Neurocomputing 243 (2017) 21–34

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction

Rohitash Chandra*, Yew-Soon Ong, Chi-Keong Goh

Rolls Royce @NTU Corporate Lab, Nanyang Technological University, 65 Nanyang View, Singapore

ARTICLE INFO

Article history:
Received 2 September 2016
Revised 6 January 2017
Accepted 22 February 2017
Available online 2 March 2017

Communicated by Zhou Xiuzhuang

Keywords:
Cooperative coevolution
Neuro-evolution
Feedforward networks
Multi-task learning
Multi-step time series prediction

ABSTRACT

Multi-task learning employs a shared representation of knowledge for learning several instances of the same problem. Multi-step time series problem is one of the most challenging problems for machine learning methods. The performance of a prediction model face challenges for higher prediction horizons due to the accumulation of errors. Cooperative coevolution employs in a divide and conquer approach for training neural networks and has been very promising for single step ahead time series prediction. Recently, co-evolutionary multi-task learning has been proposed for dynamic time series prediction. In this paper, we adapt co-evolutionary multi-task learning for multi-step prediction where predictive recurrence is developed to feature knowledge from previous states for future prediction horizon. The goal of the paper is to present a network architecture with predictive recurrence which is capable of multi-step prediction through a form of multi-task learning. We employ cooperative neuro-evolution and an evolutionary algorithm as baselines for comparison. The results show that the proposed method provides the best generalization performance in most cases. Comparison of results with the literature has shown to be promising which motivates further application of the approach for related real-world problems.

1. Introduction

Multi-step-ahead prediction refers to the forecasting or prediction of a sequence of future values from an observed trend in a time series [1,2]. The prediction horizon defines the extent of future prediction. The chaotic nature and noise in real-world time series makes it challenging to develop models that produce low prediction error as the prediction horizon increases [3–5]. Multi-step-ahead prediction has been approached mostly with the *recursive* and *direct* strategies. In the recursive strategy, the prediction from a one-step-ahead prediction model is used as input for future prediction horizon [6,7]. In this approach, any uncertainty or error in the prediction for the next horizon is accumulated in future horizons. The direct strategy considers the multi-step-ahead problem as a multi-output prediction problem [8,9] which can also be explored through multi-task learning.

Multi-task learning considers an appropriate sharing mechanism of common knowledge across tasks in order to enable knowledge from one task to benefit other(s) [10]. Multi-task learning is considered as a type of inductive transfer that employs the domain information contained in the training signals of related tasks as an inductive bias [11–14]. This is typically achieved by learning tasks in parallel through a shared knowledge representation. In real-world applications [15–20], the appropriate way to exploit structures among multiple learning tasks can be challenging as learning tasks consist of several disjoint or partially overlapped task groups with some outlier tasks. Negative transfer of knowledge refers to the case when related and unrelated tasks are combined that leads to the decrease the learning performance [21].

Neuro-evolution refers to the use of evolutionary algorithms for training neural networks [22]. Cooperative coevolution (CC) [23] employs divide and conquer approach to decompose the problem into subcomponents. It has been very promising method for neuro-evolution [24–26] and referred as *cooperative neuro-evolution* [27]. Coevolution has shown features more diverse solutions through the sub-populations when compared to conventional evolutionary algorithms [24] and also ensures modularity. It has also been very promising for time series prediction [28,29]. Modular neural networks are motivated from repeating structures in nature [30] and were introduced for visual recognition tasks that were trained by genetic algorithms [30]. More recently, a modular neural network was presented where the performance and connection costs were optimized through neuro-evolution which achieved better performance when compared to fully connected neural network [31] in order to learn new tasks without forgetting old ones [32].

* Corresponding author. +65 6592 3778
E-mail address: c.rohitash@gmail.com (R. Chandra).

http://dx.doi.org/10.1016/j.neucom.2017.02.065
0925-2312/

Multi-tasking evolutionary algorithms have been proposed for optimization problems by enabling transfer of knowledge between tasks that consist of different types or instances of optimization problems with some overlapping features taken from same domain [33,34]. Recently, evolutionary multi-tasking has been used for training feedforward neural networks for n-bit parity problem [35]. The different tasks were implemented as different topologies given by the number of hidden neurons resulting in improved training performance. This provides the motivation to apply neuro-evolution in multi-task learning for time series prediction.

In a dynamic programming (optimization), a large problem is broken down into sub-problems that used as building blocks to solve the bigger problem. Dynamic programming has mainly been used for optimization problems. There has not been much work done for incorporating the approach into machine learning problems that requires knowledge from previous states as building blocks for further decision making [36,37]. Multi-step time series prediction is an application where a synergy between dynamic programming and multi-task learning can be developed. Multi-task learning exploits the idea of shared knowledge representation that can be used as the building block of knowledge for bigger tasks. Recently, co-evolutionary multi-task learning has been proposed for dynamic time series prediction which was identified as a special class of problems that are encountered in adverse environments where dynamic or rapid prediction is required [38].

The underlying sub-problem in the case of multi-step-ahead prediction represents the initial steps of the prediction horizon. In multi-step ahead prediction, the prediction for future horizons depends on the knowledge of the previous prediction horizon. This can be seen as an implicit form of shared knowledge representation across different prediction horizon. Therefore, it is reasonable to use multi-task learning for multi-step ahead prediction where the different prediction horizon can be seen as different but related tasks.

In this paper, we adapt co-evolutionary multi-task learning with predictive recurrence from previous state prediction for future prediction horizon. In the proposed method, neural networks are trained using a novel coevolution approach that retains modularity in network topology for prediction of future prediction horizon(s). In this way, the neural network is able to preserve knowledge for different tasks while each task knowledge is used as a building block for future tasks that are defined by the prediction horizon. The goal of the paper is to present a network architecture with predictive recurrence which is capable of multi-step prediction through features of multi-task learning. We apply the proposed method on 4 simulated and 3 real-world benchmark chaotic time series problems with instances of 5 and 10 step ahead prediction. We employ cooperative neuro-evolution and an evolutionary algorithm as baselines for comparison.

The rest of the paper is organized as follows. Section 2 gives background and related work on multi-step-ahead time series prediction, multi-task learning and cooperative neuro-evolution. Section 3 gives details of co-evolutionary multi-task learning for multi-step-ahead time series prediction. Sections 4 and 5 present the results and discussion, respectively. Section 6 concludes the paper with directions for future research.

2. Background and related work

2.1. Multi-step ahead time series prediction

As discussed earlier, recursive and direct strategies are the major ways to implement the multi-step-ahead time series (MSA) prediction. Although relatively new, a third strategy is a combination of these approaches [6,39].

We first consider some of the methods in the recursive (also known as iterated) strategy. Ng and Young presented the initial work for formulation for recursive estimation and MSA prediction [40]. Su et al. gave one of the earliest attempts in for MSA prediction using recurrent neural networks [41]. Parlos et al. presented a dynamic recurrent network where current and delayed observations of the measured system input and output were utilized as inputs to the network and reported excellent generalization performance [42]. Girard et al. presented iterated MSA prediction using the non-parametric Gaussian process model and showed how one can formally incorporate the uncertainty about intermediate regressor values [43]. Niu and Yang presented an iterative MSA method where they employed the *DempsterShafer* regression technique for prognosis of data-driven machinery [44] with promising performance. More recently, Chang et al. presented reinforced real-time recurrent learning (RTRL) with recurrent neural networks for two-step ahead prediction with improved accuracy of flood forecasts and effectively reducing time-lag effects [4].

In the case of methods that fall under the direct strategy, Bone and Crucianu presented variations in the back-propagation through-time algorithm (BPTT) for training recurrent neural networks for MSA prediction [5] and achieved promising results. Taieb et al. presented a review of single-output vs. multiple-output approaches for MSA prediction and reported that direct strategy had shown to be promising over recursive strategy [9]. Bao et al. presented [45] multiple-output support vector regression (M-SVR) with multiple-input multiple-output prediction strategy that achieved the better forecasts with accredited computational load when compared to the standard SVR using direct and iterated strategies.

Considering the combination strategy, Zhang et al. presented multiple support vector regression models (SVR) that were trained independently based on the same training data and with different targets. The approach could be viewed as a combination of the iterative and direct method which showed promising results [6]. Moving on, an optimally pruned extreme learning machine (OP-ELM) was used using recursive, direct and a combination of the two strategies in an ensemble approach. The results showed that the combination gave better performance than direct and iterative methods used with linear models and competitive with support vector machine [39]

Taieb et al. presented a comprehensive study on the different strategies using a large experimental benchmark (NN5 forecasting competition) [46]. They considered the effects of deseasonalization, input variable selection, and forecast combination on these strategies. It was reported that direct strategy given by multiple-output perform the best, deseasonalization leads to uniformly improved forecast accuracy, and input selection is more effective when performed in conjunction with deseasonalization. Further comparison for the direct and iterative method was done for MSA macroeconomic time series where it was reported that the iterated forecasts typically outperformed the direct forecasts [47]. It was also shown that the relative performance of the iterated forecasts improved with the forecast horizon. A comparison on comparison of the direct and iterated predictors presented an encompassing representation that enabled the derivation the auto-regressive coefficients [48]. Chevillon presented a study on the properties of iterated and direct MSA strategies in the presence of in-sample location shifts defined by *breaks* in the mean using autoregressive models [49]. It was shown that direct strategy provides prediction values that are relatively robust to breaks, and its benefits increases with the prediction horizon.

There exists major scope for MSA prediction in real-world problems. Auto-regressive models have proposed with promising results for predicting critical levels of abnormality in physiological signals [50]. Other applications include flood forecasting using

recurrent neural networks [51,52], emissions of nitrogen oxides using a neural network and related approaches [53], and photovoltaic power forecasting using hybrid support vector machine [54]. Yang et al. presented multi-step-ahead prediction of strong earthquake ground motions and seismic responses based on empirical mode decomposition-extreme learning machine method with promising results [55].

Challenges in MSA prediction include the prediction for highly chaotic time series and those that have missing data. Wu et al. approached the missing data problem in time series with non-linear filters and neural networks [56]. In their method, a sequence of independent Bernoulli random variables were used to model random interruptions which was later used to construct the state-space vector in the preprocessing stage.

2.2. Multi-task Learning and applications

Multi-task learning employs a shared representation of knowledge for learning several different instance of the same or related problems [10]. A number of approaches have been presented that considers multi-task learning for different types of problems that include supervised and unsupervised learning. Ando et al. presented a general framework for multiple tasks and unlabelled data in which the structural learning problem was formulated and analysed theoretically for semi-supervised learning [11]. Jacob and Bach presented multi-task learning method which assumed that the tasks are clustered into groups and the tasks within a group have similar weight vectors. They presented a convex optimization formulation for multi-task learning and showed that the approach outperforms well-known convex and related non-convex methods for multi-task learning [12]. Chen et al. presented an improved formulation for multi-task learning based on the non-convex alternating structure optimization algorithm where all tasks are related by a shared feature representation [13]. They presented a theoretical condition through which the proposed approach could find globally optimal solution. The experiments on several benchmark datasets validated their theoretical analysis. Zhou et al. presented a clustered multi-task learning that incorporated alternating structure optimization that simultaneously performed inference on multiple tasks [14]. Chen et al. presented an approach based on a linear multi-task learning formulation with gradient algorithms for learning incoherent sparse and low-rank patterns from multiple tasks which were efficient for real-world datasets [57].

Negative transfer has been a major challenge for multi-task learning. The major approach to address it has been through task grouping where knowledge transfer is performed only within a group [58,59]. Bakker et al. 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 [59]. Zhang and Yeung presented a convex formulation for multi-task metric learning by modelling the task relationships in the form of a task covariance matrix [58]. Moreover, Zhong et al. presented flexible multi-task learning framework to identify latent grouping structures in order to restrict negative knowledge transfer [60].

Multi-task learning has recently contributed to a number of successful real-world applications that gained better performance by exploiting shared knowledge for multi-task formulation. Tang et al. presented multi-task approach for “retweet” prediction behaviour of individual users [15]. Zhang and Mahoor presented a multi-task learning approach for recognition of facial action units. In their approach, learning the discriminative hyperplanes for detecting each action unit was viewed as a task. The relations among multiple action units were modelled based on the commonalities among the functions of hyperplanes [16]. Liu et al. presented an approach automated Human Epithelial Type 2 (HEp-2) cell classification where models of multiple cell categories were jointly trained in the framework of clustered multi-task learning [17]. Qin et al. presented an approach for kin-relationship verification using visual features. Their method shared feature sets and useful structures to decompose each single task model into two parts where one is shared amongst all the tasks and the other corresponds to the task-specific structure [18]. Finally, Zhang et al. approached object tracking with multi-task learning where the tracking task is decomposed into several local tasks and the final tracking result is obtained by combining them [19]. Cheng et al. presented an object tracking approach where the discriminative tracker considers the object appearance while the generative tracker takes the local information of an object into account for handling partial occlusions using a multi-task formulation [20].

Xu et al. applied multi-task regression for forecasting problem [61] which provided a synergy of ensemble forecasting [62]with multi-task regression to estimate the optimal weights for combining the ensemble members. The weights were updated using a novel online learning with restart strategy for new observation data. The results were very promising when compared to ensemble and baseline methods for seasonal soil moisture predictions from 12 major river basins. We note that there has not been much work that employs the idea of multi-task learning for multi-step-ahead prediction from our review of the literature. Hence, the proposed method in this paper will fulfil this gap as multi-task learning naturally appeals to multi-step prediction if we consider the prediction horizon as tasks that have shared knowledge.

2.3. Cooperative neuro-evolution

Cooperative coevolution was originally proposed by Potter and Jong for global optimization [23] which later drew attention for large scale global optimization with challenges in problem decomposition and adaptation [63,64]. They later employed it for training neural networks and reported properties that showed improved modularity and diversity [24] when compared to related methods. The challenges in cooperative neuro-evolution have been in area of credit assignment for subcomponents [24,65], selection pressure for the sub-populations, problem decomposition and adaptation based on the type of the problem in terms of separability [27].

Cooperative neuro-evolution has been applied with success to pattern classification problems. Garcia-Pedrajas et al. employed cooperative neuro-evolution for pattern classification problems which gave better generalization and produced smaller networks when compared to an adaptive mixture of experts [66]. Further work was done for using cooperative neuro-evolution for generalized multilayer perception to enable a more compact representation with improved performance [67]. A multi-objective approach has been incorporated in cooperative neuro-evolution to address the problem of assigning credit to the subcomponents which showed improved performance in classification problems [65]. Zhu and Guan incorporated concurrent local and global evolution methods in cooperative neuro-evolution to further improve the classification performance [68].

Problem decomposition has been a major challenge in cooperative neuro-evolution. In previous studies, two major methods have been proposed that have different strengths and weaknesses based on the type of application that include control, times series prediction and pattern classification. Gomez et al. presented problem decomposition according to the lowest level of granularity where each synapse forms a subcomponent [69] for double pole balancing control problems. Lin et al. applied this decomposition for neural fuzzy network with cultural cooperative particle swarm optimization for chaotic time series problem [70]. Chandra et al. showed that synapse level problem decomposition faced problems for pattern classification problems [27]. They earlier proposed

neural level decomposition, where the neurons in the network act as the reference point for the subcomponent for training recurrent neural networks on grammatical inference problems [71]. This was based on a similar concept as enforced sub-populations previously proposed by Gomez and Mikkulainen [26], however, neuron level decomposition produced better results for pattern classification [27]. Chandra et al. applied neural and synapse level decomposition for chaotic time series problems using recurrent neural networks [28]. Hence, it was established that synapse level encoding was good for time series and control problems [28,69].

Chandra et al. further focused on adaptation of problem decomposition in different phases of evolution [72]. They reported that adaptation at different stages of evolution is difficult due to need of parameter turning that required knowledge on when to adapt problem decomposition during evolution. This motivated a competitive and collaborative method that eliminated the need for parameter tuning required to determine when to adapt decomposition which showed very promising performance for chaotic time series problems [29].

Although previous works have tried to address problem decomposition, there are major challenges when new types of problems are considered. In the proposed method for incorporating multi-task learning for multi-step prediction, the training algorithms are based on co-evolution. The decomposition of the overall problem is done by having a sub-population for the different network topologies that correspond to the respective prediction horizon. This will be given in detail hereafter.

3. Co-evolutionary multi-task learning for multi-step-ahead prediction

3.1. Data reconstruction

In state-space reconstruction, the original time series data is divided using overlapping windows at regular intervals that can be used for one-step-ahead and MSA prediction. Taken's theorem expresses that the vector series reproduces many important characteristics of the original time series [73]. Hence, given an observed time series $x(t)$, an embedded phase space $Y(t) = [(x(t), x(t - T), \ldots, x(t - (D - 1)T)]$ can be generated, where, T is the time delay, D is the embedding dimension (window), $t = 0, 1, 2, \ldots, N - DT - 1$ and N is the length of the original time series. The optimal values for D and T must be chosen in order to efficiently apply Taken's theorem [74]. Taken's proved that if the original attractor is of dimension d, then $D = 2d + 1$ will be sufficient to reconstruct the attractor [73]. In the case of using feedforward networks, D is the number of input neurons. State-space reconstruction can also be extended to multivariate time series. A recent study has shown that multivariate data would perform better than univariate data for MSA prediction. This is because as the prediction horizon becomes larger, multi-variate information becomes more important [75].

3.2. Proposed method

The proposed method is inspired by the strategies used in dynamic programming where a subset of the solution is used as the main building block for the optimization problem. In this case, the overall problem is learning the weights of a neural network and the base problem is the neural network for MSA prediction with the smallest architecture that represents the first prediction horizon. The knowledge represented in the network from the first prediction horizon becomes the shared knowledge for the remaining prediction horizons. As the prediction horizon increases, the number of hidden neurons increases which gives room for storing more knowledge for additional prediction horizons which utilize the base network knowledge. We name the approach as *co-evolutionary multi-task learning* (CMTL).

Algorithm 1: Co-evolutionary multi-task learning.

Data: Requires prediction data for different tasks (predictionhorizon) $D_{(task)}$
Result: Weights as model parameters for FNN $Network_{task}$ initialization;
for *each task* **do**
  1. Each task employs the same input data $D_{(task)}$ that corresponds to neural network $Network_{task}$. The number of input and output neurons are constant while the number of hidden neurons change for each prediction horizon (task): (input i, hidden j and output k);
  2. Define the weight space for the different overlapping tasks $W_{(task)}$;
  3. Initialize individuals of the sub-populations S_{task}, within the unified search space;
  4. Assign arbitrary fitness values for fitness F_i of individuals in each sub-population S_{task}.
  5. Assign depth of search, e.g. $depth = 5$ that defines the number of generation for each sub-population S_{task}.
end
while *each phase until termination* **do**
  for *each task* **do**
    for *each generation until depth* **do**
      - Get best Solution Sol from S_{task}
      if $task == 1$ **then**
        - Get the current solution and assign $TaskSol_{(task)} = Sol_1$
      end
      else
        - Append the current task solution Sol_{task} with best solutionsfrom previous tasks (prediction horizon), $TaskSol_{(task)} = [Sol_1, Sol_2, ..Sol_{task}]$
      end
      for *each Individual j in S* **do**
        - Call Algorithm 2: This will encode the $TaskSol_{(task)}$ intothe $Network_{task}$- Load data $D_{(task)}$ for the task and evaluate the $Network_{task}$ forfitness F given by *RMSE*
      end
      for *each Individual j in S* **do**
        - Select and create new offspring via evolutionary operators suchselection, crossover and mutation
      end
      - Update S;
      Update number of FE;
    end
  end
end
- Test the obtained solution
for *each task* **do**
  1. Load best solution S_{best} from S_{task}.;
  2. Map into theweight space for the task W_{task};
  3. Test the $Network_{task}$ using test data for $TestData_{task}$ and ;
  4. Report *RMSE* for each prediction horizon
end

CMTL is used for training feedforward neural networks (FNNs) for MSA prediction. It considers different tasks as neural network topologies defined by different number of hidden neurons which refer to the different tasks that represents the respective

prediction horizon. Each sub-population for coevolution is given as $S_1, S_2, \ldots, S_N$, where N is number of sub-populations. The sub-populations consist of a matrix of variables that refer to the weights of the FNN that correspond to the different tasks $S = X_{(i,j)}$; where i is the number of variables and j is the number of individuals in the respective sub-population. $S_{(task)}$ corresponds to a specific task where $task = 1, 2, \ldots, N$ which corresponds to network defined by the task ($Network_{(task)}$) with data for each task which remains the same ($D_{(task)}$). Hence, each task as the same input vector, but a different output which is defined by the different prediction horizon.

CMTL can be seen as a combination strategy for MSA prediction. The problem formulation represents the iterated MSA prediction approach through multi-task learning. Hence, the number of output neurons is 1 for each of the tasks as shown in Fig. 1.

Suppose that $W_{(task)}$ is the set of input to hidden layer and hidden to output layer weights. Therefore, we have the weights of a neural network topology for each task appended with the rest of the task knowledge that represents the respective prediction horizon x, as shown in Eq. (1).

$$x_{(task)} = \left[W_{(task)},\ W_{(task-1)}, W_{(task-2)}, \ldots, W_{(task=1)}\right] \tag{1}$$

The details of the execution in CMTL is given by Algorithm 1 that begins by initializing all the components. These include the sub-population (S_{task}) co-evolution and the network topologies defined by the tasks ($Network_{task}$) which feature the weights (W_{task}) that are shared amongst the respective tasks. The input data ($Data_{task}$) is same for all the respective tasks. The sub-populations (S_{task}) are initialized with real values in a range ($[-a, a]$). The fitness (F) vector in each of the sub-populations for the respective tasks is assigned arbitrary fitness values at the beginning of evolution (learning). This is required as the evolution begins without evaluation of the respective sub-population solutions at the initialization stage.

The algorithm moves into the evolution phase after initialization where each task is evolved for a fixed number for generations defined by the depth of search ($depth$).

The algorithm evaluates the condition; if $task == 1$, then the task solution ($TaskSol_{task}$) returns the best solution (Sol_{best}) from the sub-population (S_{task}). Otherwise, the current task solution (Sol_{task}) is appended with best solutions from previous tasks ($TaskSol_{task} = [Sol_1, Sol_2, \ldots, Sol_{task}]$). Next, the task solution obtained is given as a parameter to Algorithm 2 along with the network topology $Network_{task}$ of all the tasks in order to decode the task solution into the respective weights of the network. This procedure is done for every individual (i) in the sub-population of the task (S_{task}). This procedure is repeated for every task for different phases until the termination condition is satisfied. The termination condition can be either the maximum number of function evaluations or a minimum fitness value from the training or validation dataset. Fig. 1 shows the neural network decomposition associated with the respective tasks given by their prediction horizon.

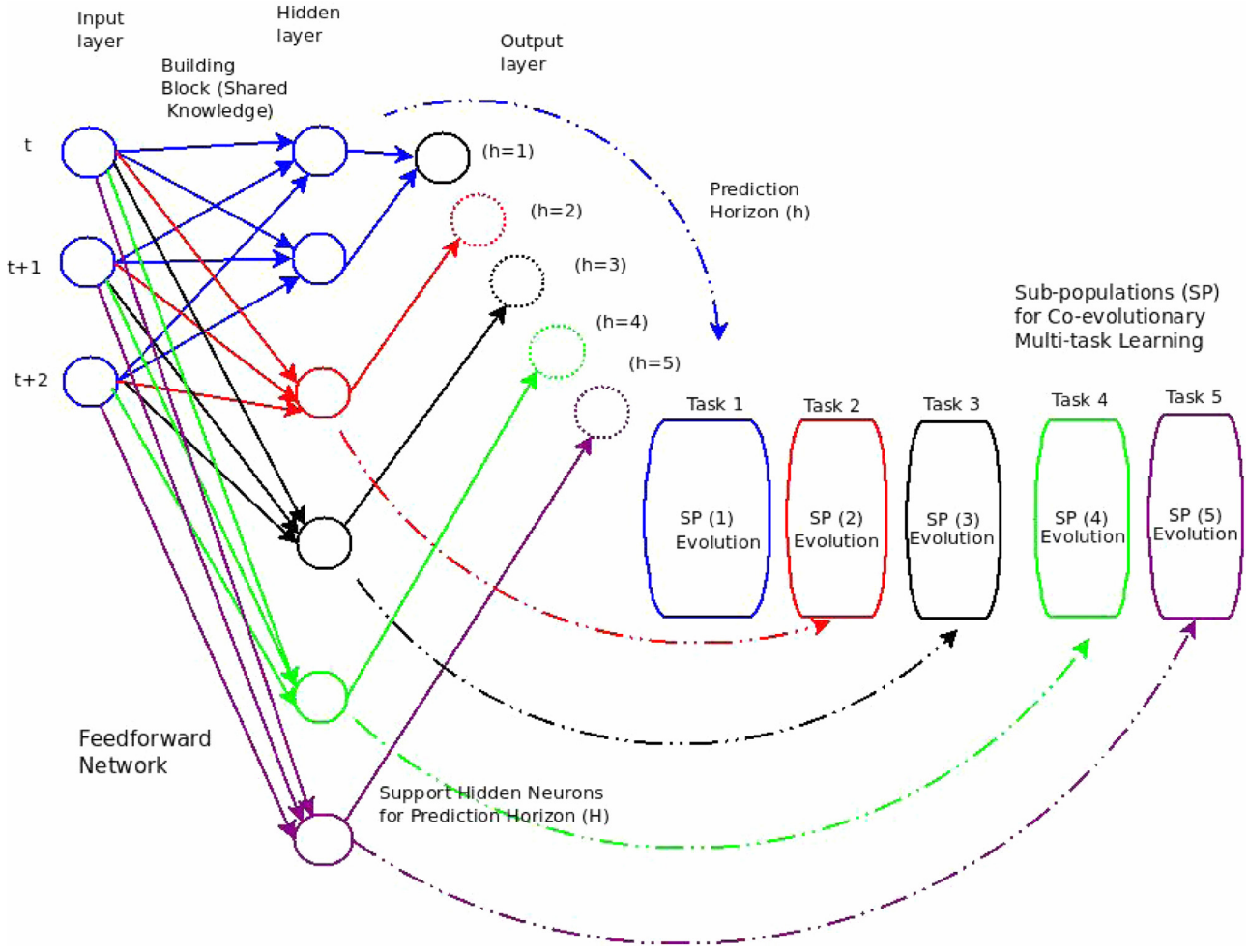


Fig. 1. Tasks highlighted for prediction horizon and sub-populations in co-evolutionary multi-task learning with predictive recurrence. Note that the synapses given by different colours in the network are linked to their sub-populations as different tasks. Task 1 employs a network topology with 2 hidden neurons while the rest of the tasks add extra hidden neurons only.

Algorithm 2: Transfer of knowledge from previous tasks.

Input Parameters: *Task*, *TaskSol* , *Input*, *Hidden* and *Output*
$BaseTask = 1$
Step 1:
for *each* $j = 1$ *to* $Hidden_{(BaseTask)}$ **do**
  for *each* $i = 1$ *to* $Input_{(BaseTask)}$ **do**
    $W_{(i,j)} = TaskSol(t)$;
    $t = t + 1$;
  end
end
Step 2:
for *each* $k = 1$ *to* $Output_{(BaseTask)}$ **do**
  for *each* $j = 1$ *to* $Hidden_{(BaseTask)}$ **do**
    $W_{(j,k)} = TaskSol(t)$;
    $t = t + 1$;
  end
end
if $task >= 2$ **then**
  Step 3
  for *each* $j = Hidden_{(task\text{-}1)} + 1$ *to* $Hidden_{(task)}$ **do**
    for *each* $i = 1$ *to* $Input_{(task)}$ **do**
      $W_{(i,j)} = TaskSol(t)$;
      $t = t + 1$;
    end
  end
  Step 4
  for *each* $k = 1$ *to* $Output_{(task)}$ **do**
    for *each* $j = Hidden_{(task\text{-}1)} + 1$ *to* $Hidden_{(task)}$ **do**
      $W_{(j,k)} = TaskSol(t)$;
      $t = t + 1$;
    end
  end
end

Fig. 2 shows the way the task solution is decomposed and mapped into the network.

The major way CMTL differs from cooperative neuro-evolution (CNE) is from the differences in problem decomposition and fitness evaluation. In CMTL, the fitness of an individual from a sub-population (S_{task}) depends on the previous tasks if the task is greater than 1. This is different for the case of CNE as the fitness of an individual is calculated when it is concatenated with the best individuals from all the respective sub-populations. This is a major difference in the approach which makes CMTL useful for multiple tasks that represent the prediction horizon with one output neuron only.

Fig. 1 only shows addition of one extra hidden neuron for each prediction horizon. Note that the output of the network is one hidden neuron which recursively predicts future prediction horizon based on previous building blocks of knowledge represented by the synapses of previous tasks. Fig. 2 gives further details of how the knowledge is transferred. Note that the respective prediction horizon (h) uses the same input features. The number of hidden neurons is unique for the prediction horizon and hence it serves as the main identity, i.e. for the prediction of horizon ($h = 2$), the network will load the input and specify the exact network topology (given by number of hidden neurons) in order to get the prediction for the respective horizon.

Fig. 1 attempts to highlight the shared knowledge representation for multi-task learning where the smallest neural network topology is used as the foundational building block. The additional hidden neurons that corresponds to different prediction horizons share the knowledge from the foundational building block that is later used as additional knowledge blocks for the respective future prediction horizon(s). Although input data and number of neurons is the same, there are different hidden neurons used for the respective prediction horizon. This can be seen as implicit transfer of knowledge for the set of prediction horizon.

Finally, when a termination criterion has been met, the algorithm moves into the testing phase where the best solutions from all the different tasks are used. Once this is done, the respective task test data is loaded and the network is used to make a decision that results in a certain measure of error. This can be given by the root-mean-squared-error (RMSE) for each prediction horizon (task), however, any other measure can also be implemented. The Matlab code for implementation of Algorithm 1 is given online.[1]

3.3. Transfer of knowledge from tasks

The challenging aspect of Algorithm 1 is the transfer of knowledge by the different tasks in CMTL while enabling a shared knowledge representation. The purpose of Algorithm 2 is to transfer neural network weights that are mapped from different sub-populations defined by the tasks. Therefore, it is used for transfer for different number of tasks. The algorithm is passed parameters as follows.

1. The current task ($task = 1, 2, \ldots, N$), where, N is the number of tasks and each task corresponds to the respective sub-population;
2. The current task solution ($TaskSol_{task} = [Sol_1, Sol_2, \ldots, Sol_N]$). The solutions are appended with solutions of previous tasks in cases when $task > 1$;
3. The topology of different tasks in terms of number of hidden neurons.

We only use tasks given by three prediction horizons to demonstrate the transfer of knowledge which is generalized for any number of prediction horizon. We describe the algorithm with reference to Fig. 2 which shows a case where the network for three prediction horizons ($task = 3$) transfers its knowledge from one task to another. Here, $task = 1$ and $task = 2$ are used as building blocks of knowledge given in the weights. Therefore, we use examples for the network topology as highlighted below.

1. *Input* is vector of number of input neurons for the respective tasks, e.g. $Input = [3, 3, 3]$;
2. *Hidden* is vector of number of input neurons for the respective tasks, e.g. $Hidden = [2, 3, 4]$;
3. *Output*: is vector of output neurons for the respective tasks, e.g. $Output = [1, 1, 1]$. Note that since our application is limited for one step ahead time series prediction, we only consider 1 output neuron for all the tasks.

The algorithm begins by assigning $BaseTask = 1$ that refers to the underlying shared knowledge for all the tasks. In Step 1, the transfer for Input-Hidden layer weights from the *TaskSol* is executed by weights (1–6) in Fig. 2. Step 2 executes the transfer for Hidden-Output layer weights from the *TaskSol* as shown by weights (7–8) in Fig. 2.

The algorithm terminates if $task = 1$ or proceeds if $task >= 2$. In Step 3, the situation is more complex as we consider $task >= 2$. In this case, Step 1 and 2 are executed before moving to Step 3 where *TaskSol* contains the appended solution sets from previous

[1] https://github.com/rohitash-chandra/CMTL-PRNN.

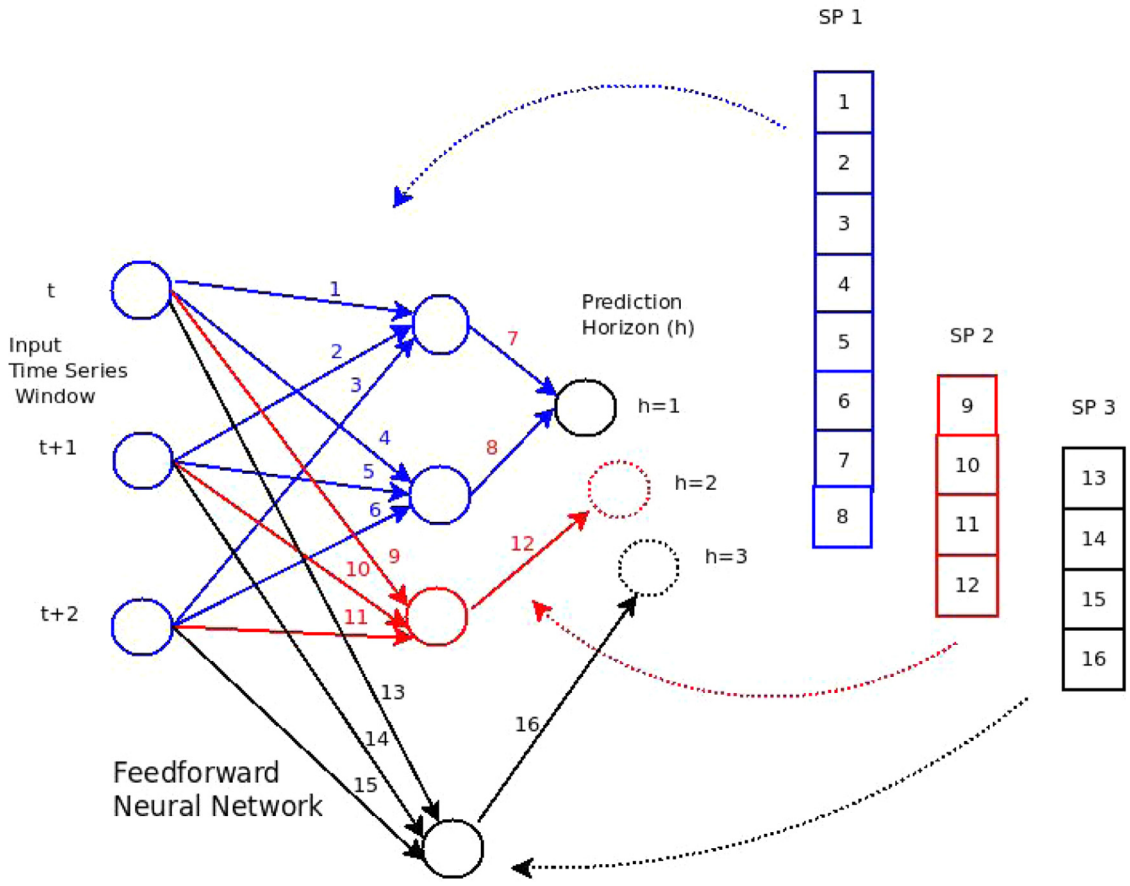


Fig. 2. Knowledge transfer from tasks for different prediction horizon (h). Note that Task 2 utilizes the knowledge of Task 1. The same concept is used for Task 3 which utilizes knowledge of the previous tasks.

task, $TaskSol_{(2)} = [Sol_{task=1}, Sol_{task=2}]$. In this case, we consider the position t of $TaskSol(t)$ for Step 3; t in principle points to the beginning of the solution given by sub-population for $task = 2$. The transfer for Input-Hidden layer weights (9–11) is executed from the $TaskSol$ for $Task = 2$, which is marked by position t that increments during the transfer of knowledge. In the case of $task = 3$, the weights (13–15) would be transferred.

Finally, in Step 4, the algorithm executes the transfer for Hidden-Output layer weights. In case of $task = 2$, this results in transferring weight (12) and for $task = 3$, the transfer is weight (16) in Fig. 2, respectively. The algorithm can therefore transfer knowledge given by variations of hidden neurons as the prediction horizon increases.

4. Simulation and analysis

This section presents an experimental study that compares the performance the multi-task learning proposed by CMTL with single-task learning (conventional) methods such as cooperative neuro-evolution (CNE) and evolutionary algorithm (EA). The covariance matrix adaptation evolution strategies (CMAES) [76] is used as the designated evolutionary algorithm in the multi-task learning and single-task learning approaches. We test the respective methods on MSA prediction with instances of 5-steps and 10-step problems, respectively.

4.1. Benchmark chaotic time series problems

We employ four simulated and three real-world benchmark time series. The simulated time series are Mackey and Glass [77], Lorenz [78], Henon [79], and Rossler [80]. The real-world time series are Sunspot [81], Lazer [82] and ACI-financial time series [83]. All the respective time series data sets are scaled in the range [0,1] in order to be used for sigmoid units in the feedforward neural network.

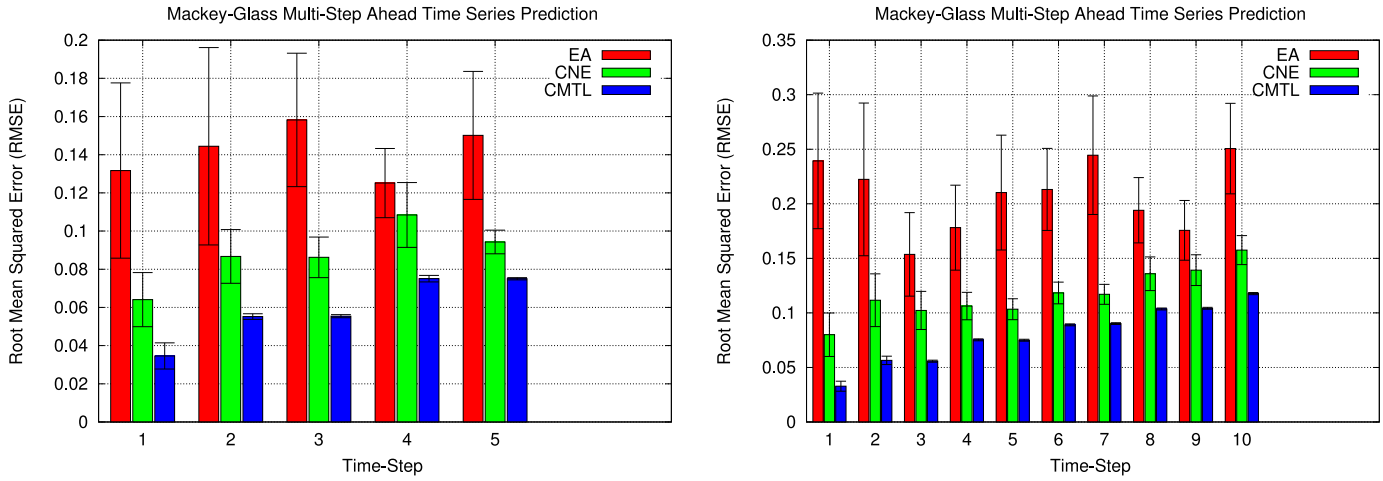
We use 1000 data points in each of the simulated time series where the first half is used for training and the second half is used for testing.

In the case of the real-world time series, the data is used as follows. The Sunspot time series indicates solar activities which impacts Earth's climate is selected from November 1834 to June 2001 and consists of 2000 data points [81]. The ACI-finance time series contains closing stock prices from December 2006 to February 2010, which is equivalent to approximately 800 data points [83]. The Lazer time series was obtained from a physics laboratory experiment and was used in the *Santa Fe competition* that consists of 500 points [82]. In the respective problems, the first half of the data points are used for training and the remaining for testing.

4.2. Experimental design

The neural network for single and multi-task learning employed sigmoid units in the hidden and output layer for all the time series problems. CMTL employs one neuron in the output layer for each prediction horizon of the respective prediction problem (5-step and 10-step) as shown in Fig. 1. The topology can be viewed as a multi-output neural network where the number of output neurons represent the prediction horizon.

The respective time series were preprocessed into a state-space vector [73] with embedding dimension $D = 5$ and time-lag $T = 2$. Note that each respective state space contained MSA prediction



(a) 5 step-ahead prediction (b) 10 step-ahead prediction

Fig. 3. Mackey–Glass performance evaluation of respective methods.

values; $MSA = 5$ and $MSA = 10$ for the different problems, respectively. The goal was to test two different experimental design for two different values of *MSA* in order to test aspects of scalability and robustness of the respective methods.

Each neural network architecture was tested with different numbers of hidden neurons in trial experiments. This led to selection of 10 hidden neurons for case of 10 MSA prediction, and 7 in the case of 5 MSA prediction, respectively.

The RMSE in Eq. (2) was used as the main performance measures for different prediction horizon,

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

where $y_i, \hat{y}_i$ are the observed and predicted data, respectively. N is the length of the observed data. Eq. (2) is applied for each prediction horizon. Moreover, for each problem, the mean error for all the respective prediction horizon was also reported.

We review the performance of the respective methods in terms of scalability and robustness. Scalability refers to the ability to maintain consistent prediction performance as the prediction horizon increases.

We employ 5 generations as the *depth of search* in CMTL and CNE. Note that all the sub-populations evolve for the same depth of search. The CMAES employs a population size of $P = 4 + floor(3 * \log(W_s))$, where W_s is the total number of weights that is encoded into the sub-population (for case of CNE and CMTL) or the population (for case of EA).

The termination condition of the three problems is when a total of 20,000 $\times$ M function evaluations have been reached for the respective problems.

4.3. Results

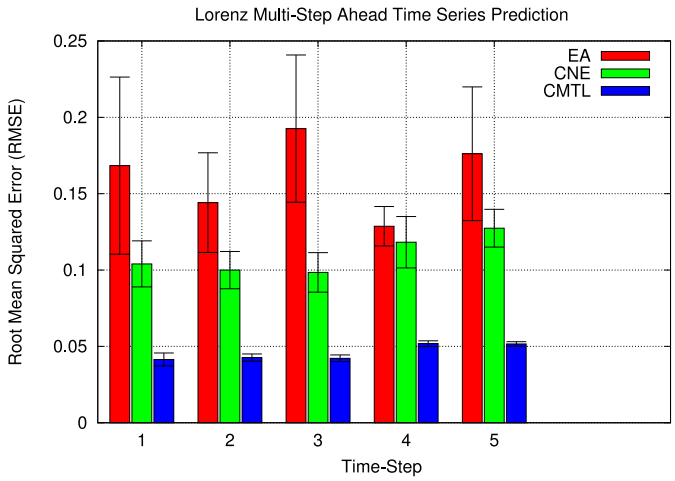
We report the mean and confidence interval of RMSE for each prediction horizon for the respective problem given 30 independent experimental runs. The results are shown in Figs. 3–6 for the simulated time series and Figs. 7–9 for the real-world time series. Note that each problem reports 5-step and 10-step-ahead prediction results. The mean and 95% confidence interval are reported as histogram and error bars for the three methods. We define robustness as the confidence interval which must be as low as possible to indicate high confidence in prediction. We consider scalability as the ability to provide consistent performance to some degree of error given the prediction horizon increases.

Fig. 3 shows that the proposed CTML gives better performance than EA and CNE for the Mackey–Glass time series problem for both cases (5-step (a) and 10-step ahead prediction (b)). Note that EA performance is poor in both cases. We find that the EA and CNE perform better for 5-steps (a) when compared to the first 5-steps of the 10-steps (b). This shows that there is difficulty in scaling up to 10-steps when compared to CMTL which is consistent in performance when we consider both the cases. We further observe that CNE is better than EA in terms of scalability and CMTL provides the best performance overall. There is a similar trend observed for the Lorenz time series shown in Fig. 4. However, in this case, the EA scales up better for the first 5-steps from case (a) to (b) when compared to Mackey-Glass. CMTL provides the best performance and scales up better than EA and CNE in all the cases.

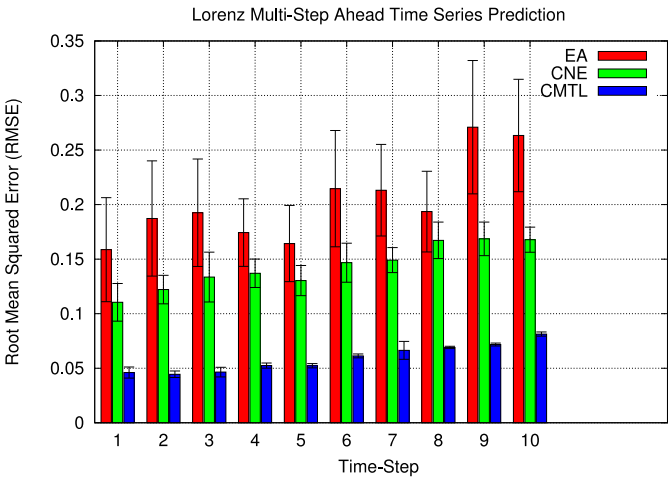
We consider the Rossler time series as shown in Fig. 6 which follows similar trend as previous problems. CMTL performs the best and also scales better when compared to EA and CNE. We move on to the case of Henon time series as shown in Fig. 5. It follows a similar trend as the previous cases. If we consider scalability, then EA and CNE are more scalable than the previous problems. Moreover, CMTL performance does not have significantly major improvement which was observed in previous problems - hence it could be seen that although the trend is similar, but the behaviour is not the same as previous problems. CMTL is closer to EA and CNE from time-step 2 to 10 in (b), when compared to previous problems.

We consider the performance for the respective real-world time series. Fig. 7 for the Sunspot problem shows that the CMTL provides the best performance. The EA does not have a trend as the time-step increases which reflects poor quality in terms of robustness. CNE and CMTL show to have better scalability. The ACI-finance problem shown in Fig. 8 shows similar trend as the Sunspot where CTML gives the best performance. Moving onto the Lazer problem in Fig. 9, the trend is similar to previous problem, however, this is the only case where CTML is not able to beat CNE for times-steps 5–10 as shown in (b). Note that the Lazer problem is highly chaotic which seems to be the major reason behind the difference in performance for the higher prediction horizon given.

Note that in all the respective problems, the confidence interval of CMTL is the lowest when compared to EA and CME. This shows that CMTL provides more robust prediction performance despite different initialization of weight space during training over the multiple experimental runs.

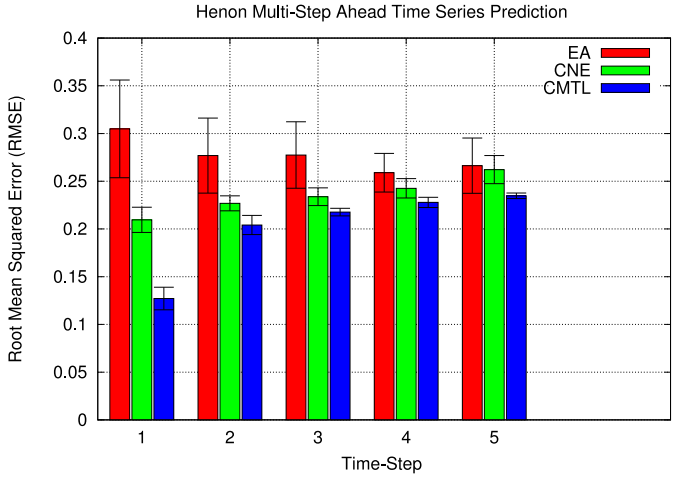


(a) 5 step-ahead prediction

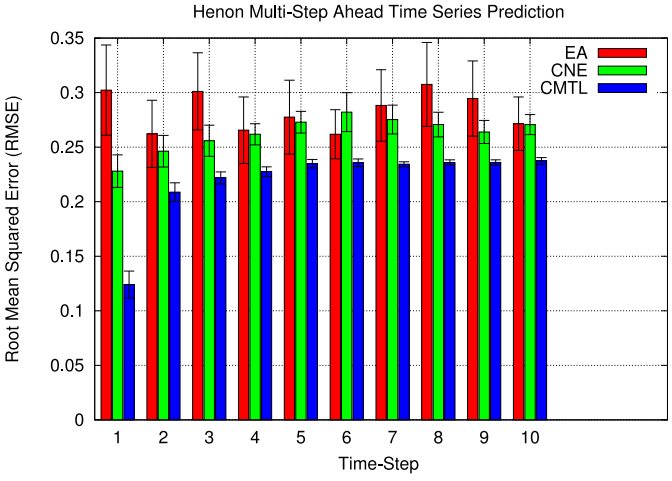


(b) 10 step-ahead prediction

Fig. 4. Lorenz performance evaluation of respective methods.

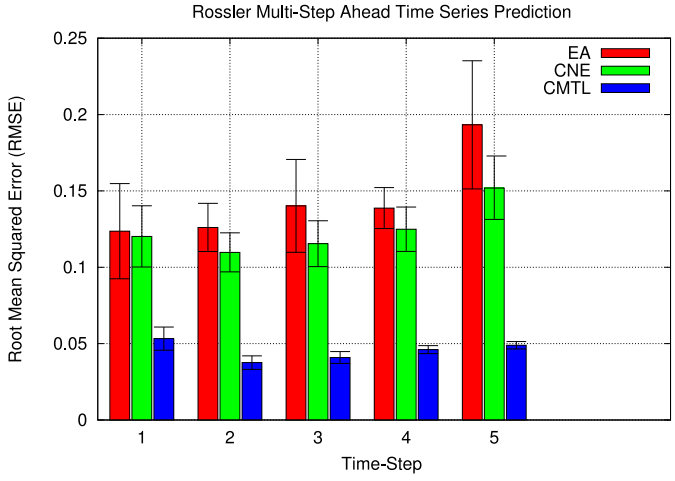


(a) 5 step-ahead prediction

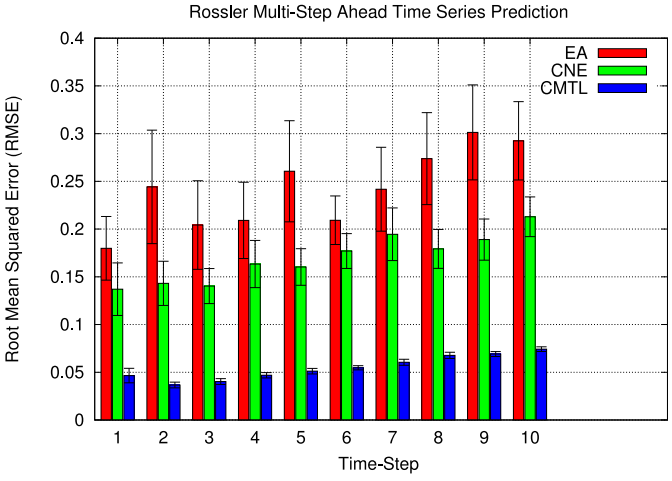


(b) 10 step-ahead prediction

Fig. 5. Henon performance evaluation of respective methods.

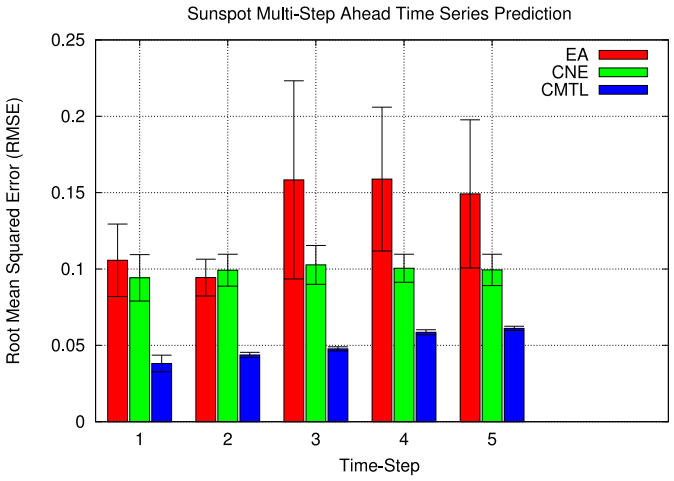


(a) 5 step-ahead prediction

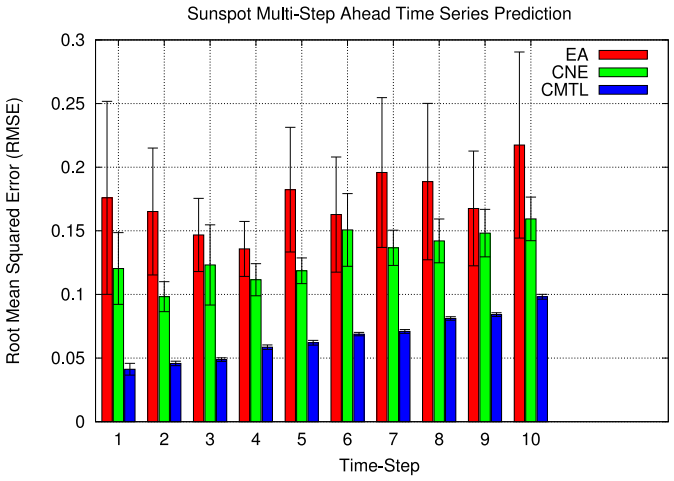


(b) 10 step-ahead prediction

Fig. 6. Rossler performance evaluation of respective methods.

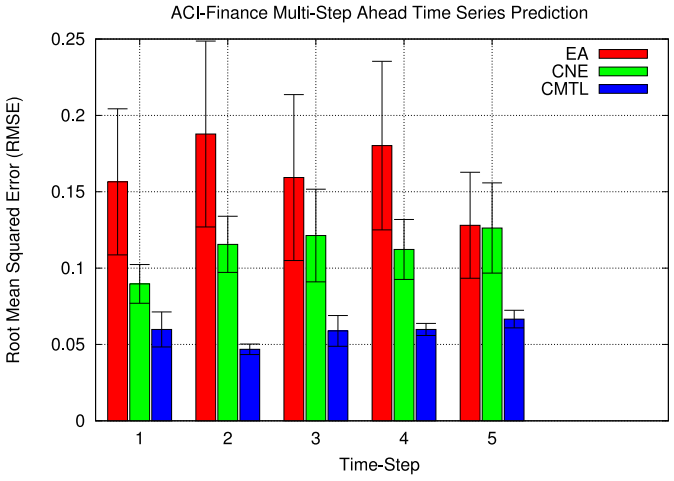


(a) 5 step-ahead prediction

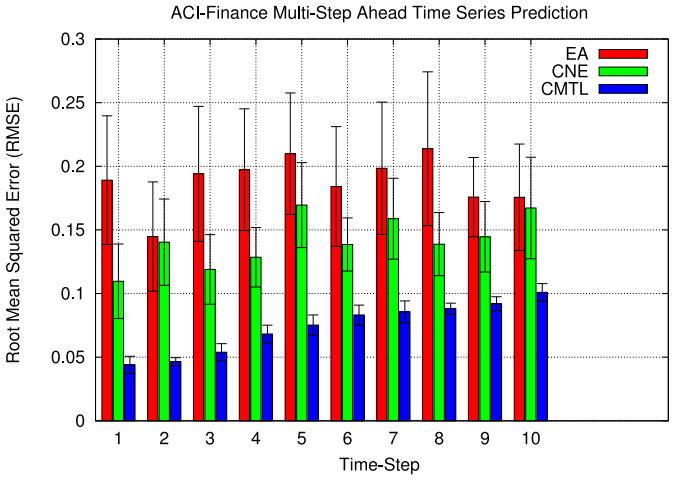


(b) 10 step-ahead prediction

Fig. 7. Sunspot performance evaluation of respective methods.

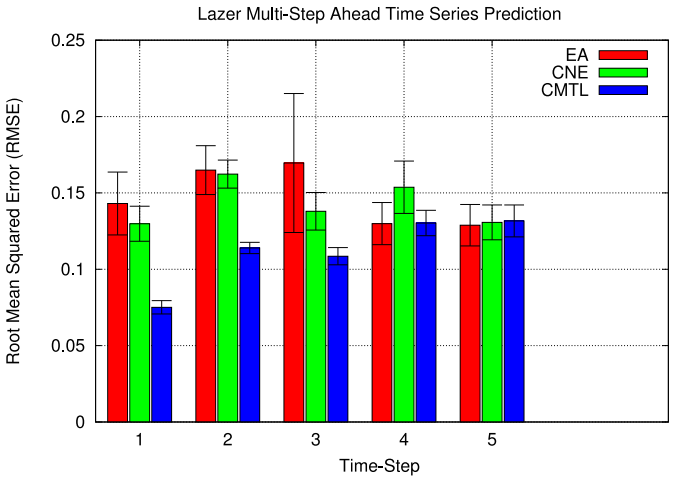


(a) 5 step-ahead prediction

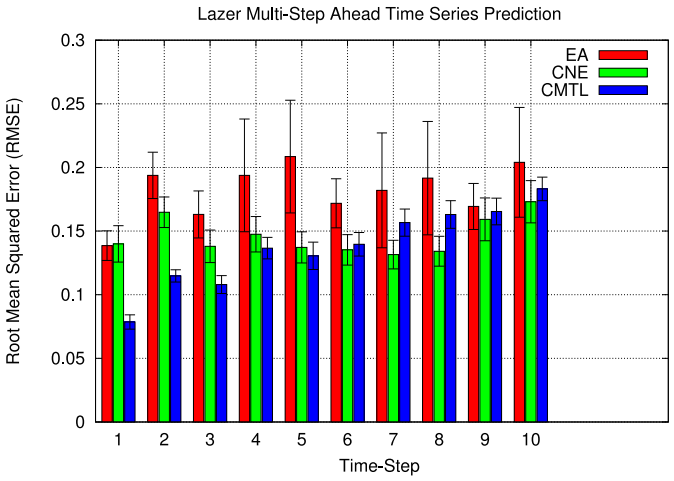


(b) 10 step-ahead prediction

Fig. 8. ACI-finance performance evaluation of respective methods.



(a) 5 step-ahead prediction



(b) 10 step-ahead prediction

Fig. 9. Lazer performance evaluation of respective methods.

Table 1
Performance for 5-step-ahead prediction.

Problem	EA	CNE	CMTL
Mackey–Glass	0.1419 ± 0.0368	0.0879 ± 0.0124	0.0590 ± 0.0022
Lorenz	0.1619 ± 0.0390	0.1096 ± 0.0138	0.0459 ± 0.0024
Henon	0.2769 ± 0.0349	0.2350 ± 0.0110	0.2023 ± 0.0068
Rossler	0.1443 ± 0.0265	0.1244 ± 0.0165	0.0453 ± 0.0041
Sunspot	0.1332 ± 0.0392	0.0992 ± 0.0115	0.0498 ± 0.0023
Lazer	0.1472 ± 0.0218	0.1429 ± 0.0123	0.1119 ± 0.0064
ACI-finance	0.1623 ± 0.0505	0.1130 ± 0.0221	0.0584 ± 0.0069

Table 2
Performance for 10-step-ahead prediction.

Problem	EA	CNE	CMTL
Mackey–Glass	0.2081 ± 0.0452	0.1171 ± 0.0145	0.0800 ± 0.0015
Lorenz	0.2032 ± 0.0460	0.1432 ± 0.0152	0.0591 ± 0.0031
Henon	0.2832 ± 0.0324	0.2627 ± 0.0125	0.2196 ± 0.0048
Rossler	0.2416 ± 0.0440	0.1696 ± 0.0221	0.0547 ± 0.0033
Sunspot	0.1738 ± 0.0508	0.1309 ± 0.0189	0.0659 ± 0.0018
Lazer	0.1816 ± 0.0306	0.1460 ± 0.0133	0.1376 ± 0.0087
ACI-finance	0.1882 ± 0.0474	0.1415 ± 0.0292	0.0737 ± 0.0064

Tables 1 and 2 show the mean and confidence interval (RMSE) across the entire span of the prediction horizon for the two cases (5-step and 10-step ahead) for the respective problems. It is evident from both Tables that CMTL gives the best performance followed by CNE, while EA gives poor performance for respective MSA problems.

4.4. Comparison with the literature

Table 3 shows a comparison with related methods from the literature. We note that the comparison has a number of constraints and not fair as other methods employed different training algorithms and models with different parameter setting in data processing and also in reporting of results with different measures of error. The Mackey-Glass and Lorenz time series performance are compared to two-step-ahead prediction by real-time recurrent learning (RTRL) and echo state networks (ESN) [4]. Note that * in Table 3 implies that the comparison was not very fair due to different embedding dimension reporting of results where it is not clear if the mean or best run has been reported. Further comparison for Mackey-Glass for 5-step-ahead has been shown using extended Kalman filtering (EKF), the unscented Kalman filtering (UKF) and the Gaussian particle filtering (GPF), along with their generalized versions G-EKF, G-UKF and G-GPF, respectively [84].

The Sunspot and Lazer series are compared with support vector regression, iterated (SVR-I), direct (SVR-D), and multiple models (M-SVR) methods [6]. Although we have highlighted the difficulties in attaining a fair comparison, we can see that CMTL performance has been close or beaten some of methods in literature, mainly for Sunspot and Lorenz problems.

5. Discussion

We observed that the performance of CMTL was better than the single-tasking methods for the two major experiment design defined by the extend of the prediction horizon. The results show that the general trend has been similar for most of the problems where CMTL gives the best performance. The major difference in performance was observed only for the Lazer time series which has a very chaotic behaviour when compared to the rest of the problems.

The multi-task learning approach employed shared representation of knowledge where the knowledge form one-step-ahead prediction horizon was used as the main building block for the rest of the prediction horizon. The knowledge from the remaining prediction horizon was utilized in their respective future prediction. The additional number of hidden neurons for each prediction horizon enabled effective storage and refinement of knowledge. This was essentially incremental development of knowledge and refinement through evolution by modular neural network architecture. The modules of knowledge correspond to the different tasks where the efficiency of tasks of the future is dependent on present task at any given time. This is a way to incremental development seems to be absent in the given single task learning methods such as EA and CNE that considered a neural network architecture with multiple outputs. Considering the two single task learning approaches, the performance of EA has been much lower than CNE. In previous work where both CNE and EA were applied to one-step-ahead time series problems [28], there was not a large difference in performance observed amongst them. We used the same benchmark time series problems in this paper, howsoever, MSA prediction was considered. In this paper, although CNE does not have modular knowledge development, it has a way to preserve modularity through problem decomposition which showed to be beneficial for MSA problems when compared to EA. Moreover, previous work employed Elman recurrent neural networks [28] whereas the proposed work employed feedforward networks and therefore, the observations are different when it comes to the performance of the training algorithms.

It can be highlighted that the output layer of the network implemented by CMTL has a form of implicit recurrence as the knowledge in previous state is accumulated for future steps. This recurrence is different from that in Elman style recurrent neural networks used for time series where the recurrence is associated with hidden and context neurons [28]. We also note that that one output neuron is used, hence there is predictive recurrence in CMTL neural network. This recurrence is mainly driven through the transfer of knowledge from the weights in the respective prediction horizon.

An important performance measure of multi-step time series is the extend of the error given as different prediction horizon. It is worthwhile to note that in real-world applications, the confidence of prediction at future prediction horizon is lower than the initial prediction horizon. The single-task learning approaches not only produced high error through the entire scope of the prediction horizon, but performed poorly during the initial prediction horizon when compared to multi-task learning. Hence, we can conclude that the single-task learning methods were least scalable and robust.

The performance of CMTL is slightly dependent on the nature of the time series problem. The most chaotic problem considered was the Lazer problem that also contains high level of noise which can be a problem when previous states of knowledge are used as building blocks for future states. This is why its performance deteriorated when the prediction horizon increased to more than 5 for the 10 step-ahead horizon cases. This has been a case where the knowledge used as the building blocks does not apply in prediction horizon that reaches beyond a certain degree of uncertainty. This could be improved by strategies where more building block resources in terms of weights in the neural network for higher prediction horizon can be employed. Moreover, the overall performance can be improved in enhancing search or learning by adding gradient based local search during learning – this can also speed up the learning process. There is scope for CMTL for multi-variate time series. It has been shown that multivariate prediction performs better as the prediction horizon becomes larger [75].

Although the proposed method has been very effective for most of the cases, there needs further verification of the methods through analytical proofs in future work. This can be done

Table 3
Comparison with literature.

Problem	Method	2-step	5-step	10-step
Mackey–Glass				
	2SA-RTRL* [4]	0.0035		
	ESN* [4]	0.0052		
	EKF [84]		0.2796	
	G-EKF [84]		0.2202	
	UKF [84]		0.1374	
	G-UKF [84]		0.0509	
	GPF [84]		0.0063	
	G-GPF [84]		0.0022	
	CMTL	0.0446	0.0550 ± 0.0046	
Lorenz				
	2SA-RTRL* [4]	0.0382		
	ESN* [4]	0.0476		
	CMTL	0.0426 ± 0.0023		
Lazer				
	M-SVR [6]			0.0244 ± 0.0100
	SVR-I [6]			0.0866 ± 0.0556
	SVR-D [6]			0.2027 ± 0.0100
	CMTL			0.1376 ± 0.0087
Sunspot				
	M-SVR [6]			0.2355 ± 0.0583
	SVR-I [6]			0.2729 ± 0.1414
	SVR-D [6]			0.2151 ± 0.0538
	CMTL			0.0659 ± 0.0018

through convergence proof used in standard evolutionary algorithms [85,86] that could further be extended for multiple subpopulations in CMTL. We note that the experiments have been designed in a way so that robustness and scalability have been tested apart from the prediction error (RMSE). Therefore, we have given comprehensive experimentations that the algorithm converges given different conditions in terms 1) initialization of the population 2) type of problem (real world and simulated) 3) size of the problem in terms of the dataset 4) prediction capability in terms of the difference in the prediction horizon (5 and 10 step-ahead-prediction) and 5) statistical evaluation such as confidence interval across multiple runs. In future work, the set of comprehensive experimentations in this paper can be used to develop theoretical works that can also be supported with convergence proofs.

6. Conclusions and future work

We presented a multi-task learning method for evolving knowledge as building blocks for multi-step-ahead time series prediction. We evaluated the performance with respect with related methods in terms of scalability and robustness. The results show that the proposed method achieves much better performance for almost all the cases for the given time series problems.

The results motivates further investigation for real-world time series applications. The proposed method is flexible and can be improved in terms of training time by incorporating gradient-based learning methods. Moreover, other challenging problems such as multi-variate time series problem can be explored. It would be interesting to find if the approach can be modified for multi-variate and multi-step ahead time series prediction that is required for climate change problems such as cyclones.

References

[1] H. Sandya, P. Hemanth Kumar, S.B. Patil, Feature extraction, classification and forecasting of time series signal using fuzzy and garch techniques, in: National Conference on Challenges in Research & Technology in the Coming Decades (CRT 2013), IET, 2013, pp. 1–7.

[2] R. Chandra, M. Zhang, Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction, Neurocomputing 86 (2012) 116–123.

[3] S.B. Taieb, A.F. Atiya, A Bias and Variance Analysis for Multistep-Ahead Time Series Forecasting, IEEE Trans. Neural Netw. Learn. Syst. 27 (1) (2016) 62–76.

[4] L.-C. Chang, P.-A. Chen, F.-J. Chang, Reinforced two-step-ahead weight adjustment technique for online training of recurrent neural networks, IEEE Trans. Neural Netw. Learn. Syst. 23 (8) (2012) 1269–1278.

[5] R. Boné, M. Crucianu, Multi-step-ahead prediction with neural networks: a review, in: 9emes rencontres internationales: Approches Connexionnistes en Sciences, 2, 2002, pp. 97–106.

[6] L. Zhang, W.-D. Zhou, P.-C. Chang, J.-W. Yang, F.-Z. Li, Iterated time series prediction with multiple support vector regression models, Neurocomputing 99 (2013) 411–422.

[7] S. Ben Taieb, R. Hyndman, Recursive and Direct Multi-Step Forecasting: The Best of Both Worlds, Technical Report, Monash University, Department of Econometrics and Business Statistics, 2012.

[8] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series, Neurocomputing 70 (16) (2007) 2861–2869.

[9] S.B. Taieb, A. Sorjamaa, G. Bontempi, Multiple-output modeling for multi-step-ahead time series forecasting, Neurocomputing 73 (1012) (2010) 1950–1957.

[10] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.

[11] R.K. Ando, T. Zhang, A framework for learning predictive structures from multiple tasks and unlabeled data, J. Mach. Learn. Res. 6 (2005) 1817–1853 http://dl.acm.org/citation.cfm?id=1046920.1194905.

[12] L. Jacob, J. philippe Vert, F.R. Bach, Clustered multi-task learning: a convex formulation, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), Advances in Neural Information Processing Systems 21, Curran Associates, Inc., 2009, pp. 745–752. http://papers.nips.cc/paper/3499-clustered-multi-task-learning-a-convex-formulation.pdf.

[13] J. Chen, L. Tang, J. Liu, J. Ye, A convex formulation for learning shared structures from multiple tasks, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09, ACM, New York, NY, USA, 2009, pp. 137–144, doi:10.1145/1553374.1553392.

[14] J. Zhou, J. Chen, J. Ye, Clustered multi-task learning via alternating structure optimization, in: J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 24, Curran Associates, Inc., 2011, pp. 702–710. http://papers.nips.cc/paper/4292-clustered-multi-task-learning-via-alternating-structure-optimization.pdf.

[15] X. Tang, Q. Miao, Y. Quan, J. Tang, K. Deng, Predicting individual retweet behavior by user similarity: a multi-task learning approach, Knowl.-Based Syst. 89 (2015) 681–688. http://dx.doi.org/10.1016/j.knosys.2015.09.008.

[16] X. Zhang, M.H. Mahoor, Task-dependent multi-task multiple kernel learning for facial action unit detection, Pattern Recognit. 51 (2016) 187–196. http://dx.doi.org/10.1016/j.patcog.2015.08.026.

[17] A. Liu, Y. Lu, W. Nie, Y. Su, Z. Yang, HEp-2 cells classification via clustered multi-task learning, Neurocomputing 195 (2016) 195–201. http://dx.doi.org/10.1016/j.neucom.2015.06.108. Learning for Medical Imaging.

[18] X. Qin, X. Tan, S. Chen, Mixed bi-subject kinship verification via multi-view multi-task learning, Neurocomputing 19 (214) (2016) 350–357. http://dx.doi.org/10.1016/j.neucom.2016.06.027.

[19] S. Zhang, Y. Sui, S. Zhao, X. Yu, L. Zhang, Multi-local-task learning with global regularization for object tracking, Pattern Recognit. 48 (12) (2015) 3881–3894. http://dx.doi.org/10.1016/j.patcog.2015.06.005.

[20] X. Cheng, N. Li, T. Zhou, L. Zhou, Z. Wu, Object tracking via collaborative multi-task learning and appearance model updating, Appl. Soft Comput. 31 (2015) 81–90. http://dx.doi.org/10.1016/j.asoc.2015.03.002.

[21] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359, doi:10.1109/TKDE.2009.191.

[22] P. Angeline, G. Saunders, J. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Netw. 5 (1) (1994) 54–65, doi:10.1109/72.265960.

[23] M. Potter, K. De Jong, A cooperative coevolutionary approach to function optimization, in: Y. Davidor, H.-P. Schwefel, R. Mnner (Eds.), Parallel Problem Solving from Nature PPSN III, Springer, 1994, pp. 249–257. Volume 866 of Lecture Notes in Computer Science.

[24] M.A. Potter, K.A. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (2000) 1–29.

[25] N. García-Pedrajas, D. Ortiz-Boyer, A cooperative constructive method for neural networks for pattern recognition, Pattern Recognit. 40 (1) (2007) 80–98.

[26] F. Gomez, R. Mikkulainen, Incremental evolution of complex general behavior, Adapt. Behav. 5 (3–4) (1997) 317–342. http://dx.doi.org/10.1177/105971239700500305.

[27] R. Chandra, M. Frean, M. Zhang, On the issue of separability for problem decomposition in cooperative neuro-evolution, Neurocomputing 87 (2012) 33–40, doi:10.1016/j.neucom.2012.02.005.

[28] R. Chandra, M. Zhang, Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction, Neurocomputing 186 (2012) 116–123, doi:10.1016/j.neucom.2012.01.014.

[29] R. Chandra, Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction, IEEE Trans. Neural Netw. Learn. Syst. 26 (2015) 3123–3136.

[30] B.L. Happel, J.M. Murre, Design and evolution of modular neural network architectures, Neural Netw. 7 (67) (1994) 985–1004. http://dx.doi.org/10.1016/S0893-6080(05)80155-8. Models of Neurodynamics and Behavior.

[31] J. Clune, J.-B. Mouret, H. Lipson, The evolutionary origins of modularity, Proc. R. Soc. Lond. B: Biol. Sci. 280 (1755) (2013) p. 20122863, doi:10.1098/rspb.2012.2863.

[32] K.O. Ellefsen, J.-B. Mouret, J. Clune, Neural modularity helps organisms evolve to learn new skills without forgetting old skills, PLoS Comput. Biol. 11 (4) (2015) 1–24.

[33] A. Gupta, Y.S. Ong, L. Feng, Multifactorial evolution: toward evolutionary multitasking, IEEE Trans. Evolut. Comput. 20 (3) (2016) 343–357.

[34] Y.S. Ong, A. Gupta, Evolutionary multitasking: a computer science view of cognitive multitasking, Cognit. Comput. 8 (2) (2016) 125–142.

[35] R. Chandra, A. Gupta, Y.S. Ong, C.K. Goh, Evolutionary multi-task learning for modular training of feedforward neural networks, in: A. Hirose, S. Ozawa, K. Doya, K. Ikeda, M. Lee, D. Liu (Eds.), Neural Information Processing. ICONIP 2016. Lecture Notes in Computer Science, vol 9948, Springer, 2016, pp. 37–46.

[36] J.N. Tsitsiklis, B.V. Roy, Neuro-dynamic programming overview and a case study in optimal stopping, in: Proceedings of the 36th IEEE Conference on Decision and Control, 1997, 2, 1997, pp. 1181–1186.

[37] X. Fang, D. Zheng, H. He, Z. Ni, Data-driven heuristic dynamic programming with virtual reality, Neurocomputing 166 (2015) 244–255.

[38] R. Chandra, YS. Ong, CK. Goh, Co-evolutionary multi-task learning for dynamic time series prediction (2017). CoRR abs/1703.01887 https://arxiv.org/abs/1703.01887.

[39] A. Grigorievskiy, Y. Miche, A.-M. Ventel, E. Scopyright verin, A. Lendasse, Long-term time series prediction using OP-ELM, Neural Netw. 51 (2014) 50–56.

[40] C.N. Ng, P.C. Young, Recursive estimation and forecasting of non-stationary time series, J. Forecast. 9 (2) (1990) 173–204.

[41] H.T. Su, T.J. McAvoy, P. Werbos, Long-term predictions of chemical processes using recurrent neural networks: a parallel training approach, Ind. Eng. Chem. Res. 31 (5) (1992) 1338–1352, doi:10.1021/ie00005a014.

[42] A. Parlos, O. Rais, A. Atiya, Multi-step-ahead prediction using dynamic recurrent neural networks, Neural Netw. 13 (7) (2000) 765–786. http://dx.doi.org/10.1016/S0893-6080(00)00048-4.

[43] A. Girard, C.E. Rasmussen, J.Q. Candela, R. Murray-Smith, Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting, in: Advances in neural information processing systems, 2003, pp. 545–552.

[44] G. Niu, B.-S. Yang, Dempstershafer regression for multi-step-ahead time-series prediction towards data-driven machinery prognosis, Mech. Syst. Signal Process. 23 (3) (2009) 740–751. http://dx.doi.org/10.1016/j.ymssp.2008.08.004.

[45] Y. Bao, T. Xiong, Z. Hu, Multi-step-ahead time series prediction using multiple-output support vector regression, Neurocomputing 129 (2014) 482–493. http://dx.doi.org/10.1016/j.neucom.2013.09.010.

[46] S.B. Taieb, G. Bontempi, A.F. Atiya, A. Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the {NN5} forecasting competition, Expert Syst. Appl. 39 (8) (2012) 7067–7083. http://dx.doi.org/10.1016/j.eswa.2012.01.039.

[47] M. Marcellino, J.H. Stock, M.W. Watson, A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series, J. Econom. 135 (1) (2006) 499–526.

[48] T. Proietti, Direct and iterated multistep {AR} methods for difference stationary processes, Int. J. Forecast. 27 (2) (2011) 266–280. http://dx.doi.org/10.1016/j.ijforecast.2010.05.014.

[49] G. Chevillon, Multistep forecasting in the presence of location shifts, Int. J. Forecast. 32 (1) (2016) 121–137. http://dx.doi.org/10.1016/j.ijforecast.2015.04.004.

[50] H. ElMoaqet, D.M. Tilbury, S.K. Ramachandran, Multi-step ahead predictions for critical levels in physiological time series, IEEE Trans. Cybern. 46 (7) (2016) 1704–1714, doi:10.1109/TCYB.2016.2561974.

[51] P.-A. Chen, L.-C. Chang, F.-J. Chang, Reinforced recurrent neural networks for multi-step-ahead flood forecasts, J. Hydrol. 497 (2013) 71–79. http://dx.doi.org/10.1016/j.jhydrol.2013.05.038.

[52] F.-J. Chang, P.-A. Chen, Y.-R. Lu, E. Huang, K.-Y. Chang, Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control, J. Hydrol. 517 (2014) 836–846. http://dx.doi.org/10.1016/j.jhydrol.2014.06.013.

[53] J. Smrekar, P. Potonik, A. Seneganik, Multi-step-ahead prediction of {NOx} emissions for a coal-based boiler, Appl. Energy 106 (2013) 89–99. http://dx.doi.org/10.1016/j.apenergy.2012.10.056.

[54] M.D. Giorgi, M. Malvoni, P. Congedo, Comparison of strategies for multi-step ahead photovoltaic power forecasting models based on hybrid group method of data handling networks and least square support vector machine, Energy 107 (2016) 360–373. http://dx.doi.org/10.1016/j.energy.2016.04.020.

[55] D. Yang, K. Yang, Multi-step prediction of strong earthquake ground motions and seismic responses of {SDOF} systems based on EMD-ELM method, Soil Dyn. Earthq. Eng. 85 (2016) 117–129. http://dx.doi.org/10.1016/j.soildyn.2016.03.015.

[56] X. Wu, Y. Wang, J. Mao, Z. Du, C. Li, Multi-step prediction of time series with random missing data, Appl. Math. Model. 38 (14) (2014) 3512–3522. http://dx.doi.org/10.1016/j.apm.2013.11.029.

[57] J. Chen, J. Liu, J. Ye, Learning incoherent sparse and low-rank patterns from multiple tasks, ACM Trans. Knowl. Discov. Data 5 (4) (2012) 22:1–22:31, doi:10.1145/2086737.2086742.

[58] Y. Zhang, D.-Y. Yeung, Transfer metric learning by learning task relationships, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'10, ACM, New York, NY, USA, 2010, pp. 1199–1208, doi:10.1145/1835804.1835954. http://doi.acm.org/10.1145/1835804.1835954.

[59] B. Bakker, T. Heskes, Task clustering and gating for bayesian multitask learning, J. Mach. Learn. Res. 4 (2003) 83–99, doi:10.1162/153244304322765658.

[60] S. Zhong, J. Pu, Y.-G. Jiang, R. Feng, X. Xue, Flexible multi-task learning with latent task grouping, Neurocomputing 189 (2016) 179–188. http://dx.doi.org/10.1016/j.neucom.2015.12.092.

[61] J. Xu, P.N. Tan, L. Luo, Orion: Online regularized multi-task regression and its application to ensemble forecasting, in: Proceedings of 2014 IEEE International Conference on Data Mining, 2014, pp. 1061–1066, doi:10.1109/ICDM.2014.90.

[62] M. Leutbecher, T.N. Palmer, Ensemble forecasting, J. Comput. Phys. 227 (7) (2008) 3515–3539, doi:10.1016/j.jcp.2007.02.014.

[63] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling up fast evolutionary programming with cooperative coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computation, San Diego, CA, USA, 2001, pp. 1101–1108.

[64] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inf. Sci. 178 (15) (2008) 2985–2999. http://dx.doi.org/10.1016/j.ins.2008.02.017.

[65] N. Garc-a-Pedrajas, C. Hervas-Martinez, J. Munoz-Perez, Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks), Neural Netw. 15 (2002) 1259–1278.

[66] N. Garcia-Pedrajas, C. Hervas-Martinez, J. Munoz-Perez, Covnet: a cooperative coevolutionary model for evolving artificial neural networks, IEEE Trans. Neural Netw. 14 (3) (2003) 575–596, doi:10.1109/TNN.2003.810618.

[67] N. Garca-Pedrajas, D. Ortiz-Boyer, C. Hervas-Martnez, Cooperative coevolution of generalized multi-layer perceptrons, Neurocomputing 56 (2004) 257–283. http://dx.doi.org/10.1016/j.neucom.2003.09.004.

[68] F. Zhu, S.-U. Guan, Cooperative co-evolution of GA-based classifiers based on input decomposition, Eng. Appl. Artif. Intell. 21 (8) (2008) 1360–1369. http://dx.doi.org/10.1016/j.engappai.2008.01.009.

[69] F. Gomez, J. Schmidhuber, R. Miikkulainen, Accelerated neural evolution through cooperatively coevolved synapses, J. Mach. Learn. Res. 9 (2008) 937–965.

[70] C.-J. Lin, C.-H. Chen, C.-T. Lin, A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications, IEEE Trans. Syst. Man. Cybern. Part C: Appl. Rev. 39 (1) (2009) 55–68.

[71] R. Chandra, M. Frean, M. Zhang, C.W. Omlin, Encoding subcomponents in cooperative co-evolutionary recurrent neural networks, Neurocomputing 74 (17) (2011) 3223–3234, doi:10.1016/j.neucom.2011.05.003.

[72] R. Chandra, M. Frean, M. Zhang, Adapting modularity during learning in cooperative co-evolutionary recurrent neural networks, Soft Comput. – Fusion Found. Methodol. Appl. 16 (6) (2012) 1009–1020.

[73] F. Takens, Detecting strange attractors in turbulence, in: Dynamical systems and turbulence, Warwick 1980, Springer, Berlin Heidelberg, 1981, pp. 366–381.

[74] C. Frazier, K. Kockelman, Chaos theory and transportation systems: instructive example, Transp. Res. Rec.: J. Transp. Res. Board 20 (2004) 9–17.

[75] M. Chayama, Y. Hirata, When univariate model-free time series prediction is better than multivariate, Phys. Lett. A 380 (3132) (2016) 2359–2365. http://dx.doi.org/10.1016/j.physleta.2016.05.027.

[76] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolut. Comput. 11 (1) (2003) 1–18, doi:10.1162/106365603321828970.

[77] M. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.

[78] E. Lorenz, Deterministic non-periodic flows, J. Atmos. Sci. 20 (1963) 267–285.
[79] M. Hénon, A two-dimensional mapping with a strange attractor, Commun. Math. Phys. 50 (1) (1976) 69–77.
[80] O. RP ssler, An equation for continuous chaos, Phys. Lett. A 57 (5) (1976) 397–398.
[81] S. S., Solar cycle forecasting: a nonlinear dynamics approach, Astron. Astrophys. 377 (2001) 312–320.
[82] A. Weigend, N. Gershenfeld, Time series prediction: Forecasting the future and understanding the past, in: Proceedings of a NATO Advanced Research Workshop on Comparative Time Series Analysis, Santa Fe, New Mexico, 1994.
[83] NASDAQ Exchange Daily: 1970–2010 Open, Close, High, Low and Volume (2015). Available at: http://www.nasdaq.com/symbol/aciw/stock-chart (accessed 02.02.15).
[84] X. Wu, Z. Song, Multi-step prediction of chaotic time-series with intermittent failures based on the generalized nonlinear filtering methods, Appl. Math. Comput. 219 (16) (2013) 8584–8594. http://dx.doi.org/10.1016/j.amc.2013.02.071.
[85] A.E. Eiben, E.H.L. Aarts, K.M.v. Hee, Global convergence of genetic algorithms: a Markov chain analysis, in: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, PPSN I, Springer-Verlag, London, UK, 1991, pp. 4–12. http://dl.acm.org/citation.cfm?id=645821.670200.
[86] R.F. Hartl, R.K. Belew, A global convergence proof for a class of genetic algorithms, University of Technology, Vienna, 1990.

Rohitash Chandra holds a Ph.D. in Computer Science (2012) from Victoria University of Wellington, M.S. in Computer Science (2008) from the University of Fiji and B.Sc. in Computer Science and Engineering Technology (2006) at the University of the South Pacific. His research interests are in areas of deep learning, recurrent neural networks, learning algorithms, neuro-evolution and time series prediction. During the course of this research in 2016, he was a Research Fellow in Machine Leaning at Rolls Royce @ NTU Corp Lab, Nanyang Technological University, Singapore. Currently, he is Chancellor's Fellow at the Centre for Translational Data Science, University of Sydney, Australia.

Yew-Soon Ong is a Professor of Computer Science at the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore. He is Director of the Data Science and Artificial Intelligence Center@NTU and also Principal Investigator of the Data Analytics & Complex Systems Programme in the NTU-Rolls Royce Corporate Lab. His research interest in computational intelligence spans across memetic computing, evolutionary design and machine intelligence. Dr. Ong is Founding Editor-In-Chief of the IEEE Transactions on Emerging Topics in Computational Intelligence, Founding Technical Editor-in-Chief of the Memetic Computing Journal, and Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics, IEEE Transactions on Big Data and others. He has published more than 180 refereed articles and delivered technical talks on computational intelligence as keynote, plenary or invited speaker at international conferences and research institutions worldwide. He received the 2015 IEEE Computational Intelligence Magazine Outstanding Paper Award, 2012 IEEE Transactions on Evolutionary Computation Outstanding Paper Award for his research works in memetic computation and was listed as a Thomson Reuters Highly Cited Researcher in 2015 and 2016.

Chi-Keong Goh received the B.Eng. and Ph.D. degrees in electrical engineering from the National University of Singapore, in 2003 and 2007, respectively. He currently holds the position of R&T Team Lead, Computational Engineering, at the Rolls-Royce Applied Technology Group (ATG) in Singapore. In his current role, Chi-Keong leads a team of technologist in the delivery of data-driven solutions to meet business requirements. He also provides strategic steer in data analytics and optimization through acquisition and development of appropriate technologies within the team as well as external research partners. Prior to joining the ATG, he is a senior research fellow in the Data Storage Institute, Agency for Science, Technology and Research (A*Star). Chi-Keong has served as a reviewer for various international journals and on conference program committees in the fields of Computational Intelligence and Data Mining.