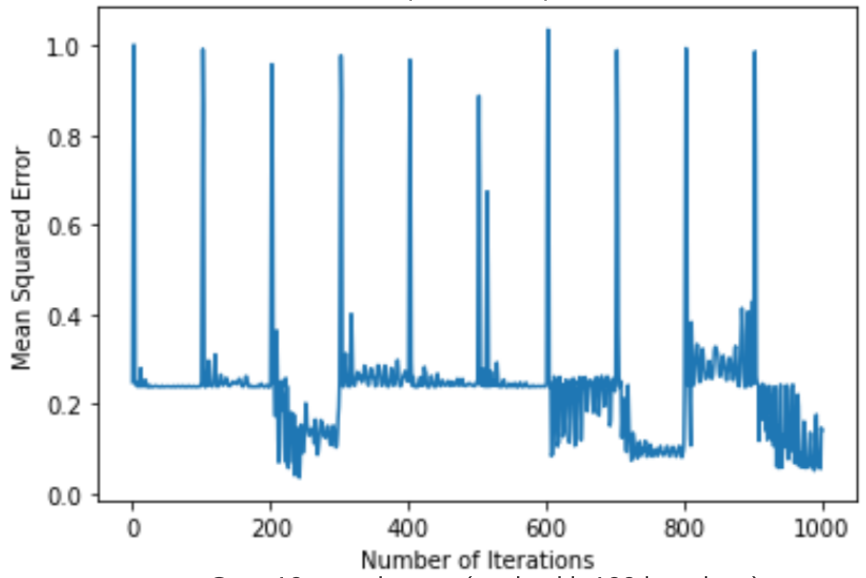
10 Independent Experiments



Over 10 experiments (each with 100 iterations)

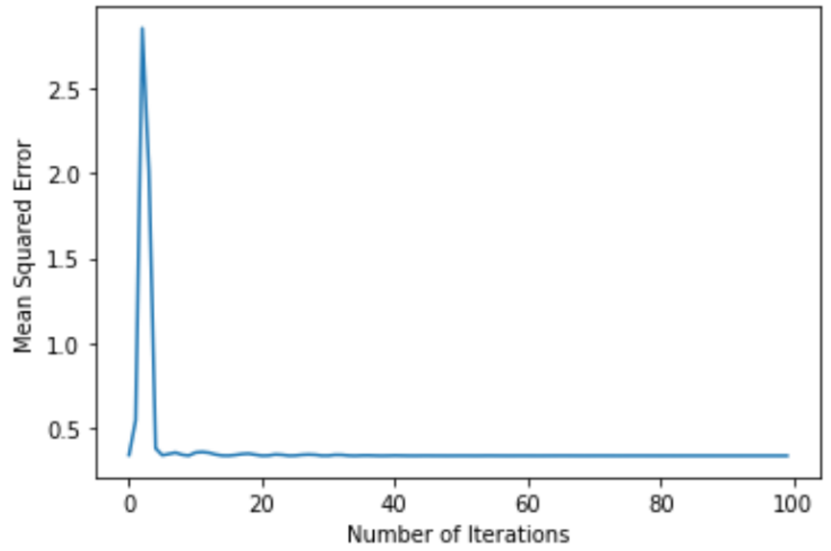
Linear Function Approximation with TanH Activation

10 Independent Experiments



Over 10 experiments (each with 100 iterations)

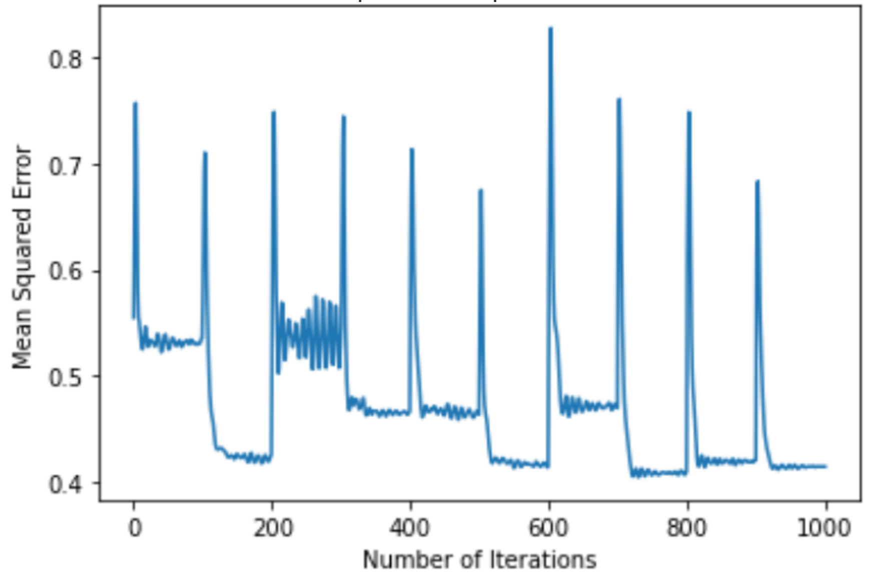
Linear Function Approximation with Linear Activation



One representative experiment

Linear Function Approximation with Sigmoid Activation without Output data range transformation

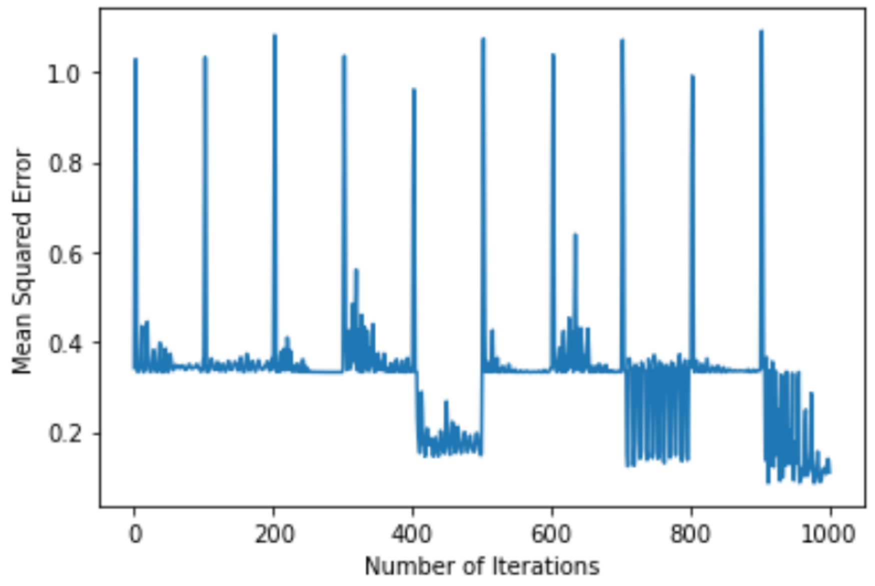
10 Independent Experiments



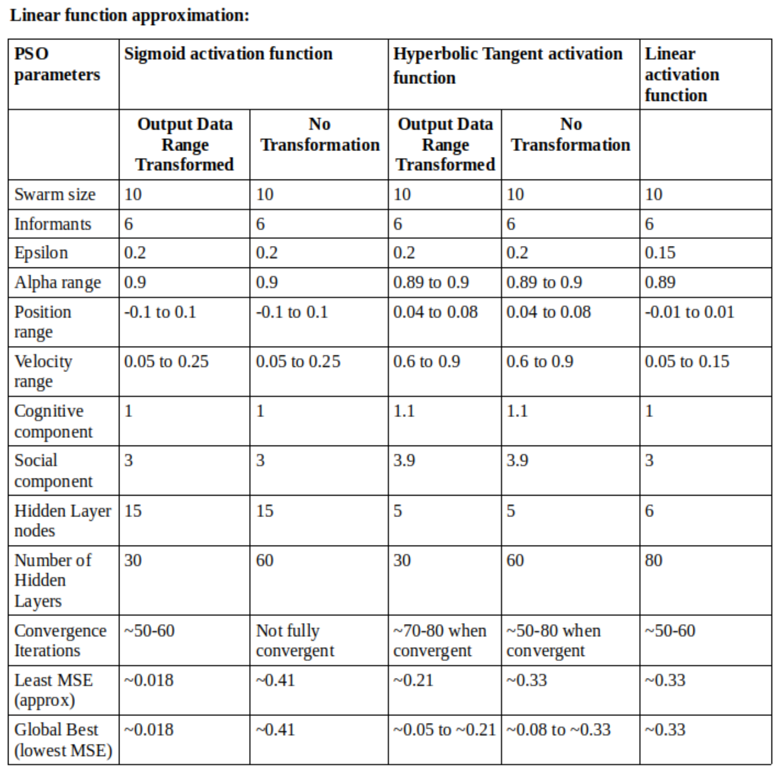
Over 10 experiments (each with 100 iterations)

Linear Function Approximation with TanH Activation without Output data range transformation

10 Independent Experiments



Over 10 experiments (each with 100 iterations)

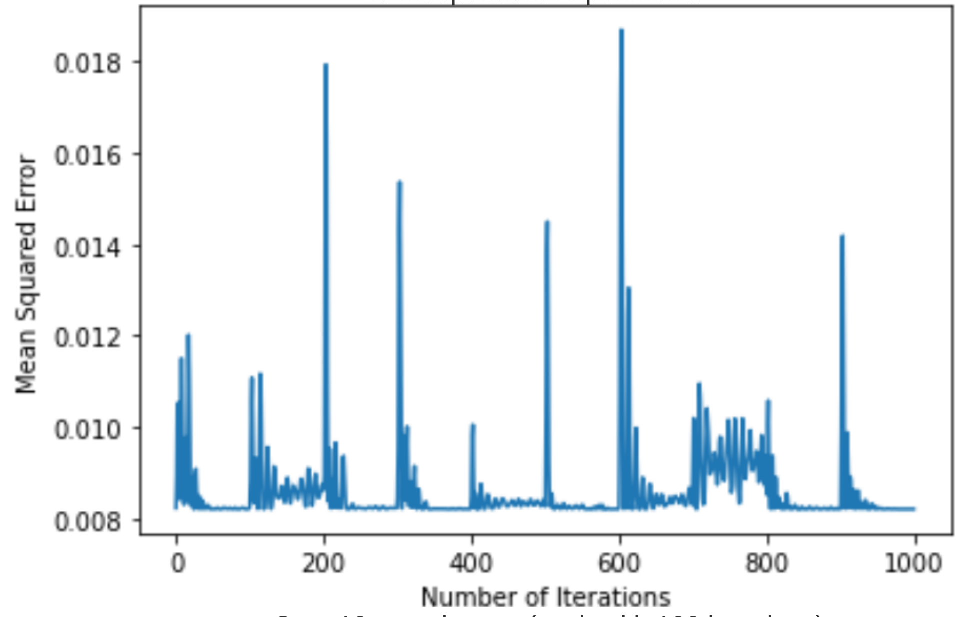


Linear function approximation:

PSO parameters	Sigmoid activation function		Hyperbolic Tangent activation function		Linear activation function
	Output Data Range Transformed	No Transformation	Output Data Range Transformed	No Transformation	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.15
Alpha range	0.9	0.9	0.89 to 0.9	0.89 to 0.9	0.89
Position range	-0.1 to 0.1	-0.1 to 0.1	0.04 to 0.08	0.04 to 0.08	-0.01 to 0.01
Velocity range	0.05 to 0.25	0.05 to 0.25	0.6 to 0.9	0.6 to 0.9	0.05 to 0.15
Cognitive component	1	1	1.1	1.1	1
Social component	3	3	3.9	3.9	3
Hidden Layer nodes	15	15	5	5	6
Number of Hidden Layers	30	60	30	60	80
Convergence Iterations	~50-60	Not fully convergent	~70-80 when convergent	~50-80 when convergent	~50-60
Least MSE (approx)	~0.018	~0.41	~0.21	~0.33	~0.33
Global Best (lowest MSE)	~0.018	~0.41	~0.05 to ~0.21	~0.08 to ~0.33	~0.33

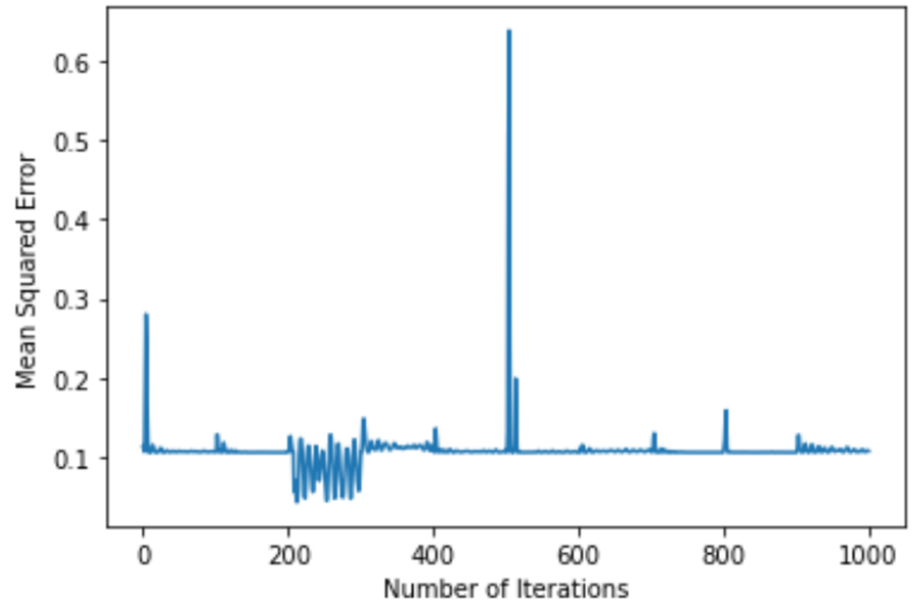
Figure 3

Cubic Function Approximation with Sigmoid Activation
10 Independent Experiments



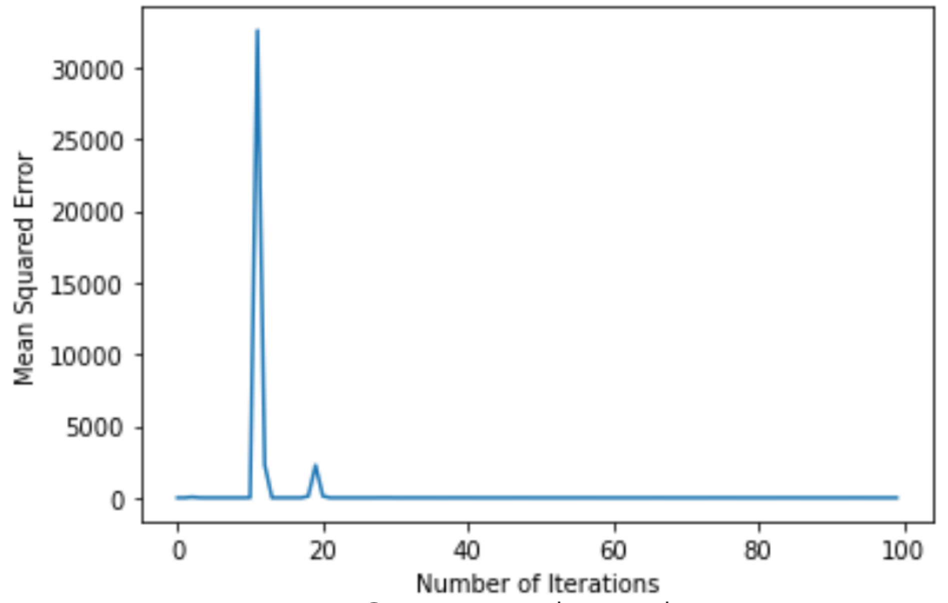
Over 10 experiments (each with 100 iterations)

Cubic Function Approximation with TanH Activation
10 Independent Experiments



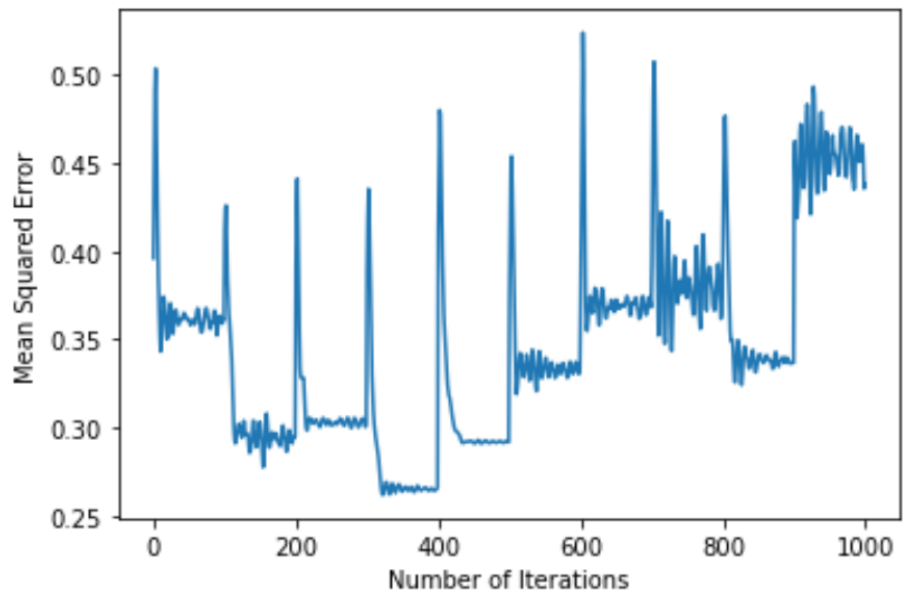
Over 10 experiments (each with 100 iterations)

Cubic Function Approximation with Linear Activation



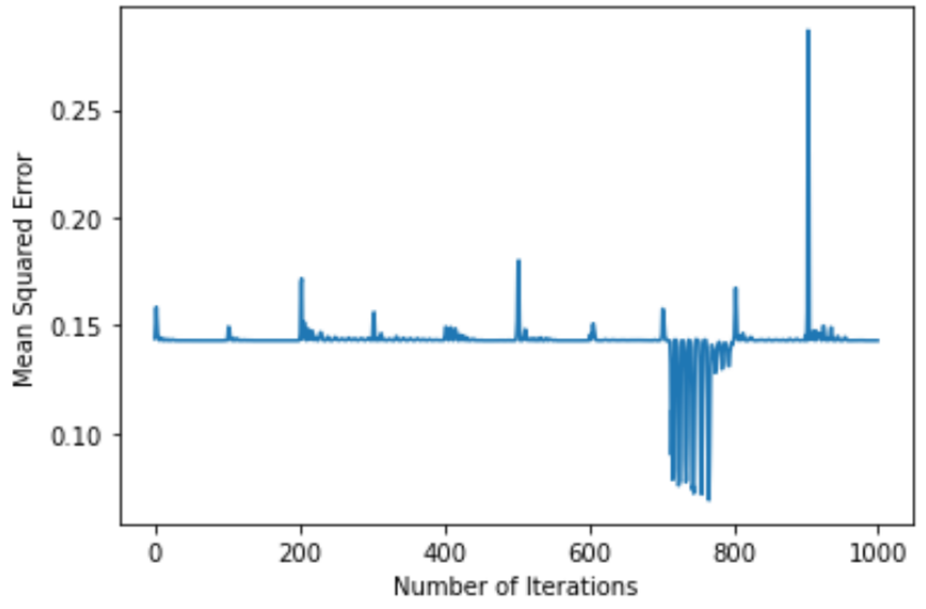
One representative experiment

Cubic Function Approximation with Sigmoid Activation without Output Data Range Transformation
10 Independent Experiments



Over 10 experiments (each with 100 iterations)

Cubic Function Approximation with TanH Activation without Output Data Range Transformation
10 Independent Experiments



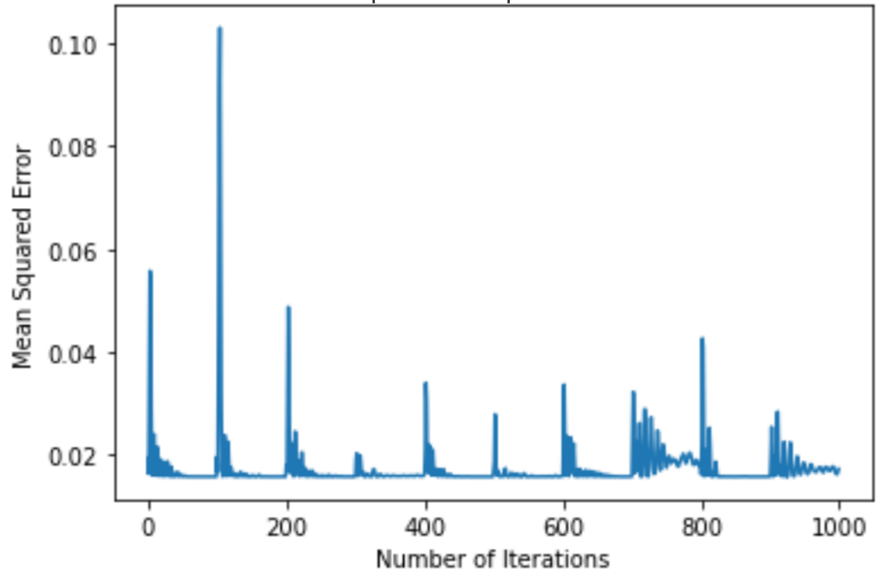
Over 10 experiments (each with 100 iterations)

Cubic function approximation:

PSO parameters	**Sigmoid activation function**		**Hyperbolic Tangent activation function**		**Linear activation function**
	Output Data Range Transformed	**No Transformation**	**Output Data Range Transformed**	**No Transformation**	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.9	0.9	0.9
Position range	-0.1 to 0.1	-0.1 to 0.1	-0.1 to 0.1	-0.1 to 0.1	-0.01 to 0.09
Velocity range	0.15 to 0.25	0.15 to 0.25	0.15 to 0.25	0.15 to 0.25	0.05 to 0.25
Cognitive component	1	1	1	1	1
Social component	3	3	2.7	2.7	3
Hidden Layer nodes	5	5	10	10	15
Number of Hidden Layers	45	45	60	60	30
Convergence Iteration	~60-100	Rare convergence	~40-60	~60-100	~5-20
Least MSE (approx)	~0.008	~0.26 to ~0.41	~0.1	~0.14	~0.14
Global Best (lowest MSE)	~0.008	~0.26 to ~0.41	~0.05 to ~0.1	~0.05 to ~0.14	~0.056 to ~0.142

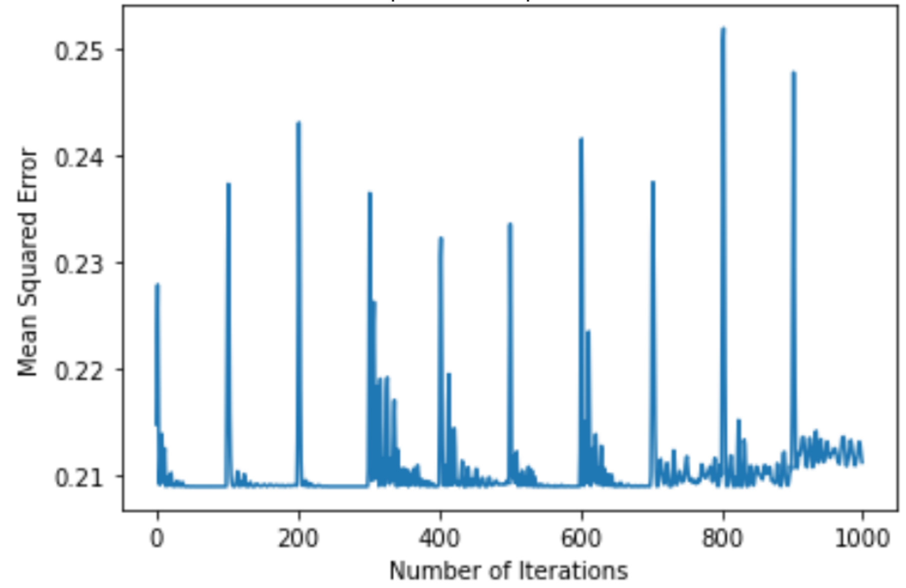
Figure 4

Sine Function Approximation with Sigmoid Activation
10 Independent Experiments



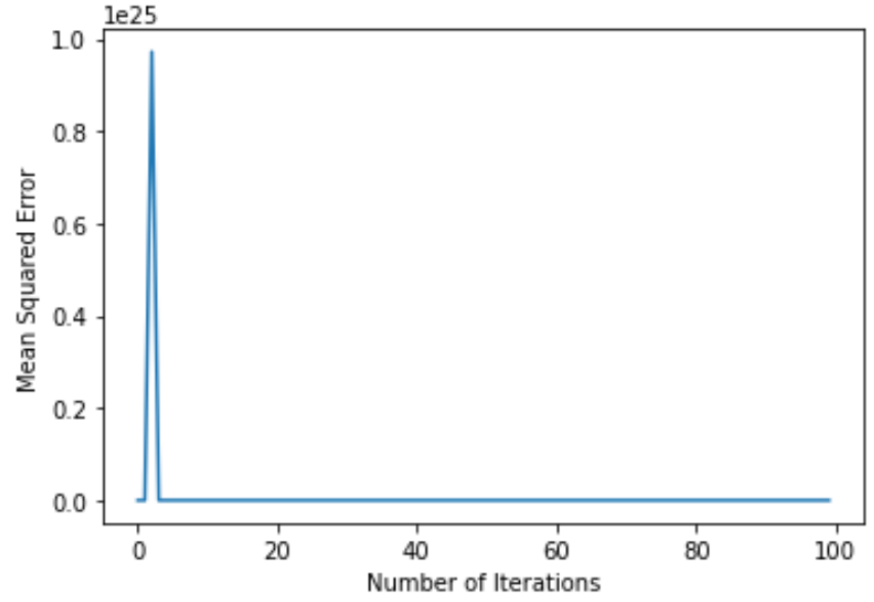
Over 10 experiments (each with 100 iterations)

Sine Function Approximation with TanH Activation
10 Independent Experiments



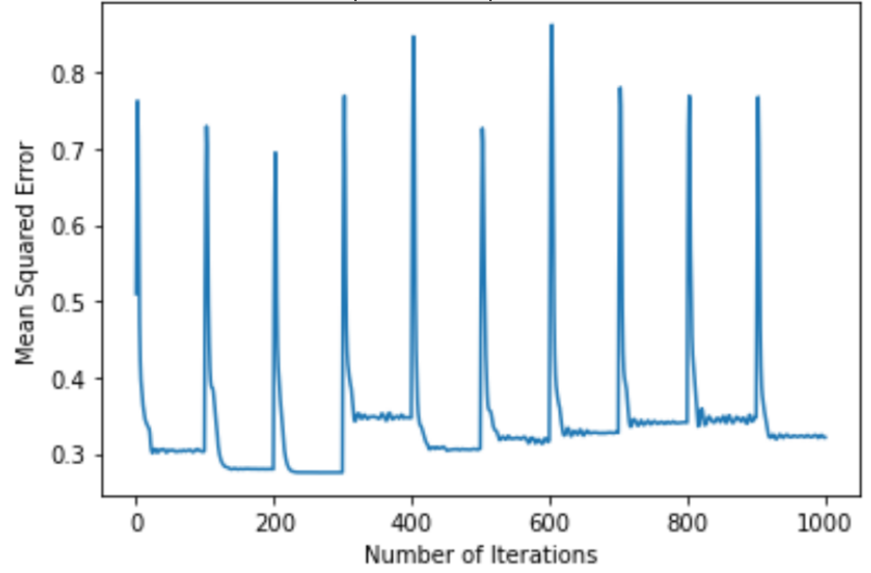
Over 10 experiments (each with 100 iterations)

Sine Function Approximation with Linear Activation



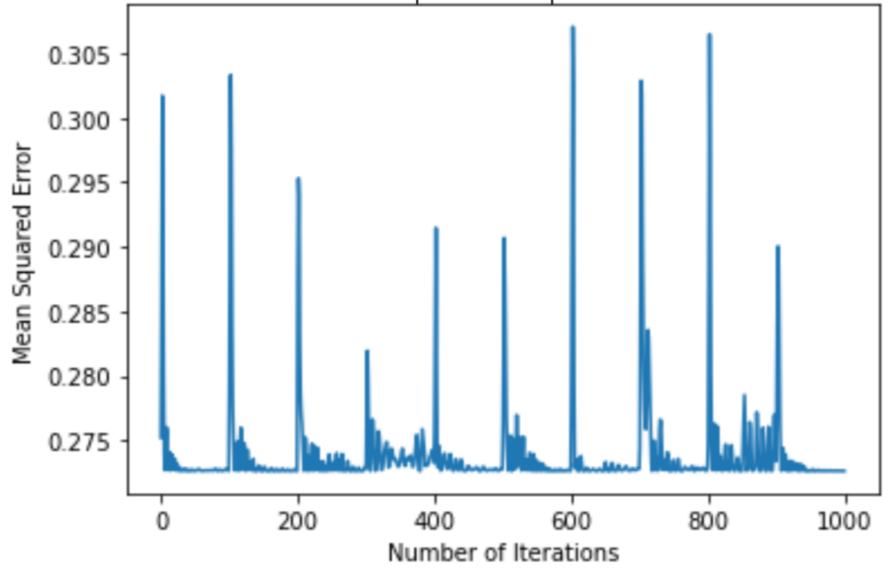
One representative experiment

Sine Function Approximation with Sigmoid Activation without Output Data Transformation
10 Independent Experiments



Over 10 experiments (each with 100 iterations)

Sine Function Approximation with TanH Activation without Output Data Transformation
10 Independent Experiments



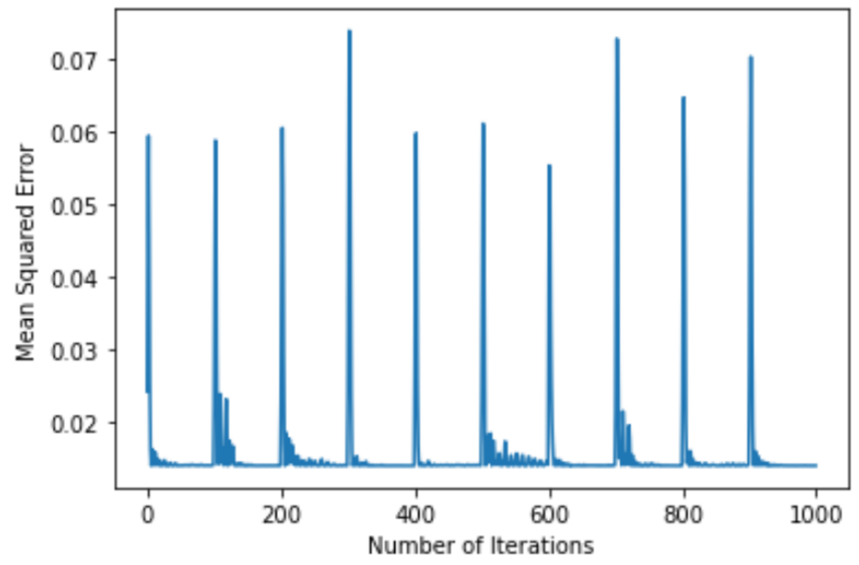
Over 10 experiments (each with 100 iterations)

Sine function approximation:

PSO parameters	Sigmoid activation function		Hyperbolic Tangent activation function		Linear activation function
	Output Data Range Transformed	No Transformation	Output Data Transformed	No Transformation	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.9	0.9	0.9
Position range	-0.1 to 0.09	-0.1 to 0.09	-0.1 to 0.09	-0.1 to 0.09	-0.1 to 0.09
Velocity range	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25
Cognitive component	1	1	1.4	1.4	1.7
Social component	3	3	2.9	2.9	3.2
Hidden Layer nodes	20	20	6	6	20
Number of Hidden Layers	60	60	80	80	60
Convergence Iterations	~40-60	~40-60	Very rarely convergent	~40-60 when convergent	~5
Least MSE (approx)	~0.016	~0.32	~0.21	~0.27	~0.27
Global Best (lowest MSE)	~0.015	~0.31	~0.21	~0.27	~0.27

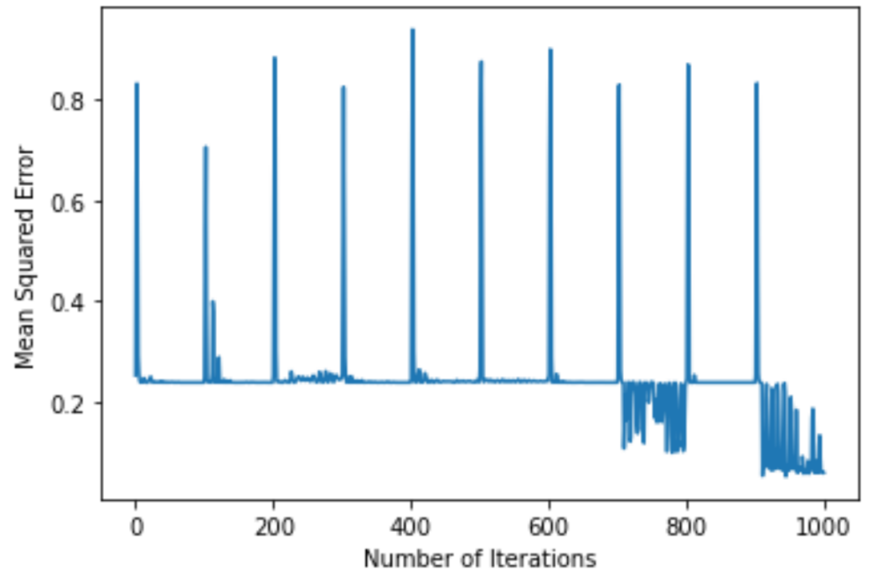
Figure 5

TanH Function Approximation with Sigmoid Activation
10 Independent Experiments



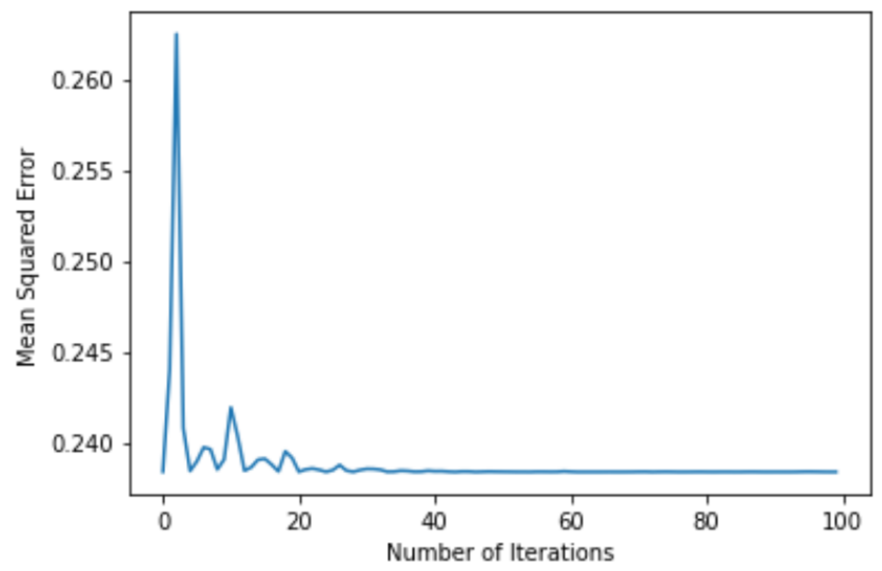
Over 10 experiments (each with 100 iterations)

TanH Function Approximation with TanH Activation
10 Independent Experiments



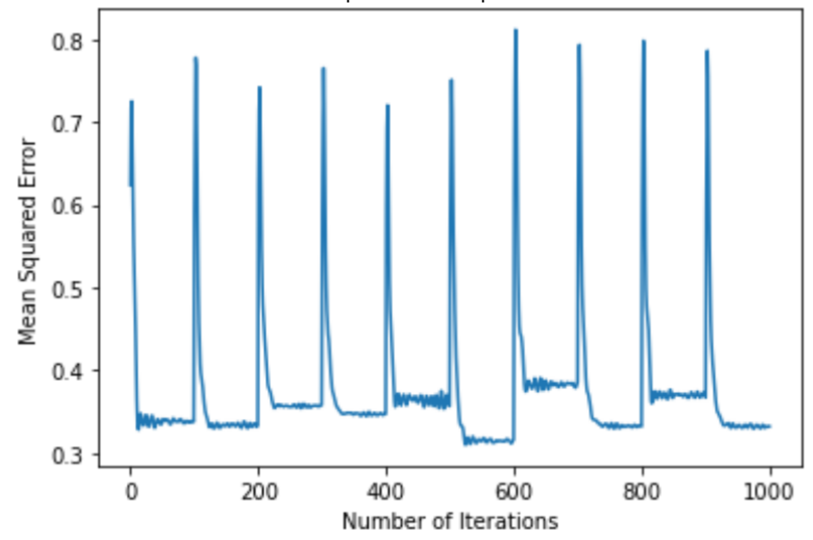
Over 10 experiments (each with 100 iterations)

TanH Function Approximation with Linear Activation



Represenative experiment

TanH Function Approximation with Sigmoid Activation without Output data range transformation
10 Independent Experiments



Over 10 experiments (each with 100 iterations)

TanH function approximation:

PSO parameters	**Sigmoid activation function**	**Sigmoid activation function (Without Output Data Range Transformation)**	**Hyperbolic Tangent activation function**	**Linear activation function**
Swarm size	10	10	10	10
Informants	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.9 to 0.91	0.9
Position range	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.01
Velocity range	0.05 to 0.25	0.05 to 0.25	0.15 to 0.2	0.15 to 0.25
Cognitive component	1.2	1.2	1.2	1.5
Social component	3.7	3.7	3.9	4
Hidden Layer nodes	15	15	15	15
Number of Hidden Layers	60	60	60	80
Convergence Iterations	~40-50	~20-40	~30-40	~30-40
Least MSE (approx)	~0.014	~0.33	~0.23	~0.23
Global Best (lowest MSE)	~0.014	~0.32	~0.23	~0.23

Figure 6

A

Complex function approximation with single hidden layer:

PSO parameters	**Sigmoid activation function**		**Hyperbolic Tangent activation function**		**Linear activation function**
	Output Data Range Transformed	**No Transformation**	**Output Data Range Transformed**	**No Transformation**	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.89	0.89	0.9
Position range	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09
Velocity range	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25
Cognitive component	1	1	1	1	1
Social component	3	3	2.7	2.7	3
Input Layer nodes	2	2	2	2	2
Hidden Layer nodes	10	30	30	30	25
Number of Hidden Layers	1	1	1	1	1
Convergence Iterations	~60-80	~20-30	Non-convergent	Not fully convergent	Not fully convergent
Least MSE (approx)	~0.006	~0.17	~0.12	~0.126	~0.13
Global Best (lowest MSE)	~0.006	~0.17	~0.12	~0.126	~0.13

B

XOR function approximation with single hidden layer:

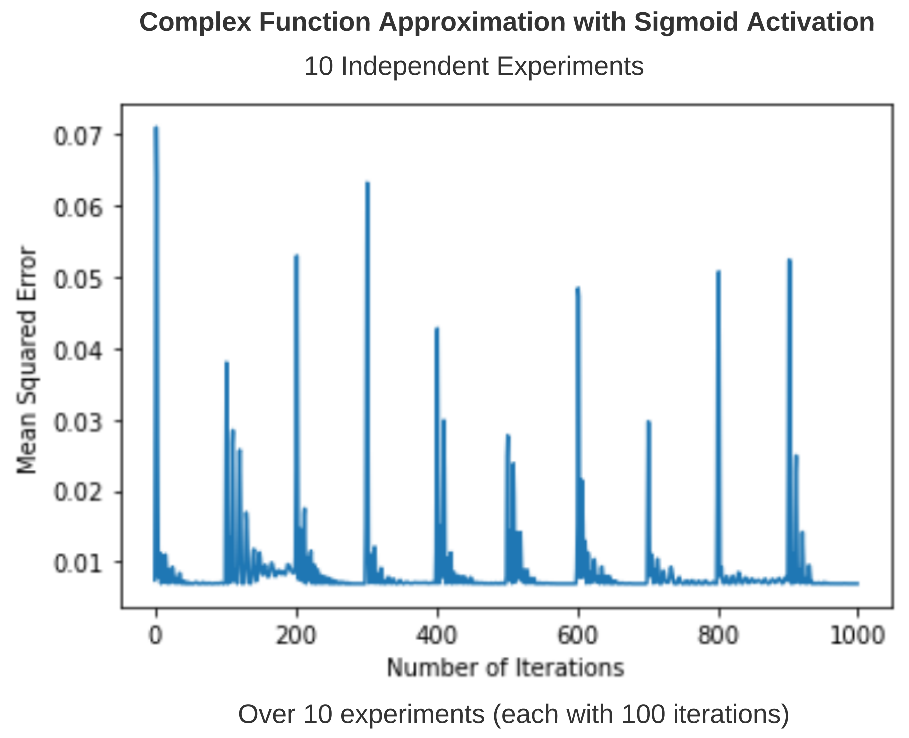
PSO parameters	**Sigmoid activation function**	**Hyperbolic Tangent activation function**
Swarm size	10	10
Informants	6	6
Epsilon	0.2	0.2
Alpha range	0.9	0.89
Position range	-0.1 to 0.1	-0.01 to 0.01
Velocity range	0.05 to 0.15	0.05 to 0.15
Cognitive component	1.8	1
Social component	3.2	3
Input Layer nodes	2	2
Hidden Layer nodes	5	5
Number of Hidden Layers	1	1
Convergence Iterations	~30-40 when convergent	~40
Least MSE (approx)	~0.013	~0.15
Global Best (lowest MSE)	~0.013	~0.15

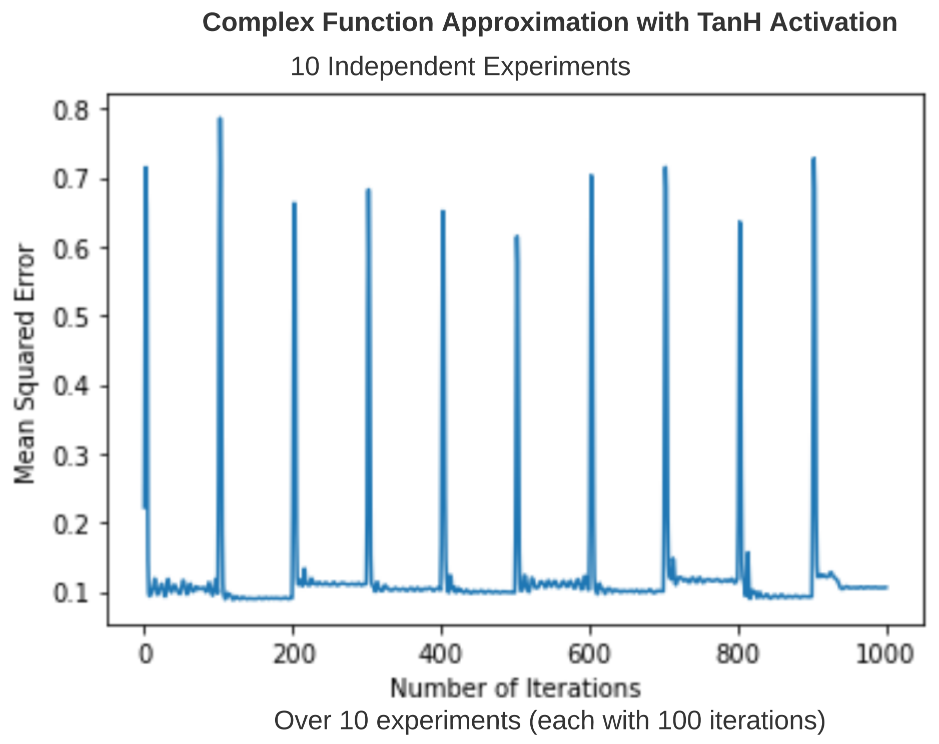
C

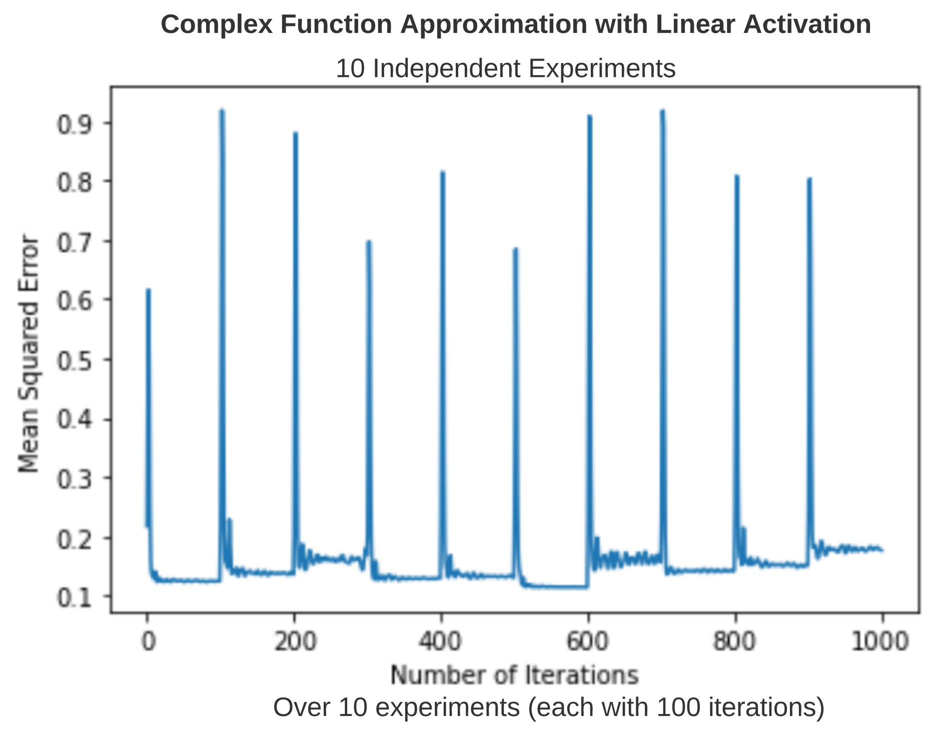
XOR function approximation without Output data range transformation with single hidden layer:

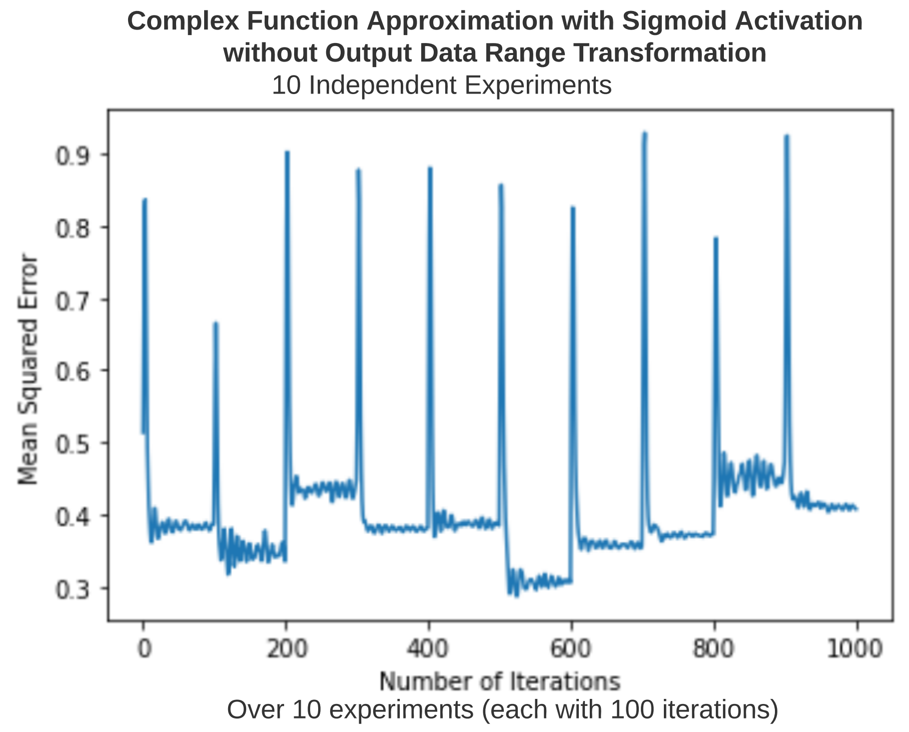
PSO parameters	**Sigmoid activation function**	**Hyperbolic Tangent activation function**
Swarm size	10	10
Informants	6	6
Epsilon	0.2	0.2
Alpha range	0.9	0.89
Position range	-0.1 to 0.1	-0.01 to 0.01
Velocity range	0.05 to 0.15	0.05 to 0.15
Cognitive component	1.8	1
Social component	3.2	3
Input Layer nodes	2	2
Hidden Layer nodes	5	5
Number of Hidden Layers	1	1
Convergence Iterations	Rarely convergent (~80)	~40
Least MSE (approx)	~0.24	~0.27
Global Best (lowest MSE)	~0.24	~0.27

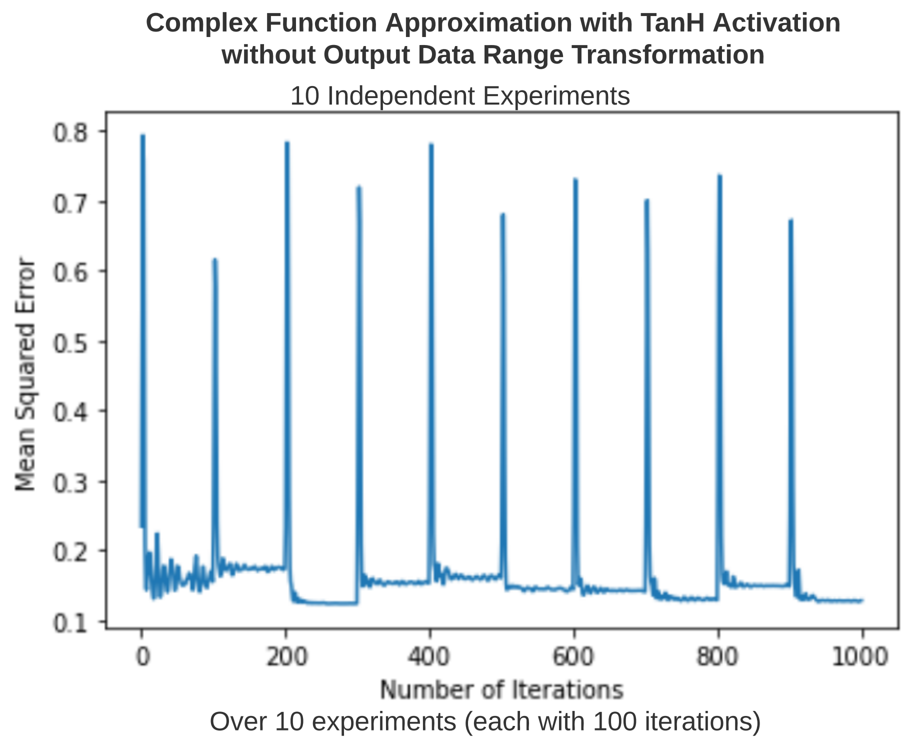
Figure 7







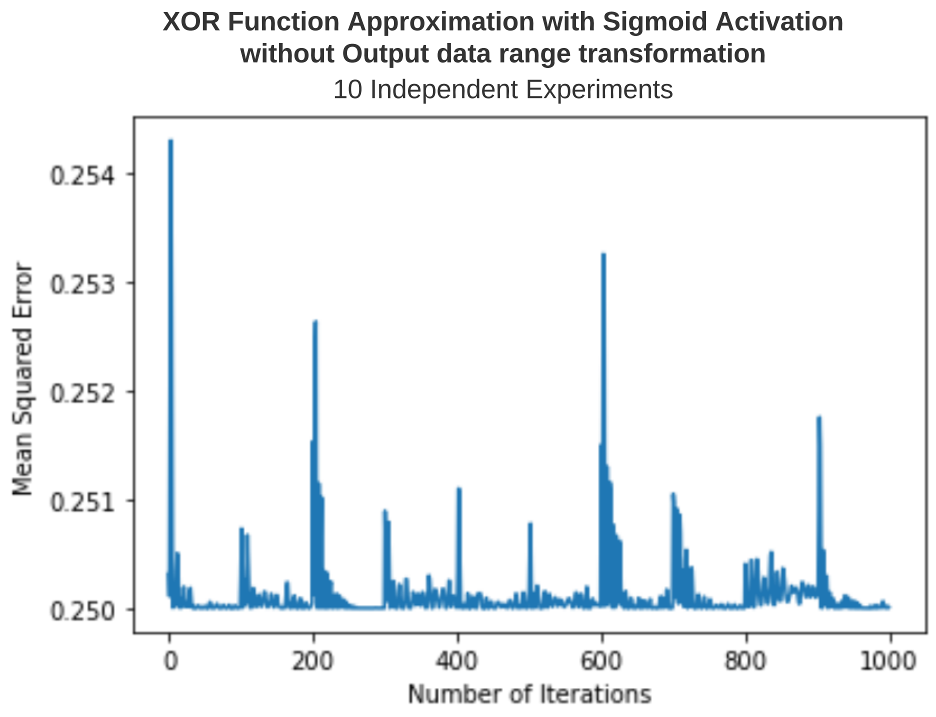




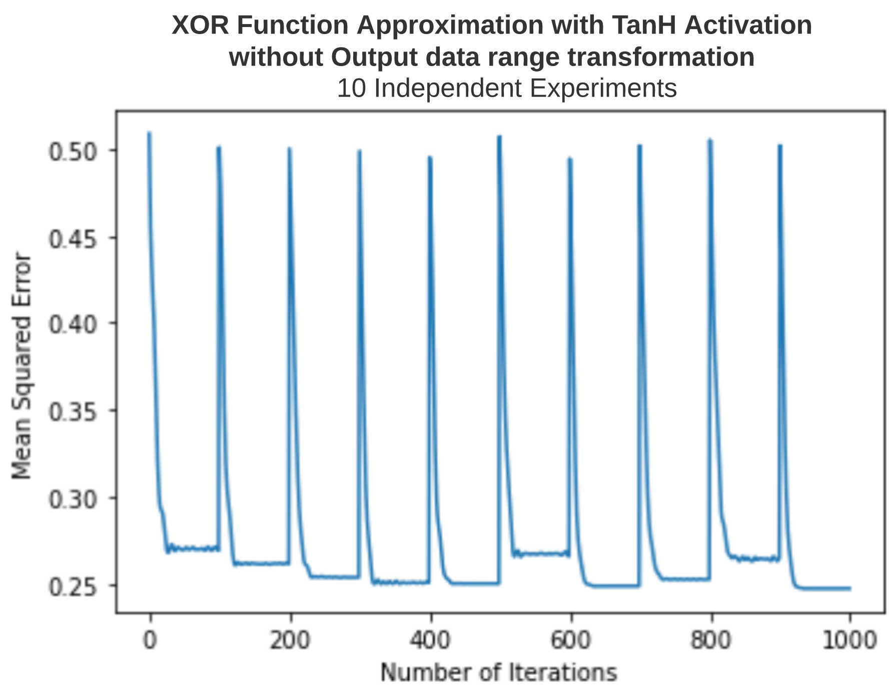
Complex function approximation:

PSO parameters	**Sigmoid activation function**		**Hyperbolic Tangent activation function**		**Linear activation function**
	Output Data Range Transformed	**No Transformation**	**Output Data Range Transformed**	**No Transformation**	
Swarm size	10	10	10	10	10
Informants	6	6	6	6	6
Epsilon	0.2	0.2	0.2	0.2	0.2
Alpha range	0.9	0.9	0.89	0.89	0.9
Position range	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09	-0.01 to 0.09
Velocity range	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25	0.05 to 0.25
Cognitive component	1	1	1	1	1
Social component	3	3	2.7	2.7	3
Input Layer nodes	2	2	2	2	2
Hidden Layer nodes	10	30	30	30	25
Number of Hidden Layers	60	70	60	60	40
Convergence Iterations	~40-60	Not fully convergent	~20-40 when convergent	~20-40	~60-80
Least MSE (approx)	~0.006	~0.4	~0.106	~0.127	~0.17
Global Best (lowest MSE)	~0.006	~0.4	~0.104	~0.125	~0.162

Figure 8



Over 10 experiments (each with 100 iterations)



Over 10 experiments (each with 100 iterations)

XOR function approximation without Output data range transformation:

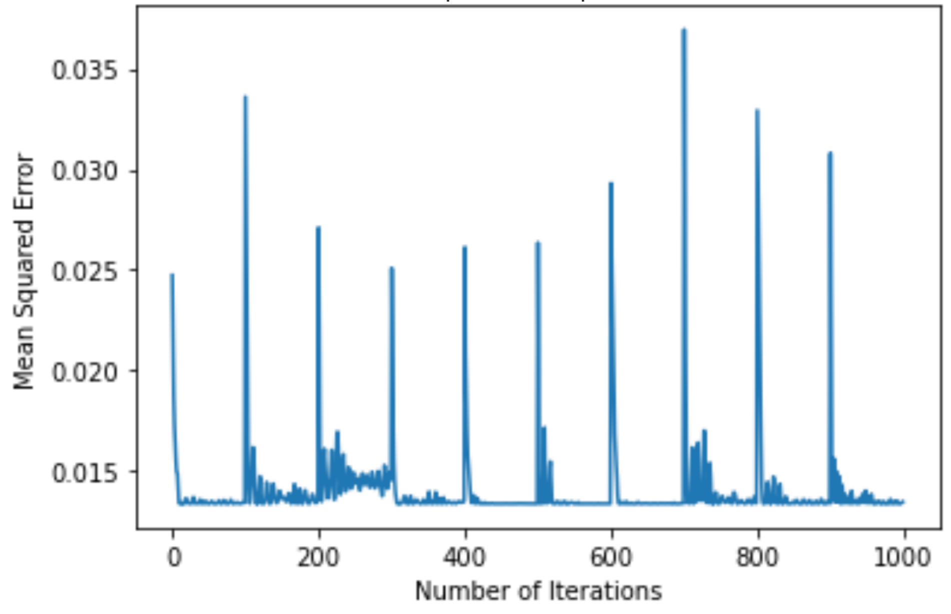
PSO parameters	Sigmoid activation function	Hyperbolic Tangent activation function
Swarm size	10	10
Informants	6	6
Epsilon	0.2	0.2
Alpha range	0.9	0.89
Position range	-0.1 to 0.1	-0.01 to 0.01
Velocity range	0.05 to 0.15	0.05 to 0.15
Cognitive component	1.8	1
Social component	3.2	3
Input Layer nodes	2	2
Hidden Layer nodes	5	5
Number of Hidden Layers	40	40
Convergence Iterations	Rarely convergent (~50)	~40
Least MSE (approx)	~0.25	~0.24
Global Best (lowest MSE)	~0.24	~0.24

Figure 9

XOR Function Approximation with Output Data-range Transformation

XOR Function Approximation with Sigmoid Activation

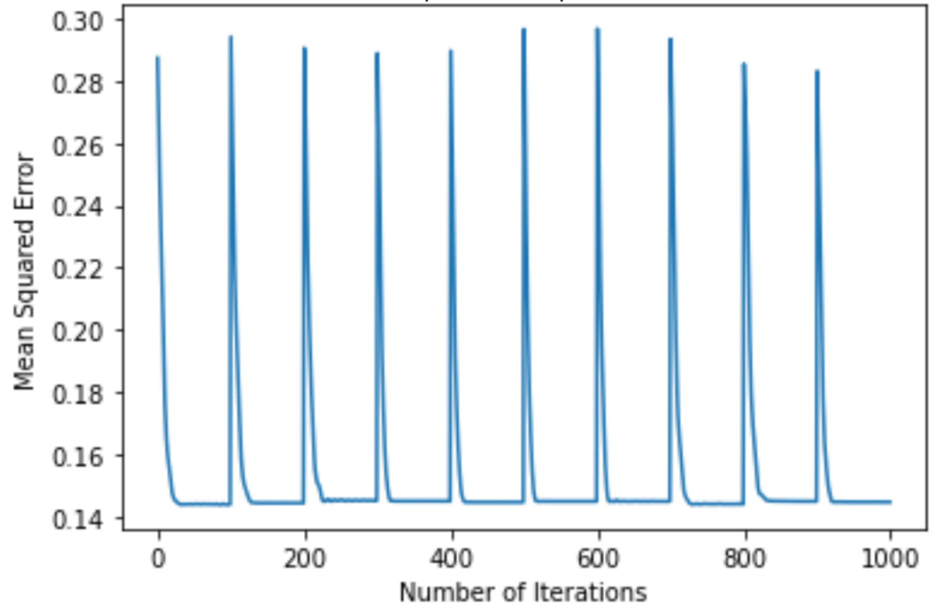
10 Independent Experiments



Over 10 experiments (each with 100 iterations)

XOR Function Approximation with TanH Activation

10 Independent Experiments



Over 10 experiments (each with 100 iterations)

XOR function approximation:

PSO parameters	**Sigmoid activation function**	**Hyperbolic Tangent activation function**
Swarm size	10	10
Informants	6	6
Epsilon	0.2	0.2
Alpha range	0.9	0.89
Position range	-0.1 to 0.1	-0.01 to 0.01
Velocity range	0.05 to 0.15	0.05 to 0.15
Cognitive component	1.8	1
Social component	3.2	3
Input Layer nodes	2	2
Hidden Layer nodes	5	5
Number of Hidden Layers	30	30
Convergence Iterations	~40-50 when convergent	~50
Least MSE (approx)	~0.013	~0.14
Global Best (lowest MSE)	~0.013	~0.14

Figure 10

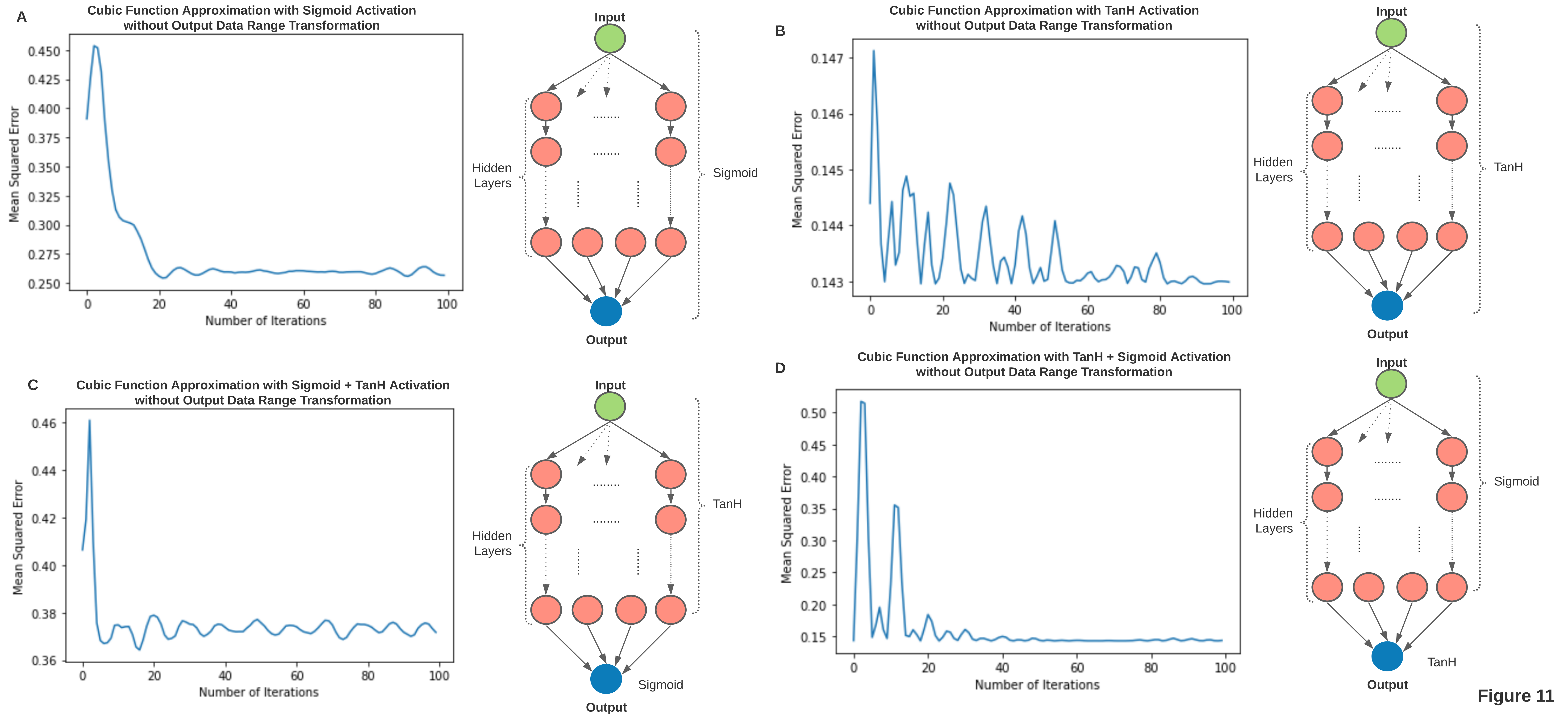

A
Cubic Function Approximation with Sigmoid Activation without Output Data Range Transformation
Mean Squared Error
Number of Iterations
Input
Hidden Layers
Sigmoid
Output
B
Cubic Function Approximation with TanH Activation without Output Data Range Transformation
Mean Squared Error
Number of Iterations
Input
Hidden Layers
TanH
Output
C
Cubic Function Approximation with Sigmoid + TanH Activation without Output Data Range Transformation
Mean Squared Error
Number of Iterations
Input
Hidden Layers
TanH
Sigmoid
Output
D
Cubic Function Approximation with TanH + Sigmoid Activation without Output Data Range Transformation
Mean Squared Error
Number of Iterations
Input
Hidden Layers
Sigmoid
TanH
Output

Figure 11