Neural Process Lett (2018) 47:993–1009
https://doi.org/10.1007/s11063-017-9718-z

Evolutionary Multi-task Learning for Modular Knowledge Representation in Neural Networks

Rohitash Chandra[1] · **Abhishek Gupta**[2] · **Yew-Soon Ong**[2] · **Chi-Keong Goh**[2]

Published online: 9 October 2017

Abstract The brain can be viewed as a complex modular structure with features of information processing through knowledge storage and retrieval. Modularity ensures that the knowledge is stored in a manner where any complications in certain modules do not affect the overall functionality of the brain. Although artificial neural networks have been very promising in prediction and recognition tasks, they are limited in terms of learning algorithms that can provide modularity in knowledge representation that could be helpful in using knowledge modules when needed. Multi-task learning enables learning algorithms to feature knowledge in general representation from several related tasks. There has not been much work done that incorporates multi-task learning for modular knowledge representation in neural networks. In this paper, we present multi-task learning for modular knowledge representation in neural networks via modular network topologies. In the proposed method, each task is defined by the selected regions in a network topology (module). Modular knowledge representation would be effective even if some of the neurons and connections are disrupted or removed from selected modules in the network. We demonstrate the effectiveness of the method using single hidden layer feedforward networks to learn selected n-bit parity problems of varying levels of difficulty. Furthermore, we apply the method to benchmark pattern classification problems. The simulation and experimental results, in general, show that the proposed method retains performance quality although the knowledge is represented as modules.

Keywords Evolutionary multitasking · Neuro-evolution · Modular design · Multi-task learning · Knowledge representation · Learning algorithms

✉ Rohitash Chandra
rohitash.chandra@sydney.edu.au

[1] Centre for Translational Data Science, The University of Sydney, Sydney, NSW 2006, Australia

[2] Rolls Royce @ NTU Corporate Lab, Nanyang Technological University, Nanyang View, Singapore, Singapore

1 Introduction

The brain can be viewed as a complex modular structure with features of information processing through knowledge storage and retrieval [1,2]. The modular viewpoint of the brain has motivated a number of studies in the past. A study was conducted for hierarchical modular decomposition that considered modules within modules of human brain functional networks using functional magnetic resonance imaging [2]. There have been computational studies on the roles and effects of modules in the brain. Recently, graph theory was used to study connectivity among modules in brain connectivity networks [3]. Moreover, the role of modularity in human learning was investigated by functional connectivity measurements of brain activity. This was implemented from initial training of the brain for mastery of a simple motor skill [1]. Hence, modularity could be viewed as an important feature of biological learning systems that could motivate artificial systems.

Modular artificial neural networks are motivated from repeating structures in nature [4]. They were introduced for visual recognition tasks that were trained by genetic algorithms which produced promising generalisation capability [4]. Block-based neural networks are modular structures that feature simultaneous optimization of structure and weights that can be implemented and reconfigured in digital hardware such as field programmable gate arrays [5]. An approach for evolving block neural networks was presented using radial basis networks [6] for the field of medical diagnosis in diabetes. A hardware implementation of block-based neural networks was given in [7]. A modular neural network was presented where the performance and connection costs were optimised through neuro-evolution which achieved better performance when compared to fully connected neural networks [8]. They have also been designed with the motivation to learn new tasks without forgetting old ones [9]. It was shown that modular networks learn new tasks faster from knowledge of previous tasks. Modular neural network architectures have also been beneficial for hardware implementations [10]. In particular, they enable smaller networks to be used as building blocks for a larger network. Such transfer of knowledge from simpler to progressively more complex tasks can be achieved by the process of multi-task learning [11]. The notion of multi-task learning has been prevalent in the machine learning community for at least the past two decades [11]. The main motivation has been the potential for exploiting relevant information available in related tasks by simultaneous learning using a shared knowledge representation.

Neuro-evolution views a learning problem as black-box optimisation where the solution space (given via weights and connections) is evolved through evolutionary algorithms [12,13]. Neuro-evolution has been used to evolve modular neural networks [4]. Essentially, any population-based evolutionary algorithm can be used for neuro-evolution. Evolutionary multi-task learning has the feature of representation of the solution space that can be heterogeneous, i.e., different neural network topologies as tasks with certain shared knowledge representation. The shared knowledge representation could mean that a certain region of the neural network defined by layer or hidden neurons is shared by other neural network topologies. Hence, we can investigate if multi-task learning can be used to develop modular neural network topologies via neuro-evolution.

Note that in the literature, the notion of "tasks" in multi-task learning is referred to a group of datasets that are usually from the same or related domains. This for example, could be seen in the wine quality classification based on physicochemical tests given by two datasets given by red and white wine which could be seen as different tasks [14]. In order to consider evolutionary multi-tasking for evolving modular neural networks, we regard a topology of modular neural network as a subtask in the multi-task learning framework. Rather than exploring tasks as different datasets given by existing multi-task learning problems, we only focus on conventional single task learning problems while focusing on the modules or building blocks of the neural network as subtasks. Hence, we refer to subtask as the neural network module throughout the paper. In this paper, we employ multi-task learning for modular knowledge representation in neural networks. The knowledge modules are defined by the network topologies and considered as tasks for selected conventional classification problems. Therefore, each of the tasks is defined by the network module features building blocks for transfer of knowledge. The method intends to produce a modular neural network that is effective even if some of the neurons and connections are removed from selected modules in the trained network. We demonstrate the effectiveness of the method using single hidden layer feedforward neural networks (FNNs) that learn different instances of the n-bit parity problem. Furthermore, we apply the method to benchmark pattern classification problems. This paper extended previous results given by evolutionary multi-task learning for feedforward neural networks [15].

The rest of the paper is organised as follows. Section 2 presents a review of related work while Sect. 3 presents the proposed method. Section 4 presents simulation with experiments and results. Section 5 concludes the paper with a discussion of future work.

2 Related Work

The notion of multi-task learning is natural due to the resemblance of learning several tasks or using knowledge from previous tasks in the human workforce [16]. A number of approaches have been presented that considers multi-task learning for different types of problems that include supervised and unsupervised learning [17–20]. A number of challenges have emerged in the implementation of multi-task learning. An approach to address negative transfer in multi-task learning has been through task grouping where knowledge transfer is performed only within each group [21,22]. Bakker et. al. [22] for instance, presented a Bayesian approach in which some of the model parameters were shared and others loosely connected through a joint prior distribution learnt from the data. Zhang and Yeung [21] presented a convex formulation for multi-task metric learning by modeling the task relationships in the form of a task covariance matrix. Moreover, Zhong et. al presented flexible multi-task learning framework to identify latent grouping structures in order to restrict negative knowledge transfer [23]. A number of very promising applications for multi-task learning exists. Most recently, it has been shown that multi-task learning across cancer drug datasets strongly improves the accuracy of drug prediction models [24].

Neuro-evolution employs evolutionary algorithms for training neural networks [12] which can be classified into direct [12,13], and indirect encoding strategies [30]. In direct encoding, every connection and neuron is specified directly and explicitly in the genotype [12,13]. Direct encoding has also been used in the evolution of feedforward networks for pattern recognition problems using conventional evolutionary algorithms [25], memetic based approaches [26] and cooperative coevolution [27–29]. In direct neuro-evolution, the evolutionary algorithm is essentially composed of a population of p individuals (chromosomes) to represent the weights and bias in the network. The individuals are mapped into the network where all the weights are encoded in a consecutive order; hence, each individual consists of total number of weights and biases in the network. In indirect encoding, the genotype specifies rules or some other structure for generating the network. Neuro-evolution of augmenting topologies (NEAT) has been a popular indirect encoding that begins evolution with the simplest network topology and adapts nodes and weights together during evolution [30]. Performance of direct and indirect encoding vary for specific problems. Although indirect encoding seem very intuitive and have biological motivations, in some cases, they have not outperformed direct encoding strategies [28,31].

3 Evolutionary Multi-task Learning for Modular Neural Networks

3.1 Preliminaries

Evolutionary multi-tasking for optimisation problems is motivated by multi-task learning from machine learning. Also known as multi-factorial optimisation [32], evolutionary multitasking has shown to facilitate autonomous knowledge exchange in the form of implicit genetically encoded information transfer. The different tasks are essentially optimisation problems that are evolved concurrently in a unified solution representation space, thereby often leading to improved convergence characteristics. The key idea of evolutionary multitasking is in exploiting the solution space of different but related optimisation problems during evolution. In doing so, conventional evolutionary operators such as selection, crossover and mutation are used with a method for transfer of solution from one optimisation problem into another in order to develop better quality solutions. Evolutionary multitasking has been used in several domains that includes real parameter optimisation [32], multi-objective optimisation [33] and path planning [34].

3.2 Modular Knowledge Representation

Although neural networks have been successfully applied to several real-world problems, their parameters such as the optimal network topology is typically based on trial-and-error based experimentation. Neuro-evolution has been popular alternative methods which can potentially feature the simultaneous evolution of weights and the network topology during the learning process. Multi-task learning typically considers the different tasks as different problems with some overlapping characteristics or features that could enable shared knowledge representation. The potential for multi-task learning through neuro-evolution has not been fully explored.

Knowledge representation in the case of neural networks refers to how fundamental knowledge has been organised and represented. Shared knowledge representation refers to a set of weights that could be effective for two different problems or datasets taken from the same or related domain. An example would be a wine classification problem selected from two different regions that contain a different number of samples and features. These types of problems have also been explored through transfer learning where knowledge from a problem is refined and used for related problems [35].

Modularity ensures that the knowledge is stored in a manner where any complications in certain modules do not affect the overall functionality of the neural network. Given that a conventional neural network is trained to learn a recognition task; for instance, learning to recognize facial expressions from data of extracted facial features. The neural network is trained with a fixed topology in terms of h number of hidden neurons which has been selected for optimal results. In the future, a substantially large number of samples are added when compared to the previous one. Therefore, additional hidden neurons would be needed to learn the new set of samples while retaining knowledge from the previous dataset. The weights (knowledge) from the previous neural network would need to be retained while increasing

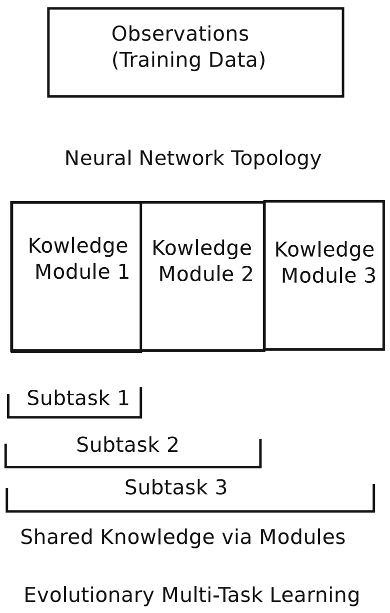

Fig. 1 The development of knowledge modules for the related subtasks (Subtask 1, Subtask 2, and Subtask 3) that feature the modules. Note that the multi-task learning considers a single classification problem for learning modules of knowledge defined by regions of the neural network

the number of hidden neurons, $h + n$. In typical neural network learning, this case would risk losing previous knowledge when additional neurons are added. Therefore, modularity is needed in order to ensure that previous knowledge is retained while new knowledge is added using the new dataset and weights connected to the additional hidden neurons. This is the main motivation of using the modular neural network as it can enable better transfer of knowledge when additional subtasks or data is considered in the future. If the additional data set is not useful, the additional knowledge can be easily discarded without affecting the overall functionality of the neural network as previous knowledge would be secured through modular knowledge representation. Hence, modularity enforced through evolutionary multi-task learning helps the neural system not to forget knowledge as featured in biological neural systems [1,2].

Figure 1 gives a general top-down formulation of the multi-task learning that considers a single classification problem for learning modules of knowledge defined by regions of the neural network. Module 1 is the fundamental knowledge module while Module 2 and 3 are the additional knowledge modules. The fundamental knowledge module is shared by the rest of the tasks. Note that the fundamental knowledge module refers to the first subtask. The additional tasks feature the fundamental knowledge module and those from previous tasks, for instance, subtask $\Omega_3 = [\phi_1, \phi_2, \phi_3]$, where ϕ refers to the the knowledge module given by selected weights in the network partitioned by different number of hidden neurons. The selected neural network regions that make up the knowledge modules are explicitly shown in Fig. 2.

In the proposed method, multi-task learning is used for a classification problem where different subtasks refer to the respective knowledge modules. It employs direct encoding for weight representation in the evolutionary algorithm. The knowledge modules are evolved through evolutionary multi-task learning. This could be implicitly seen as a heterogeneous transfer learning problem. Therefore, the overall learning task is formulated with varied length real-parameter chromosomes in the evolutionary algorithm as shown in Fig. 2. Hence,

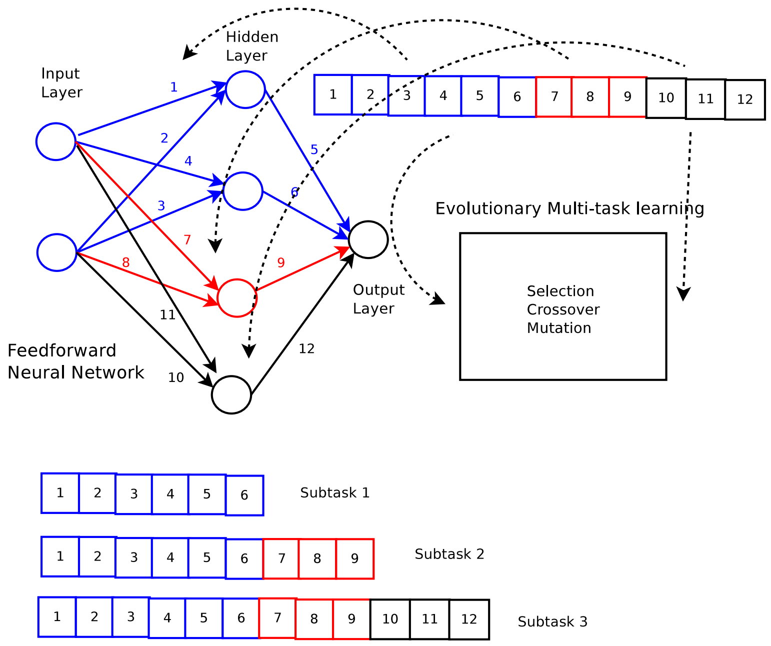

Fig. 2 Subtasks in co-evolutionary multi-task learning. The synapses in the network topology make up the modules that constitute the different subtasks. In this example, Subtask 1 employs a module with 2 hidden neurons while the remaining tasks concatenate other modules with additional hidden neurons

Alg. 1 Evolutionary multi-task learning

```
Step 1: Define different subtasks given by number of hidden neurons
Subtask 1 ($p$ hidden neurons)
Subtask 2 ($q$ hidden neurons)
Subtask 3 ($r$ hidden neurons)
Step 2: Initialise $popsize$ individuals in the unified search space
i) Arbitrarily associate every individual with any one of Subtask (1, 2, or 3)
ii) Evaluate individuals for their associated Subtask
for each generation until termination do
    i) Select and create new offspring via crossover and mutation
    ii) Associate each offspring with any one of Subtask (1, 2, or 3) via imitation
    ii) Evaluate offspring for their associated Subtask only
    iii) Select ($popsize/3$) elite individuals from each of the Subtasks for next generation
end for
```

we employ the strategies for creating new offspring using genetic operators proposed in evolutionary multi-tasking [32].

Algorithm 1 gives further details of the proposed evolutionary multi-task learning (EMTL) method. Without loss of generality, we assume there exists three subtasks at a time. The algorithm follows a similar work-flow as the multi-factorial evolutionary algorithm proposed in [32]. The critical step for successful evolutionary multi-task learning is the formulation of the unified solution representation scheme encompassing the search space for the respective subtasks. Such unification can be achieved in a straightforward manner in a multi-dimensional real space for real-parameter optimization (as is the case with training neural network weights). The main challenge is with regard to accommodating for the heterogeneity in search space dimensionality of constitutive subtasks, hence, we follow the methodology proposed in [32]. If the three subtasks in Algorithm 1 possess dimensionality D_1, D_2, and D_3, respectively, then the dimensionality of the unified search space is given by $D_{multitask} = \max\{D_1, D_2, D_3\}$. Thus, a candidate solution in the unified space is characterized by a vector of $D_{multitask}$ elements. While evaluating an individual associated with the jth subtask, we simply refer to (or extract) D_j relevant elements from the list of $D_{multitask}$ elements. The motivation behind employing such a search space merging scheme (instead of simplistic concatenation) is to facilitate the underlying evolutionary algorithm in providing implicit information transfer across related subtasks. This transfer, at least within the present framework, is achieved during the crossover operation as discussed in [33]. At this juncture, an appropriate choice of D_j elements to be extracted for the jth subtask is crucial for effective multi-task learning. This must be customized by accounting for the properties of the multi-task learning instance at hand. For the present example, we assume p (hidden neurons in Subtask 1) $< q$ (hidden neurons in Subtask 2) $< r$ (hidden neurons in Subtask 3); which implies that $D_1 < D_2 < D_3$. Thus, we enforce that all weights connected to the first p hidden neurons (regardless of Subtask [1, 2, or 3]) refer to the same $D_p = D_1$ elements among $D_{multitask}$. Similarly, all the weights connected to the next $q - p$ hidden neurons (regardless of Subtask [2 or 3]) refer to the same $D_{q-p} = D_2 - D_1$ elements among $D_{multitask}$. Finally, all the weights connected to the last $r - q$ hidden neurons in Subtask 3 refer to the last $D_{r-q} = D_3 - D_2$ elements among $D_{multitask}$. Notice that in the proposed procedure $D_p + D_{q-p} + D_{r-q} = D_{multitask}$. The encoding process is further illustrated in Fig. 2.

To conclude the description of the algorithm, we highlight the association of every individual with *any one* subtask in the multi-task learning environment. Doing so primarily leads to a saving in the computational cost as evaluating every individual exhaustively for every subtask is likely to be expensive. While in the initial population the associations are randomly assigned (whilst ensuring uniform representation for all subtasks), an assignment strategy based on the memetic concept of *vertical cultural transmission* [36] is adopted in subsequent generations for the genetically modified offspring. In particular, it is prescribed that an offspring randomly *imitates* the association of any one of the parents from which it is created. For further details, refer to [32].

4 Experiments and Results

This section presents the simulation with experimental results for evolutionary multi-task learning (EMTL) for modular knowledge representation in selected n-bit even parity and benchmark classification problems.

4.1 Design of Experiments

We design the experiments in order to demonstrate that modular knowledge representation through multi-task learning (EMTL) reaches the same level of performance as conventional non-modular approach (single-task learning). Therefore, we compare EMTL with the performance for the evolutionary single-task learning (ESTL) approach given a different number of hidden neurons.

The ESTL is implemented through an evolutionary algorithm that employs a population size ($popsize$) of 30 individuals. Accordingly, a population of 90 individuals is employed for

EMTL which executes three self-contained subtasks at once. The algorithm terminates once a total of at least 30,000 function evaluations are completed for each subtask. Furthermore, note that identical crossover and mutation operators are employed for both methods. In particular, a simulated binary crossover (SBX) operator (with distribution index of 2) [37] and polynomial mutation (with distribution index of 5) are used [38]. The choice of low distribution indices encouraging enhanced exploration is preferred as we employ an elitist selection strategy ensuring preservation of high quality genetic material.

The mean squared error (MSE) given in Eq. 1 and the classification performance are used to evaluate the performance of the respective methods.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2 \tag{1}$$

where y_i and $\hat{y}_i$ are the observed and predicted output, respectively. N is the length of the data.

4.2 Experimental Design

The n-bit parity problem essentially is defined as the number of even parity bits in a binary string. Typically, n is the size of the string and all the possible permutations are used to make a dataset.

The n-bit parity problem has been used in the literature to demonstrate the effectiveness of a particular neural network architecture [39] or training algorithm [40]. For instance, the n-bit parity problem was solved using a neural network that allowed direct connections between the input layer and the output layer and trained with linear programming [39]. The effectiveness of certain neuro-evolution methods has also been demonstrated using n-bit problem [40–42]. Although 3 or 4 bit parity problems have also been used in the literature [41,42], we specifically chose 4, 6 and 8-bit parity problems in order to demonstrate the effectiveness of the proposed method in order to cater for increasing levels of difficulty.

We selected Ionosphere, Cancer, Heart, Tic-Tac-Toe, Balloon and Credit Approval task from the University of California, Irvine (UCI) Machine Learning Data repository [43]. Table 1 shows the configuration for the classification problems where the experimental design settings are given. Note that the different modules are defined by number of hidden neurons. Hence, $Modules(hidden) = [6, 8, 10]$ in the case of Ionosphere problem refers to neural network with specific regions for the respective module. An example is shown in Fig. 2 where $Modules(hidden) = [2, 3, 4]$. The data partitioning was done in a consecutive order where the first 70% was used for training while the remaining was used for testing. We only considered binary classification problems for our experiments, however, the proposed method could be extended to problems with more than two classes. The results are further compared to the canonical backpropgation method that employs gradient descent (GD) as shown in Table 8. The Python implementation of the backpropgation method is given online.[1]

4.3 Results for the n-Bit Parity Problem

The results for 4, 6 and 8 bit problems that compare EMTL with ESTL are given in Tables 2, 3, 4, 5, 6 and 7, for MSE and classification performance, respectively. In Tables 2, 4 and 6, EMTL achieves better performance than ESTL for all the cases for the respective problems.

[1] https://github.com/rohitash-chandra/VanillaFNN-Python.

Table 1 Configuration for classification problems

Dataset	Features	Classes	Instances	Modules (hidden)	Max-FE
Ionosphere	34	2	351	[6, 8, 10]	30,000
Cancer	9	2	699	[4, 6, 8]	15,000
Heart	13	2	270	[14, 16, 18]	300,000
Tic-Tac-Toe	9	2	958	[20, 22, 24]	300,000
Balloon	4	2	20	[3, 5, 7]	15,000
Credit approval	15	2	690	[18, 20, 22]	90,000

Table 2 MSE (SD) of EMTL versus ESTL for the 4-bit even parity problem

Method	Subtask 1	Subtask 2	Subtask 3
EMTL (4, 5, 6)	0.0653 (0.0591)	0.0472 (0.0433)	0.0361 (0.0326)
ESTL (4, 5, 6)	0.1557 (0.0364)	0.1020 (0.0232)	0.0699 (0.0285)
EMTL (6, 7, 8)	0.0071 (0.0219)	0.0058 (0.0178)	0.0043 (0.0132)
ESTL (6, 7, 8)	0.0699 (0.0285)	0.0427 (0.0336)	0.0325 (0.0309)

Table 3 Classification performance (SD) of EMTL versus ESTL for the 4-bit even parity problem

Method	Subtask 1	Subtask 2	Subtask 3
EMTL (4, 5, 6)	86.6667% (3.1714)	90.4167% (4.2590)	95.4167% (5.4272)
ESTL (4, 5, 6)	73.3333% (15.3912)	87.0833% (6.1267)	90.4167 % (6.5104)
EMTL (6, 7, 8)	91.8750 % (4.6857)	94.3750 % (3.0041)	98.1250% (2.9131)
ESTL (6, 7, 8)	90.4167 % (6.5104)	94.1667 % (4.9057)	94.7917 % (5.2119)

Table 4 MSE (SD) of EMTL versus ESTL for the 6-bit even parity problem

Method	Subtask 1	Subtask 2	Subtask 3
EMTL (4, 5, 6)	0.0977 (0.0240)	0.0789 (0.0264)	0.0639 (0.0224)
ESTL (4, 5, 6)	0.1343 (0.0392)	0.1153 (0.0235)	0.0949 (0.0254)
EMTL (5, 6, 7)	0.0745 (0.0321)	0.0581 (0.0287)	0.0459 (0.0201)
ESTL (5, 6, 7)	0.1153 (0.0235)	0.0949 (0.0254)	0.0695 (0.0304)
EMTL (6, 7, 8)	0.0642 (0.0221)	0.0459 (0.0148)	0.0492 (0.0269)
ESTL (6, 7, 8)	0.0949 (0.0254)	0.0695 (0.0304)	0.0643 (0.0272)

The same trend is also given for the classification performance in Tables 3, 5 and 7. The results clearly demonstrate that the EMTL is better than ESTL given that the overall optimisation time. Note that the Tables show the results for the number of hidden neurons *(i, j, k)* in the modular and non-modular methods $EMTL(i, j, k)$ and $ESTL(i, j, k)$, respectively.

Figures 3 and 4 show the convergence trend for the respective strategies for the 8-bit even parity problem. In the case, the mean MSE for 30 experimental runs is shown at different stages of evolution. It is clear that EMTL converges with higher quality solutions.

Table 5 Classification performance (SD) of EMTL versus ESTL for the 6-bit even parity problem

Method	Subtask 1	Subtask 2	Subtask 3
EMTL (4, 5, 6)	85.3125 % (8.8064)	90.5208% (6.1129)	93.1250 % (4.0538)
ESTL (4, 5, 6)	81.1458 % (10.0337)	83.4375 % (9.2174	86.1458 % (8.1214)
EMTL (5, 6, 7)	90.9896% (5.1198)	91.8750 % (4.0483)	91.9922 % (5.4425)
ESTL (5, 6, 7)	81.9010 % (9.5784)	84.1406 % (6.1272)	85.7813 % (6.8659)
EMTL (6, 7, 8)	92.7604% (4.5597)	95.6250 % (2.6081)	95.0000 % (4.5211)
ESTL (6, 7, 8)	86.1458 % (8.1214)	92.0833 % (7.4533)	92.5521 % (4.7087)

Table 6 MSE (SD) of EMTL versus ESTL for the 8-bit problem

Method	Subtask 1	Subtask 2	Subtask 3
EMTL (5, 6, 7)	0.1108 (0.0249)	0.0949 (0.0257)	0.0861 (0.0232)
ESTL (5, 6, 7)	0.1349 (0.0279)	0.1223 (0.0279)	0.1269 (0.0322)
EMTL (7, 8, 9)	0.0843 (0.0274)	0.0661 (0.0178)	0.0618 (0.0161)
ESTL (7, 8, 9)	0.1269 (0.0322)	0.1093 (0.0275)	0.0936 (0.0217)
EMTL (8, 9, 10)	0.0759 (0.0269)	0.0643 (0.0218)	0.0653 (0.0287)
ESTL (8, 9, 10)	0.1093 (0.0275)	0.0936 (0.0217)	0.0888 (0.0239)

Table 7 Classification performance (SD) of EMTL versus ESTL for the 8-bit even parity problem

Method	Subtask 1	Subtask 2	Subtask 3
EMTL (5, 6, 7)	86.5234 % (4.3712)	88.6198 % (4.6375)	89.0885 % (5.3569)
ESTL (5, 6, 7)	82.0313 % (6.4415)	81.9010 % (9.5784)	84.1406 % (6.1272)
EMTL (7, 8, 9)	90.5859 % (5.4334)	92.9427 % (3.2954)	93.5417 % (2.3810)
ESTL (7, 8, 9)	84.1406 % (6.1272)	85.7813 % (6.8659)	88.4115 % (5.0280)
EMTL (8, 9, 10)	91.4714 % (4.9909)	93.2031 % (3.6867)	93.4766 % (4.6978)
ESTL (8, 9, 10)	85.7813 % (6.8659)	88.4115 % (5.0280)	89.4271 % (4.3594)

4.4 Results for Benchmark Classification Problems

We report the classification performance for the respective problem in Figs. 5, 6, 7, 8, 9 and 10 that highlight both the training and generalisation performance. The mean and standard deviation of 30 independent experimental runs are reported as histogram and error bars for the respective cases. It is clear for all the respective problems that EMTL achieves the same performance when compared to non-modular representation using ESTL. Hence, there is no loss in quality of performance by modular knowledge representation while it has the advantage of using knowledge modules.

In Table 8, we observe that EMTL and ESTL for Subtask 1 (ST1) gives similar performance when compared to backpropgation via GD for training dataset in almost all cases, except for the Heart and Credit problem. Note that both ESTL and EMTL give better performance for generalization performance for the Ionosphere and the Tic-Tac-Toe problem. The strengths of the gradient descent and evolutionary algorithms seems to vary for different times of problems

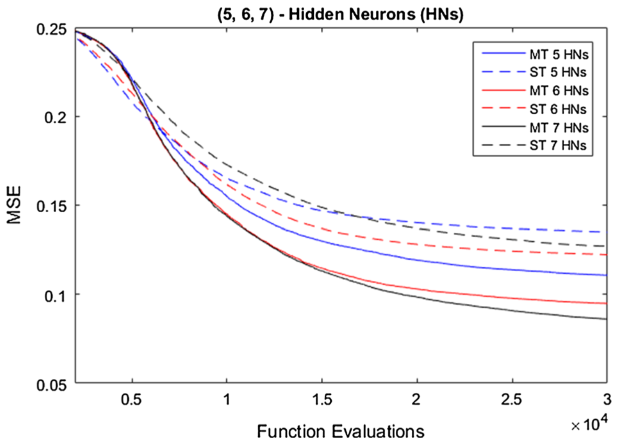

Fig. 3 Convergence trend for (5, 6, 7) neurons for the 8-bit even parity problem. Note that the multi-task learning strategies (MT) with solid lines converge with higher solution quality when compared to single-task learning (ST) as the number of function evaluation increases

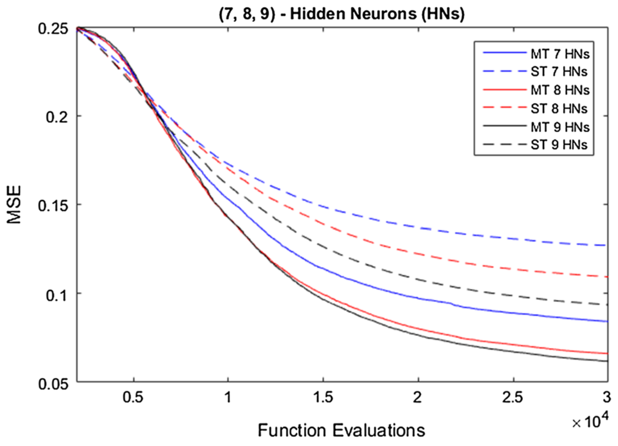

Fig. 4 Convergence trend for (7, 8, 9) neurons for the 8-bit even parity problem. The multi-task learning strategies (MT) with solid lines converge with higher solution quality when compared to single-task learning (ST) as the number of function evaluation increases

according to their optimisation landscape which have different global (evolutionary) and local (gradient descent) search requirements. This opens up directions for future work for them to be combined for modular neural networks.

4.5 Discussion

The experiments were conducted to observe the behaviour and quality of performance of modular knowledge representation using evolutionary multi-task learning. In the case for the n-bit even parity problems, in general, we observed that the multi-task learning performed better than non-modular single-task learning. This could be due to the way the problem was decomposed using a dynamic programming approach where the main task was divided into subtasks as building blocks that transfer to larger subtasks given by larger network topologies.

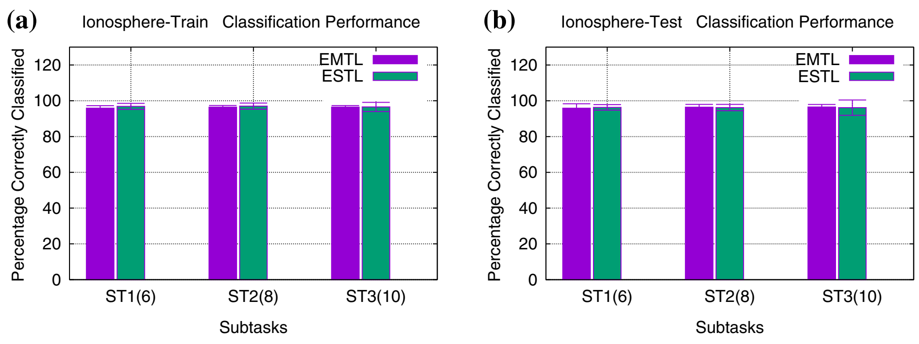

Fig. 5 Performance evaluation of featured modularity using EMTL verses non-modular ESTL across different subtasks [ST1(n), ST2(n), ST3(n)]. **a** Training dataset and **b** test dataset

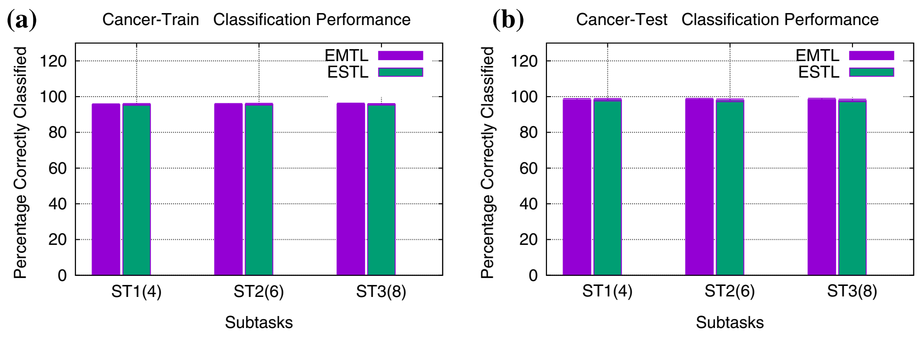

Fig. 6 Performance evaluation of featured modularity using EMTL verses non-modular ESTL across different subtasks [ST1(n), ST2(n), ST3(n)]. **a** Training dataset and **b** test dataset

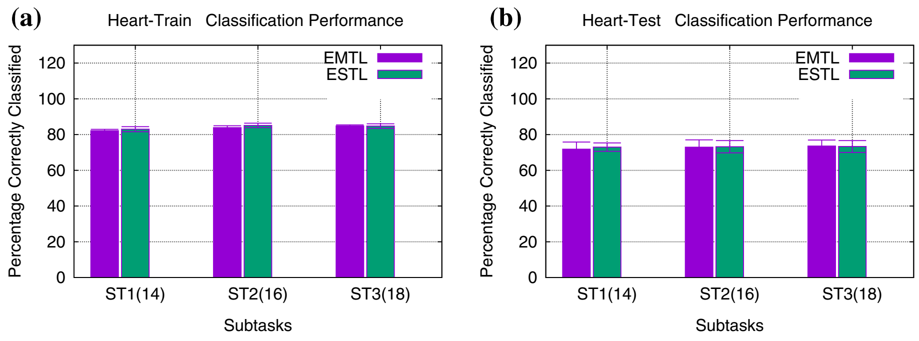

Fig. 7 Performance evaluation of featured modularity using EMTL verses non-modular ESTL across different subtasks [ST1(n), ST2(n), ST3(n)]. **a** Training dataset and **b** test dataset

The results show that the knowledge in smaller subtasks was effectively transferred and refined by the larger subtasks.

One can argue that the design of experiments for the n-bit even parity are biased towards the multi-task learning approach as it was given more computational budget (90,000) when compared to single task learning approach (30,000). Note that this is because the multi-task learning approach is solving 3 subtasks at the same time exhibiting parallelism, whereas,

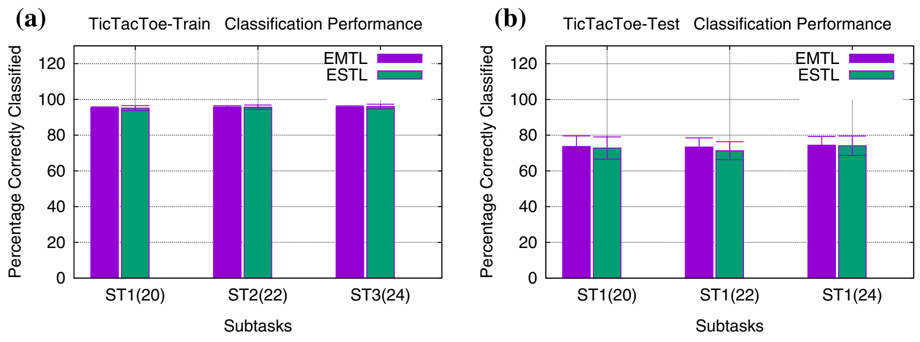

Fig. 8 Performance evaluation of featured modularity using EMTL verses non-modular ESTL across different subtasks [ST1(n), ST2(n), ST3(n)]. **a** Training dataset and **b** test dataset

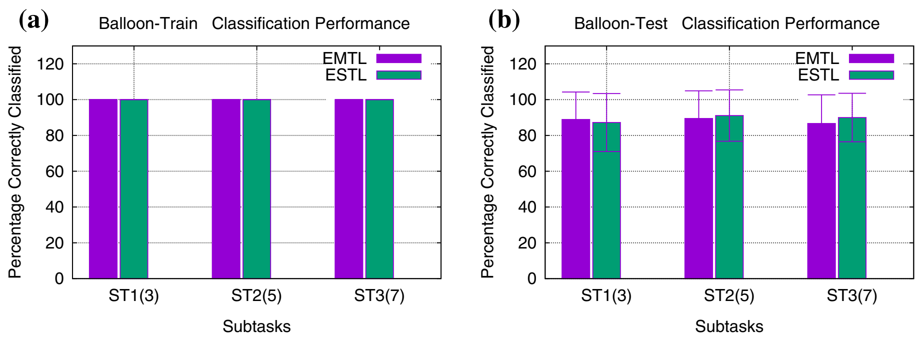

Fig. 9 Performance evaluation of featured modularity using EMTL verses non-modular ESTL across different subtasks [ST1(n), ST2(n), ST3(n)]. **a** Training dataset and **b** test dataset

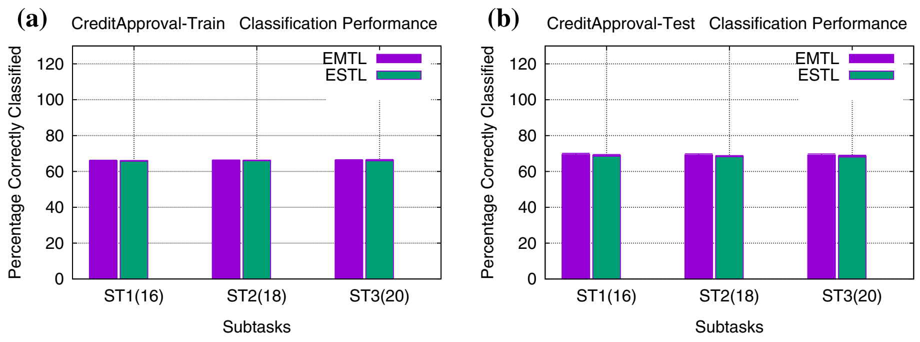

Fig. 10 Performance evaluation of featured modularity using EMTL verses non-modular ESTL across different subtasks [ST1(n), ST2(n), ST3(n)]. **a** Training dataset and **b** test dataset

single task learning approaches the problem sequentially. Therefore, the total time for the respective learning methods is equal when we consider all the subtasks. However, as shown in the results, multi-task learning significantly outperforms the single task approach given an equal computational budget. The same strategy was used for the selected benchmark pattern classification problems.

Table 8 Comparison with related methods

Problem	Dataset	GD	EMTL-ST1	ESTL-ST1
Ionosphere	Train	95.18 ± 1.33	95.86 ± 1.39	96.88 ± 1.68
	Test	83.33 ± 2.53	95.872 ± 2.47	96.33 ± 1.54
Cancer	Train	93.46 ± 0.38	95.52 ± 0.40	95.66 ± 0.48
	Test	96.22 ± 0.64	98.19 ± 0.70	98.27 ± 0.67
Balloon	Train	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Test	90.00 ± 15.27	88.88 ± 15.37	87.22 ± 16.19
Heart	Train	92.49 ± 1.88	82.14 ± 0.79	83.02 ± 1.41
	Test	72.41 ± 3.60	71.85 ± 3.96	73.00 ± 2.34
Tic-Tac-Toe	Train	96.78 ± 0.78	95.31 ± 0.53	95.13 ± 1.37
	Test	63.90 ± 5.16	73.67 ± 5.91	72.81 ± 6.21
Credit	Train	90.43 ± 0.50	66.00 ± 0.25	65.91 ± 0.18
	Test	79.47 ± 1.38	69.50 ± 0.64	68.97 ± 0.47

The results for the benchmark pattern classification problems showed that EMTL with feature of modular knowledge representation retained the performance quality when compared to non-modular method (ESTL). In the case of the n-bit even parity problems, there was a trend that the performance quality improves as additional knowledge modules were used. However, the case of the classification problem is different as n-bit even parity problem is usually considered a special type of classification problem. In principle, the knowledge module which contains the optimal total number of hidden neurons would be giving the best performance. Therefore, in certain cases, additional knowledge modules may deteriorate the performance of the system as they will consume computational budget for training.

Evolutionary multi-task learning helps to retain knowledge within the modules, while at the same time, refine knowledge in additional modules. This can be very helpful when there are not enough neurons or weights to represent additional knowledge in the case when the training data is significantly increased at a later stage. In this case, the fundamental knowledge module will be used to guide the evolution of a solution space in the additional module(s) with help of multi-tasking implementations of evolutionary operators such as selection and crossover [32].

In this way, multi-task learning with modularity does not forget or discard the knowledge learnt in smaller network topologies (modules) which can be useful if a lower number of neurons or connections are needed. This could be required for hardware implementations [10]. Furthermore, it is essential to have a modular trained neural network that is operational with a degree of error for *safe mode operation* when some of the neurons or links in selected modules are damaged during an event. In the case when the additional knowledge modules are damaged, the fundamental knowledge module will be used for safe-mode operation.

Evolutionary multi-task learning delivers a neuro-evolution strategy which employs direct encoding along with the advantages of indirect encoding such as NEAT [30] where the topology is also evolved during evolution. In comparison with the motivation and formulation of NEAT, the proposed modular knowledge representation through EMTL is equipped to remember knowledge through modularity while NEAT only attempts to evolve the network topology in order to escape a local minima. In principle, the respective modules can feature learning algorithms which could reorganise neurons similar to NEAT. The overall modification and feature of EMTL is to use knowledge modules for memory, however, it can also be used to escape a local minima. A local minima could emerge when enough neurons or weights are not available and this could be addressed by additional knowledge modules. Therefore, during evolution, strategies can be implemented that can increase the the number of modules when the algorithm is trapped in a local minima or if given additional training examples that are substantially large.

A limitation of the study is that it has not been explicitly shown that removing certain synapses or perturbing them in the associated modules will deteriorate the classification performance. In the literature, pruning neural networks have shown some advantages for generalization performance. Currently methods that include dropouts have gained attention for deep learning [44,45]. The goal of the paper was to show the effectiveness of modular knowledge representation through evolutionary multi-tasking and only a small class of problems has been considered. The decomposition of the network into modules needs to be generalized further for more hidden layers as we only considered single hidden layer and associated neurons as reference points of modules. Future work could consider recurrent networks and related deep learning architectures.

The study could also motivate further work in studying or simulating biological neural systems. The human brain and learning has features of modularity. It is unclear as to exactly what type of learning happens in biological neural systems when compared to artificial learning algorithms [46]. However, it is clear that there is organisation in the way knowledge is stored and retrieved. In conventional neural networks, there is no modularity enforced to ensure that certain components or the fundamental knowledge is protected. The proposed modular knowledge representation through evolution could motivate further insights to learning and knowledge representation in both biological and artificial neural systems.

5 Conclusions and Future Work

We presented an evolutionary multi-task learning for modular knowledge representation in neural systems which have the property to retain knowledge in some of the modules when others are affected. This can be particularly helpful for hardware implementations where certain defects in the modules do not affect the entire knowledge representation and hence the network can operate in *safe mode*. The motivation behind modularity through evolutionary multi-tasking was to demonstrate that knowledge can be stored in modules without losing classification performance when compared to conventional single-task learning through evolutionary algorithms or gradient based approaches. The experiments were designed to evaluate if evolutionary multi-tasking has been effective in modular storage of knowledge. The results show that the multi-task learning approach was able to provide modular knowledge representation without losing quality in performance for single hidden layer feedforward networks. Moreover, in the case of n-bit party problems, it achieved better solution quality with help of additional modules.

In future work, the approach can be extended to subtasks that consist of different problems rather than modules in neural networks. The approach could motivate simulations for biological neural systems for knowledge and memory representation. There is scope for using other evolutionary techniques such as coevolution with application to problems that have building blocks or require multi-task learning. Moreover, rather than treating the subtask as network modules, the application of the method to subtasks that refer to different problems or groups of features in the datasets can also be explored.

Acknowledgements This work was partially conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

References

1. Bassett DS, Wymbs NF, Porter MA, Mucha PJ, Carlson JM, Grafton ST (2011) Dynamic reconfiguration of human brain networks during learning. Proc Natl Acad Sci 108(18):7641–7646
2. Meunier D, Lambiotte R, Fornito A, Ersche KD, Bullmore ET (2009) Hierarchical modularity in human brain functional networks. Front. Neuroinformatics 3:37. doi:10.3389/neuro.11.037.2009
3. Nicolini C, Bifone A (2016) Modular structure of brain functional networks: breaking the resolution limit by surprise. Sci Rep 6. doi:10.1038/srep19250 (2016)
4. Happel BL, Murre JM (1994) Design and evolution of modular neural network architectures. Neural Netw 7(67):985–1004 (Models of neurodynamics and behavior)
5. Moon SW, Kong SG (2001) Block-based neural networks. IEEE Trans Neural Netw 12(2):307–317
6. San PP, Ling SH, Nguyen H (2014) Evolvable rough-block-based neural network and its biomedical application to hypoglycemia detection system. IEEE Trans Cybern 44(8):1338–1349
7. Nambiar VP, Khalil-Hani M, Sahnoun R, Marsono M (2014) Hardware implementation of evolvable block-based neural networks utilizing a cost efficient sigmoid-like activation function. Neurocomputing 140:228–241
8. Clune J, Mouret JB, Lipson H (2003) The evolutionary origins of modularity. Proc R Soc Lond B Biol Sci 280(1755). doi:10.1098/rspb.2012.2863
9. Ellefsen KO, Mouret JB, Clune J (2015) Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Comput Biol 11(4):1–24
10. Misra J, Saha I (2010) Artificial neural networks in hardware: a survey of two decades of progress. Neurocomputing 74(13):239–255 (Artificial brains)
11. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
12. Angeline P, Saunders G, Pollack J (1994) An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans Neural Netw 5(1):54–65
13. Moriarty DE, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evolut Comput 5(4):373–399
14. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 47(4):547–553
15. Chandra R, Gupta A, Ong YS, Goh CK (2016) Evolutionary multi-task learning for modular training of feedforward neural networks. In: Neural information processing—23rd international conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part II. 37–46
16. Lindbeck A, Snower DJ (2000) Multitask learning and the reorganization of work: from tayloristic to holistic organization. J Labor Econ 18(3):353–376
17. Ando RK, Zhang T (2005) A framework for learning predictive structures from multiple tasks and unlabeled data. J Mach Learn Res 6:1817–1853
18. Jaco L, philippe Vert J, Bach FR (2009) Clustered multi-task learning: a convex formulation. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in neural information processing systems, vol 21. Curran Associates, Inc., Dutchess, pp 745–752
19. Chen J, Tang L, Liu J, Ye J (2009) A convex formulation for learning shared structures from multiple tasks. In: Proceedings of the 26th annual international conference on machine learning. ICML '09, New York, NY, USA, ACM 137–144
20. Chen J, Liu J, Ye J (2012) Learning incoherent sparse and low-rank patterns from multiple tasks. ACM Trans Knowl Discov Data 5(4):22:1–22:31
21. Zhang Y, Yeung DY (2010) Transfer metric learning by learning task relationships. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '10, New York, NY, USA, ACM, pp 1199–1208
22. Bakker B, Heskes T (2003) Task clustering and gating for bayesian multitask learning. J Mach Learn Res 4:83–99
23. Zhong S, Pu J, Jiang YG, Feng R, Xue X (2016) Flexible multi-task learning with latent task grouping. Neurocomputing 189:179–188
24. Yuan H, Paskov I, Paskov H, González AJ, Leslie CS (2016) Multitask learning improves prediction of cancer drug sensitivity. Sci. Rep 6. doi:10.1038/srep31619
25. Sexton RS, Dorsey RE (2000) Reliable classification using neural networks: a genetic algorithm and backpropagation comparison. Decis Support Syst 30(1):11–22

26. Cant-Paz E, Kamath C (2005) An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. IEEE Trans Syst Man Cybern B Cybern 35(5):915–933
27. Garcia-Pedrajas N, Hervas-Martinez C, Munoz-Perez J (2003) COVNET: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Trans Neural Netw 14(3):575–596
28. Gomez F, Schmidhuber J, Miikkulainen R (2008) Accelerated neural evolution through cooperatively coevolved synapses. J Mach Learn Res 9:937–965
29. Chandra R (2015) Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction. IEEE Trans Neural Netw Learn Syst 26:3123–3136
30. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evolut Comput 10(2):99–127
31. Heidrich-Meisner V, Igel C (2009) Neuroevolution strategies for episodic reinforcement learning. J Algorithms 64(4):152–168 (Special issue: reinforcement learning)
32. Gupta A, Ong YS, Feng L (2016) Multifactorial evolution: toward evolutionary multitasking. IEEE Trans Evolut Comput 20(3):343–357
33. Gupta A, Ong YS, Feng L, Tan KC (2016) Multiobjective multifactorial optimization in evolutionary multitasking. IEEE Trans Cybern (accepted)
34. Ong YS, Gupta A (2016) Evolutionary multitasking: a computer science view of cognitive multitasking. Cognit Comput 8(2):125–142
35. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
36. Chen X, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evolut Comput 15(5):591–607
37. Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Syst 9(2):115–148
38. Deb K, Deb D (2014) Analysing mutation schemes for real-parameter genetic algorithms. Int J Artif Intelli Soft Comput 4(1):1–28
39. Liu D, Hohil ME, Smith SH (2002) N-bit parity neural networks: new solutions based on linear programming. Neurocomputing 48(14):477–488
40. Mangal M, Singh MP (2007) Analysis of pattern classification for the multidimensional parity-bit-checking problem with hybrid evolutionary feed-forward neural network. In: Advances in computational intelligence and learning 14th European symposium on artificial neural networks 2006. Neurocomputing 70(79):1511–1524
41. Mirjalili S, Hashim SZM, Sardroudi HM (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137
42. Chandra R, Frean MR, Zhang M (2012) Crossover-based local search in cooperative co-evolutionary feedforward neural networks. Appl Soft Comput 12(9):2924–2932
43. Asuncion A, Newman D (2007) UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets.html
44. Reed R (1993) Pruning algorithms—a survey. IEEE Trans Neural Netw 4(5):740–747
45. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
46. Meltzoff AN, Kuhl PK, Movellan J, Sejnowski TJ (2009) Foundations for a new science of learning. Science 325(5938):284–288