Neurocomputing 000 (2018) 1–12

Contents lists available at ScienceDirect

Neurocomputing

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

Coevolutionary multi-task learning for feature-based modular pattern classification

Rohitash Chandra [a,b,*], Sally Cripps [a,c]

[a] Centre for Translational Data Science, The University of Sydney, NSW 2006, Australia
[b] School of Geosciences, The University of Sydney, NSW 2006, Australia
[c] School of Mathematics and Statistics, The University of Sydney, NSW 2006, Australia

ARTICLE INFO

Article history:
Received 20 September 2017
Revised 18 March 2018
Accepted 8 August 2018
Available online xxx

Communicated by Prof. Zidong Wang.

Keywords:
Neuro-evolution
Modular neural networks
Multi-task learning
Modular knowledge representation

ABSTRACT

Due to modular knowledge representation in biological neural systems, the absence of certain sensory inputs does not hinder decision-making processes. For instance, damage to an eye does not result in loss of one's entire vision. In our earlier work, we presented coevolutionary multi-task learning that featured a synergy between multi-task learning and coevolutionary algorithms. In this paper, we extend this method for robust decision making in pattern classification problems given incomplete information. The method trains a cascaded neural network architecture to autonomously address the absence of certain input features and disruptions to neural connections. The results show that the method is comparable to conventional learning methods whilst having the advantage decision making given incomplete information. Moreover, the method provides a way for developmental learning and simultaneously quantifies feature contribution.

1. Introduction

The concept of modular knowledge representation and learning in the brain has motivated a number of studies, such as human brain functional networks [1] and studies on the roles and effects of modules in the brain [2]. Developmental learning algorithms differ from conventional learning algorithms as the goal is not only to build task-specific knowledge and models, but also to support the growth of learning for continuous development [3,4]. Modular knowledge representation refers to the representation of information learned from data in a complex system into modules, while developmental learning is the process by which new tasks are learnt as they are encountered [5–7]. Although the study of the brain motivated development of artificial neural networks, both network systems use different learning strategies to solve related tasks [8]. For instance, human brain networks use implicit learning while artificial neural networks employ error back-propagation [9–11].

A significant portion of algorithms developed in the early days of machine learning were motivated by studies in cognitive science [12–16]. An example is multi-task learning [17] which is a methodology where learning one task is helpful in learning several other related tasks. For example, learning to drive a car is useful if one wishes to learn to drive another vehicle, such as a tractor or truck. Implicit in this notion of multi-task learning is the assumption that tasks share fundamental knowledge. The knowledge transfer between the tasks is achieved by learning tasks in parallel [17–19].

Multi-task learning was initially developed as a methodology in the neural network literature [17] where mainly gradient-based methods have been used for training. Other methods such as neuroevolution that feature the use of evolutionary algorithms for training neural networks have the potential to be used in the context of multi-task learning [20]. Cooperative coevolution [21] is an evolutionary algorithm that has been effective for training neural networks. The methodology is also referred to as *cooperative neuro-evolution* [22] which is a divide and conquer approach to decompose the network into subcomponents that are implemented as subpopulations. This enforces better modularity in the search space as opposed to conventional evolutionary algorithms since the subpopulations are typically evolved in isolation and cooperation takes place mainly for fitness evaluations [22–26]. In the context of multi-task learning, coevolutionary multi-task learning (CMTL) algorithm has been recently proposed for dynamic time series problems that require robust prediction given partial information [27]. Furthermore, CMTL has been adapted to feature recurrence in order to be used for multi-step time series prediction [28].

* Corresponding author. Tel.: +61413071839.
E-mail addresses: rohitash.chandra@sydney.edu.au (R. Chandra), sally.cripps@sydney.edu.au (S. Cripps).

https://doi.org/10.1016/j.neucom.2018.08.011
0925-2312/

Incomplete information in datasets poses a huge challenge for learning algorithms designed for automated decision making [29]. The Bayesian methodology handles missing data by considering them as unknown parameters and uses marginalisation techniques such as Markov chain Monte Carlo (MCMC) to integrate over these missing values [29,30]. However, in the Bayesian framework, certain assumptions need to be made in regard to the data generating process as specified by the likelihood function and the manner in which the data is incomplete. For instance, an assumption could be that the data is "missing at random" [29,31]. Ensemble learning via boosting is another method of handling incomplete information in datasets [32].

Given the motivations of multi-task learning and modularity in knowledge representation, some of the challenges in addressing incomplete information in datasets can be addressed. In this paper, we present a technique for handling incomplete information through modular pattern classification by learning sequentially from sets of features that are the building blocks of knowledge. Although the concept of modularity is used widely in cooperative neuroevolution [22,26,33], there has not been much work that associates feature groups with modular knowledge representation in the network architecture. The advantage of this representation is that the absence of certain features does not hinder the decision-making process. Hence, decision making is done in a modular manner through the observed subset of features with coevolutionary multi-task learning [27].

The major contribution of the paper is the extension and application of coevolutionary multi-task learning for modular pattern classification. Another contribution is the quantification of the relative importance of feature groups from the dataset. We demonstrate the effectiveness of the method through simulation using a group of binary and multi-class classification problems that are altered to show the contribution of the proposed algorithm given missing information. This paper extends preliminary results in a conference proceeding [34] by providing a formal mathematical representation of the problem and by including a comprehensive evaluation of the performance and behaviour of the algorithm.

The rest of the paper is organised as follows. Section 2 gives an overview of related works while Section 3 presents the proposed methodology with details of multi-task learning and modular pattern classification. Section 4 presents the results and discussion while Section 5 concludes the paper with directions for future research.

2. Related work

2.1. Multi-task learning and modularity in cognitive science

There is a growing use of models of brain function and modularity for the representations for knowledge acquisition and transfer. For example, Anderson et al. presented a discussion on the concept of the reuse of neural circuitry as a fundamental organisational principle of the brain [35]. Related example is the ability of neuronal circuits to function in the case of incomplete wiring via modulatory mechanisms that allow neural circuits to change their properties dynamically for a fixed set of neurons [36]. Other examples include the work by Eliasmith [37], who showed how attractor networks can be used as subsystems in larger neural systems that reiterates concepts that use modularity. Furthermore, Donnarumma et al. presented a neural network hierarchical architecture where multiple modules used the notion of multi-task learning to model complex behaviour [38]. The method was motivated by theories of brain functioning in which skilled behaviours can be generated by combining functional set of different motor primitives [39].

2.2. Modularity in learning via cooperative neuroevolution

Cooperative neuroevolution features problem decomposition of a neural network into subcomponents that are also known as modules [23]. The subpopulation that implements the modules are evolved in isolation and cooperation typically takes place to evaluate the fitness. This ensures that the weights or knowledge learned in the different region of the neural network is preserved as modules which overcome the effects of genetic operators such as the crossover [40,41]. Fitness evaluation in cooperative coevolution can suffer from the challenges in credit assignment since the respective modules provide a partial solution for the overall problem [42]. The ordering and difference in size and interaction between the subcomponents can influence fitness assignment and selection pressure [43]. Hence, the need for robust problem decomposition and adaptation is necessary.

A number of problem decomposition methods have been presented that have strengths and limitations depending on the nature of the problem. Gomez et. al presented problem decomposition according to the lowest level of granularity where each synapse forms a subcomponent [44] for double pole balancing control problems. Chandra et al. [22], showed that using synapse level subcomponents, with the implicit assumption of independence between the synapses, resulted in poor pattern classification. To address this issue, Chandra et al. [45] presented a method whereby initial synapse level subcomponents merged into larger subcomponents different phases of evolution. This motivated a competitive and collaborative methodology for problem decomposition [26] which showed promising performance for time series prediction. Other applications of cooperative neuro-evolution include pattern classification [46]. Moreover, enhancements to address fitness assignment has been done through multi-objective optimisation [47,48]. Furthermore, Chandra et al. presented coevolutionary multi-task learning for providing decision given partial information [27,28]. In this case, the problem decomposition method is based on principles from dynamic programming where knowledge from the previous subtasks are reused.

3. Coevolutionary multi-task learning for modular pattern classification

3.1. Modular knowledge representation

We give an overview of the motivation for the proposed method using a robot navigation task that needs four input sensors as shown in Fig. 1. The robot navigation task is essentially a multi-class pattern classification problem where a neural network is used to control a forward moving robot from obstacles. A modular knowledge representation would ensure that the robot can continue navigation even if one of the input sensors fail, provided that the sensor is not important to the decision to be made during that stage of the journey. We denote the sensors by the letter $\mathbf{x}$. The decision made by the robot depends upon observations from the sensors, where the contribution of a single for decision making depends upon the motion of the robot. If the robot is moving forward, the two front sensors ($\mathbf{x}_1$ and $\mathbf{x}_2$ in Fig. 1) contribute the most towards decision making. If the robot is turning, then the two side sensors are the most important. The goal is to develop a dynamic control system that can provide navigational support even if certain sensors fail during the journey. In a conventional or non-modular control, the system would encounter serious navigation problems when a sensor fails. However, in a modular control system, the sensors $\mathbf{x}_1, \ldots, \mathbf{x}_4$, could be grouped into two feature groups, which we denote by X, with feature groups $X_1 = (\mathbf{x}_1, \mathbf{x}_2)$ and $X_2 = (\mathbf{x}_3, \mathbf{x}_4)$. Therefore, if the robot is moving forward, then only input from feature group X_1 is needed and the control system

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

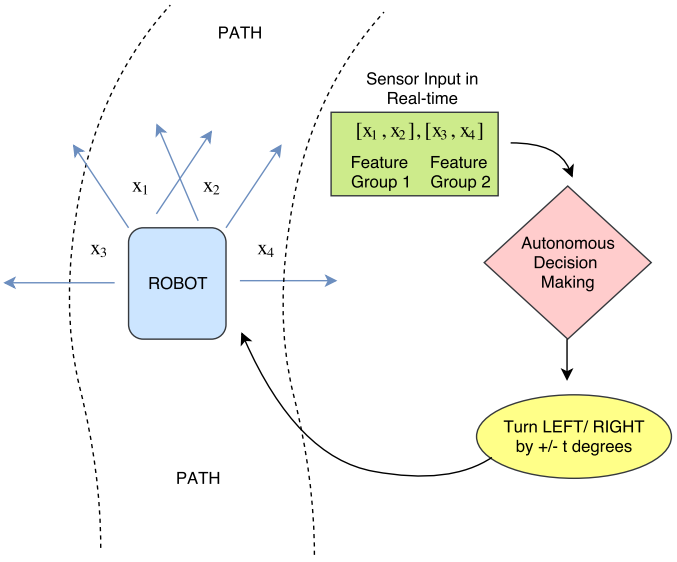

Fig. 1. Control for an autonomous robot based on sensor inputs ($\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$). Note that the input features are grouped according to their position in the robot. The font and side sensors define the two feature groups. The robot is designed to only move forward, hence the front sensors have a higher level of importance for decision making when compared to the side sensors. Therefore, the front sensors will become the first or base subtask.

will be able to make a decision even if observations from the sensors in feature group X_2 are not available.

The modules of knowledge which are the weights corresponding to the feature and neuron groups serve as sequential building blocks of knowledge used in decision making. The analogy to multi-task learning is to consider feature groups as subtasks and the weights corresponding to these feature groups as the shared knowledge for multi-task learning. In this manner, adaptive and modular pattern classification can continue when certain input features (X_2) are unavailable.

In the example of the robot navigation problem in Fig. 1, we decompose the complete feature space into two feature groups. The first and most important feature group is $\mathbf{x}_1$ and $\mathbf{x}_2$ as these constitute the front sensors used for a forward moving robot. The first neuron group contains only one neuron. The weights corresponding to the input and output layer of the first feature and neuron group is the first knowledge modules. The second feature group consists of the lateral sensors, $\mathbf{x}_3$ and $\mathbf{x}_4$. The way the feature groups are decomposed and implemented in a cascaded network architecture is shown in Fig. 2 where the decomposition of the hidden neurons in relation to the feature groups is also shown. The module of knowledge contained in the first feature group are the weights which map features $\mathbf{x}_1$ and $\mathbf{x}_2$ to the first hidden neuron, as well as the weight which maps that hidden neuron to the output. The knowledge modules featured in the second feature group are similarly defined. We define a feature space to be the increasing union of previous feature groups. If data from the sensors $\mathbf{x}_3$ and $\mathbf{x}_4$ are not available, then the robot will still be able to continue in a safe mode using information contained in feature group 1.

3.2. Coevolutionary multi-task learning

In this section, we extend coevolutionary multi-task learning which was originally designed for dynamic time series prediction [27] for modular pattern classification. Through CMTL, coevolution is used to evolve the respective modules via the sub-populations in a round-robin fashion. In the extension of CMTL, there are two levels of decomposition, one on the dataset space and the other on the weight space. The dataset space considers the feature groups for decomposition whereas the weight space features the decomposition of the neural network for modules of knowledge which is highlighted in Fig. 2. Note that although conventional multi-task learning refers to tasks defined in the output or decision space, the proposed method decomposes a conventional classification problem as subtasks that constitute the input space given by feature groups. Therefore, the overall classification problem is decomposed into subtasks that learn sequentially through the use of groups of the feature groups and modules of network weights. The base problem is a feedforward neural network (FNN) with the lowest number of hidden neurons that represents the first or foundational subtask. Hence, the number of hidden neurons would increase for every additional subtask.

More formally, we partition the F features into $M \leq F$ collectively exhaustive and mutually exclusively feature modules, denoted by $\mathcal{M}$. Similarly, we partition the H hidden neurons into $G \leq H$ collectively exhaustive and mutually exclusive modules. Define $\gamma_f = m$, if feature f is in module m, let n_{f_m} be the corresponding number of features in module m for $m = 1, \ldots, M$ and $f = 1, \ldots, F$. Similarly, define $\delta_h = m$, if neuron h is in module m, for $h = 1, \ldots, H$ and $m = 1, \ldots, M$.

Let X be the matrix containing N samples on all F features. Define $A_m = \{f; \gamma_f = m\}$, to be a set of indices allocating features to modules so that X_{A_m} is the sub-matrix containing samples on the features in module m. For instance, suppose $X = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$, so that $F = 4$. Let the number of modules $M = 3$ and set $A_1 = \{1, 2\}$, $A_2 = \{3\}$, $A_3 = \{4\}$. Then,

$$\begin{aligned} X_{A_1} &= [\mathbf{x}_1, \mathbf{x}_2] \\ X_{A_2} &= [\mathbf{x}_3] \\ X_{A_3} &= [\mathbf{x}_4]. \end{aligned} \tag{1}$$

In addition we will define $A_{p:\,q}$ to be the union of the set of indices in the sets A_p and A_q. For example, $A_{1:2} = \{1, 2, 3\}$ so that the feature space $\mathcal{F}_2 = X_{A_{1:2}} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$.

Similarly, let Z be the N by H matrix of hidden neuron values. Define $B_m = \{h; \gamma_h = m\}$, to be a set of indices allocating hidden neurons to module m, so that Z_{B_m} is the sub-matrix containing values of the hidden neurons in module m. Let W be the $F \times H$ matrix of weights with elements w_{fh} which map the input feature f to hidden neuron h, for $f = 1, \ldots, F$ and $h = 1, \ldots H$. This matrix is partitioned as follows; Let P_{ij} be the Cartesian product of sets A_i and B_j, and let $\mathbf{w}_{P_{ij}}$ be a vector containing the elements of the sub-matrix of weights which maps the values X_{A_i} to Z_{B_j}. Let n_{f_m} be the number of features in module m and let n_{h_m} be the number of hidden neurons in module m. For instance, if $H = 4$, $M = 2$, $B_1 = \{1, 2\}$ and $B_2 = \{3, 4\}$, then

$$\begin{aligned} Z_{B_1} &= [\mathbf{z}_1, \mathbf{z}_2] \\ Z_{B_2} &= [\mathbf{z}_3, \mathbf{z}_4] \end{aligned} \tag{2}$$

and

$$\begin{aligned} P_{1,1} &= \{(1,1), (1,2), (2,1), (2,2)\} \\ P_{1,2} &= \{(1,3), (1,4), (2,3), (2,4)\} \\ P_{2,1} &= \{(3,1), (3,2)\} \\ P_{2,2} &= \{(3,3), (3,4)\} \end{aligned} \tag{3}$$

and therefore

$$\begin{aligned} \mathbf{w}_{P_{1,1}} &= [w_{1,1}, w_{2,1}, w_{1,2}, w_{2,2}]', \\ \mathbf{w}_{P_{1,2}} &= [w_{1,3}, w_{2,3}, w_{1,3}, w_{2,4}]', \\ \mathbf{w}_{P_{2,1}} &= [w_{3,1}, w_{3,2}]', \\ \mathbf{w}_{P_{2,2}} &= [w_{3,3}, w_{3,4}]', \end{aligned} \tag{4}$$

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

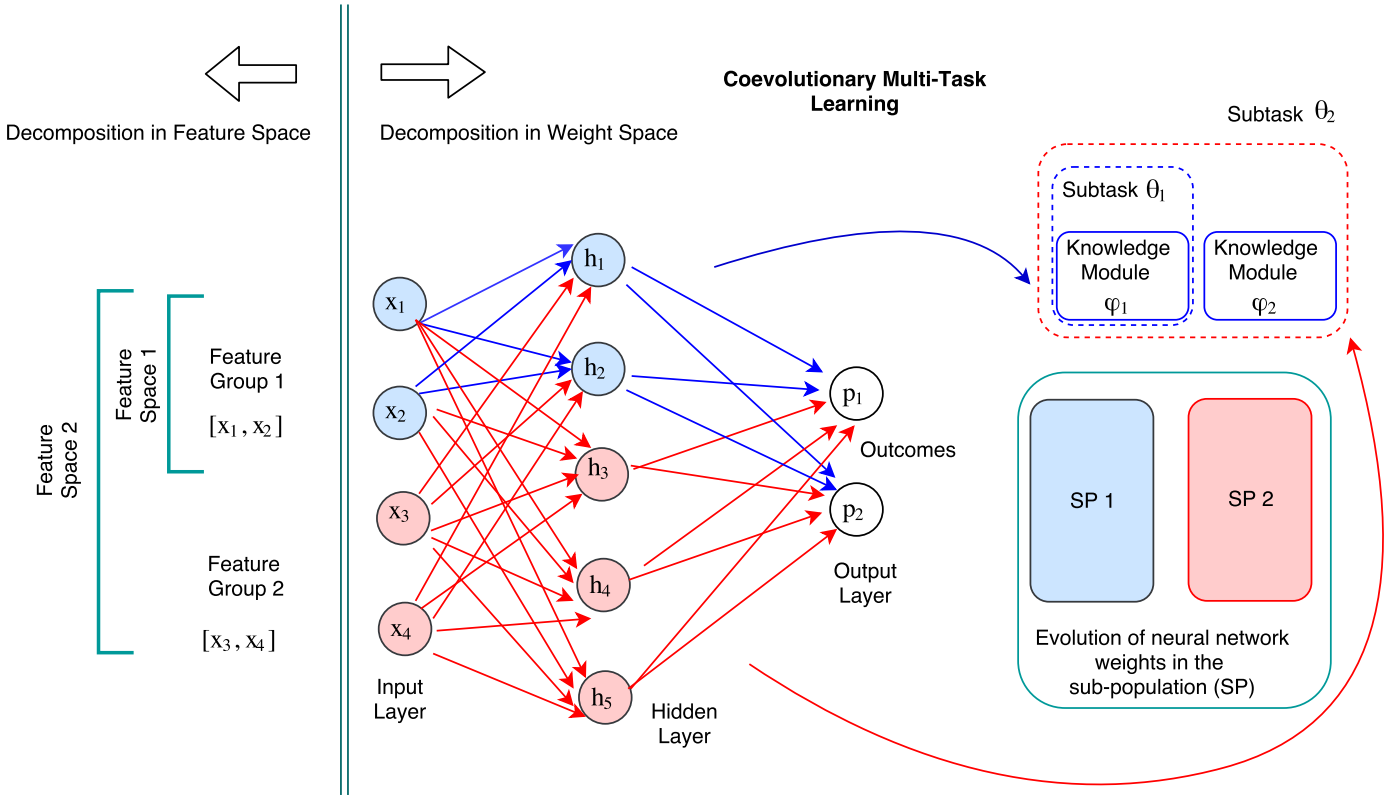

Fig. 2. We highlight the decomposition process in feature and weight space for modular pattern classification by coevolutionary multi-task learning. Subtask 1 employs a network topology with 2 hidden neurons while Subtask 2 adds extra 2 input features and 3 hidden neurons. Note the cascading approach in the relationship of feature space with feature groups and the subtasks with the knowledge modules.

Define V to be the $H \times C$ matrix of weights, with elements v_{hc} which maps hidden neuron h to output class c , for $h = 1, \ldots, H$ and $c = 1, \ldots, C$. Let Q_{jc} be the Cartesian product of sets B_j and c, and define $\mathbf{v}_{Q_{jc}}$ to be the sub-vector of weights which map Z_{B_j} to y_c. For ease of notation, we assume that y is binary in order to drop the dependence on c. However, the extension to outputs with more than two classes is straightforward. For instance, if $C = 1$ and $Y = [y_1]$, then

$$Q_1 = \{(1,1),(2,1)\}$$
$$Q_2 = \{(3,1),(4,1)\} \tag{5}$$

and therefore

$$\mathbf{v}_{Q_1} = [v_{1,1}, v_{2,1}],$$
$$\mathbf{v}_{Q_2} = [v_{3,1}, v_{4,1}] \tag{6}$$

The feature groups are ordered $X_{A_1}, X_{A_2}, \ldots, X_{A_S}$, in some fashion, related to their "relevance".

We now define a subtask θ_m. A subtask consists of an increasing sequence of knowledge modules defined by the weights that correspond to the feature and hidden neuron group. For example, subtask θ_1 consists of knowledge learnt from feature group X_{A_1}, while subtask θ_2 features knowledge from subtask X_{A_2}. The sets of indices allocating features and hidden neurons to subtask m are the union of indices of the modules, up to and including module m. We use the notation $A_{1:m} = \prod \bigcup_{s=1}^{m} A_s$, and $B_{1:m} = \prod \bigcup_{s=1}^{m} B_s$ to denote these subtask specific indices. The vector weights corresponding to each subtask, θ_m, are constructed as follows;

$$\Phi_1 = [\mathbf{w}_{P_{1,1}}, \mathbf{v}_1]; \quad \theta_1 = (\Phi_1)$$
$$\Phi_2 = [\mathbf{w}_{P_{1:2,2}}, \mathbf{w}_{P_{2,1:1}}, \mathbf{v}_{Q_2}]; \quad \theta_2 = [\theta_1, \Phi_2]$$
$$\vdots$$
$$\Phi_M = [\mathbf{w}_{P_{1:M,M}}, \mathbf{w}_{P_{M,1:M-1}}, \mathbf{v}_{Q_M}]; \quad \theta_M = [\theta_{M-1}, \Phi_M] \tag{7}$$

The total vector of weights is therefore $\mathbf{\Phi} = (\Phi_1, \ldots, \Phi_M)$. We obtain an estimate of $\hat{\mathbf{\Phi}} = (\hat{\Phi}_1, \ldots, \hat{\Phi}_M)$ where

$$\hat{\Phi}_m = \text{argmin}_{\mathbf{\Phi}_m} L(\Phi_m, \hat{\theta}_{t-1})$$

and

$$L(\Phi_m, \hat{\theta}_{m-1}) = \sum_{i=1}^{n} \left(\hat{y}_i(\Phi_m, \hat{\theta}_{m-1}) - y_i \right)^2, \tag{8}$$

where $\hat{y}_i(\Phi_m, \hat{\theta}_{m-1})$ is the fitted value of output y_i, and given by

$$\hat{y}_i(\Phi_m, \hat{\theta}_{m-1}) = f\left\{ \sum_{j \in B_{1:m-1}} \hat{v}_j z_{i,j} + \sum_{j \in B_m} v_j^{[k]} z_{i,j} \right\} \tag{9}$$

where

$$z_{i,j} = f\left\{ \sum_{p \in A_{1:m-1}} \hat{w}_{p,j} x_{i,p} + \sum_{p \in A_m} w_{p,j}^{[k]} x_{i,p} \right\} \text{ for } j \in B_{1:m-1}$$

$$z_{i,j} = f\left\{ \sum_{p \in A_{1:m}} w_{p,j}^{[k]} x_{i,p} \right\} \text{ for } j \in B_m$$

and where f is the sigmoid function. This is implemented by Algorithm 1 which returns the loss of a subtask m given data from the feature group X_{A_m} and response Y.

Algorithm 2 gives further details of the coevolutionary multi-task learning algorithm used. The dataset is divided into feature groups as given by X_{A_m} in Eq. (2). The algorithm begins with cycle $c = 0$, by generating a $J \times N_m$ matrix, $S_m^{[c]}$, for $m = 1, \ldots, M$. Where $N_m = n_{f_m} \times n_{h_m} + n_{h_m}$ is the length of $\boldsymbol{\theta}_m$, and J is number of replications of $\boldsymbol{\theta}_m$, which are the individuals of the respective

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

Algorithm 1: Loss function evaluation.

Data: Feature space X_{A_m}, response Y, and subtask θ_m.
Result: Loss $L(\Phi_m, \hat{\theta}_{m-1})$
Compute forward pass for subtask:
for $m = 1, \ldots, M$ **do**
  if $m > 1$ **then**
    Fix
    $\Phi_{m-1} = [\mathbf{w}_{P_{1:m-1,1:m-1}}, \mathbf{v}_{Q_m}] = [\hat{\mathbf{w}}_{P_{1:m-1,1:m-1}}, \hat{\mathbf{v}}_{Q_m}]$
  end
  -Initialize $k = 0$,
  $\Phi_m^{[k]} = (\mathbf{w}_{P_{1:m-1,m}}^{[k]}, \mathbf{w}_{P_{m,m}}^{[k]}, \mathbf{v}_{Q_m}^{[k]})$
  for $i = 1, \ldots, N$ **do**
    Compute $\hat{y}_i$ as given by Equation (9)
  end
  Compute $L(\Phi_m^{[k]}, \hat{\theta}_{m-1})$ as given by Equation (8)
end

Algorithm 2: Coevolutionary Multi-task Learning.

Data: Feature space X_{A_m}, response Y, and subtask reference m
Result: Optimal values for knowledge modules (weights) for subtasks ϕ_m
Initialization Stage ($c = 0$);
for *each m* **do**
  In cycle $c = 0$ Initialize the values for the matrix in $S_m^{[c=0]}$;
end
for *c=1:C* **do**
  for *m=1:M* **do**
    for *d=1:D* **do**
      for *j=1:J* **do**
        compute $L(\mathbf{s}_{g,m_j}^{[c]} | \hat{\mathbf{s}}_1^{[c]}, \ldots, \hat{\mathbf{s}}_{m-1}^{[c]})$ via Algorithm 1
      end
      for *j=1:J* **do**
        compute fitness ratio

$$FR_{g,m_j}^{[t]} = \frac{L(\mathbf{s}_{g,m_j}^{[c]} | \hat{\mathbf{s}}_1^{[c]}, \ldots, \hat{\mathbf{s}}_{m-1}^{[t]})}{\sum_{j=1}^{J} L(\mathbf{s}_{g,m_j}^{[c]} | \hat{\mathbf{s}}_1^{[c]}, \ldots, \hat{\mathbf{s}}_{m-1}^{[t]})}$$

      end
      for *j=1:J* **do**
        With probability given by $\mathbf{FR}_{g,m_j}^{[c]} = (FR_{g,m_1}^{[c]}, \ldots, FR_{g,m_j}^{[c]})$, select a pair of parents and create new offspring, $\mathbf{s}_{g+1,m_j}^{[c]}$, via evolutionary operators, such as crossover andmutation
      end
    end
  end
  $S_m^{[c+1]}$
end
Set $\hat{\phi}_m = \text{argmin}_j \; L(\mathbf{s}_{G,m_j} | \hat{\phi}_1, \ldots, \hat{\phi}_{m-1})$

sub-population, for $m = 1, \ldots, M$. Each element of $S_m^{[0]}$ is generated from a uniform distribution $U(-\alpha, \alpha)$, where α is defined by the user. Each row of $S_m^{[c]}$, denoted by $\mathbf{s}_{m_j}^{[c]}$, for $j = 1, \ldots, J$, represents a set of values for knowledge module $\boldsymbol{\phi}_m$.

At cycle $c = 1$, the algorithm computes a relative loss function score, which we term fitness ratio and denote by $FR_{m_j}^{[c]}$, for the values in $\mathbf{s}_{m_j}^{[c]}$.

The $FR_{m_j}^{[c]}$, for $m > 1$ is given by,

$$FR_{m_j}^{[c]} = \frac{L(\mathbf{s}_{m_j}^{[c]} | \hat{\mathbf{s}}_1^{[c]}, \ldots, \hat{\mathbf{s}}_{m-1}^{[c]})}{\sum_{j=1}^{J} L(\mathbf{s}_{m_j}^{[c]} | \hat{\mathbf{s}}_1^{[c]}, \ldots, \hat{\mathbf{s}}_{m-1}^{[c]})}$$

so that $\sum_{j=1}^{J} FR_{m_j}^{[t]} = 1$ and

$$\hat{\mathbf{s}}_m^{[t]} = \text{argmin}_j \; L(\mathbf{s}_{m_j}^{[t]} | \hat{\mathbf{s}}_1^{[t]}, \ldots, \hat{\mathbf{s}}_{m-1}^{[t]})$$

The fitness scores are used to create new values, of $\mathbf{s}_{m_j}^{[t]}$, using genetic operators such as selection, crossover, and mutation which are the basis of evolution in coevolutionary algorithms. Note that the method is flexible and any population-based evolutionary algorithms can be used in the subpopulation. In the context of neuroevolution, real-coded genetic algorithms have been typically used in the subpopulations [33,44]. More recently, certain variants such as generalised generation gap with parent-centric crossover evolutionary algorithm (G3-PCX) [26] and covariance matrix adaption evolution strategies (CMAES) [28] have been used in the subpopulations with very promising results. In the implementation of this paper, we will use CMAES in the sub-populations of CMTL.

Until the termination condition is reached, the evolution for each sub-population is carried for the depth of evolution (d generations) in a round-robin fashion as shown in Algorithm 2. Each knowledge module evolution is implemented via the different subpopulations shown in Fig. 2. The termination condition can be either the maximum number of function evaluations or a given fitness value (loss) from the training or validation dataset.

The difference between CMTL and cooperative neuro-evolution (CNE) is the way in which the problem is decomposed, and the way in which the individual fitness is calculated [22,23]. In CMTL, the fitness of an individual from a sub-population S_m depends on the previous subtask which is a dynamic programming approach. In CNE, a divide and conquer approach is used to evaluate the fitness of an individual. Once the termination criterion has been met, the algorithm moves into the testing phase where the best solutions from all the different subtasks are mapped into the network. The implementation code in Matlab has been provided which is available online.[1]

In order to consider the time complexity for the transfer of knowledge from one subtask to another, we need to consider the time taken for transfer of solutions for the cases where the subtask is not the base subtask. Therefore, the worst case time complexity can is given by

$$\begin{aligned} T(m<=1) &= \mathcal{O}(1) \\ T(m) &= T(m-1) + T(m-2), ..., +\mathcal{O}(1) \\ T(m) &= \mathcal{O}(2^m) \end{aligned} \tag{10}$$

Hence, for the entire algorithm, the time complexity can be given by $\mathcal{O}(n \times 2^m)$, where m is the number of modules that corresponds to the subtask and n is the number of individuals in the sub-population.

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

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

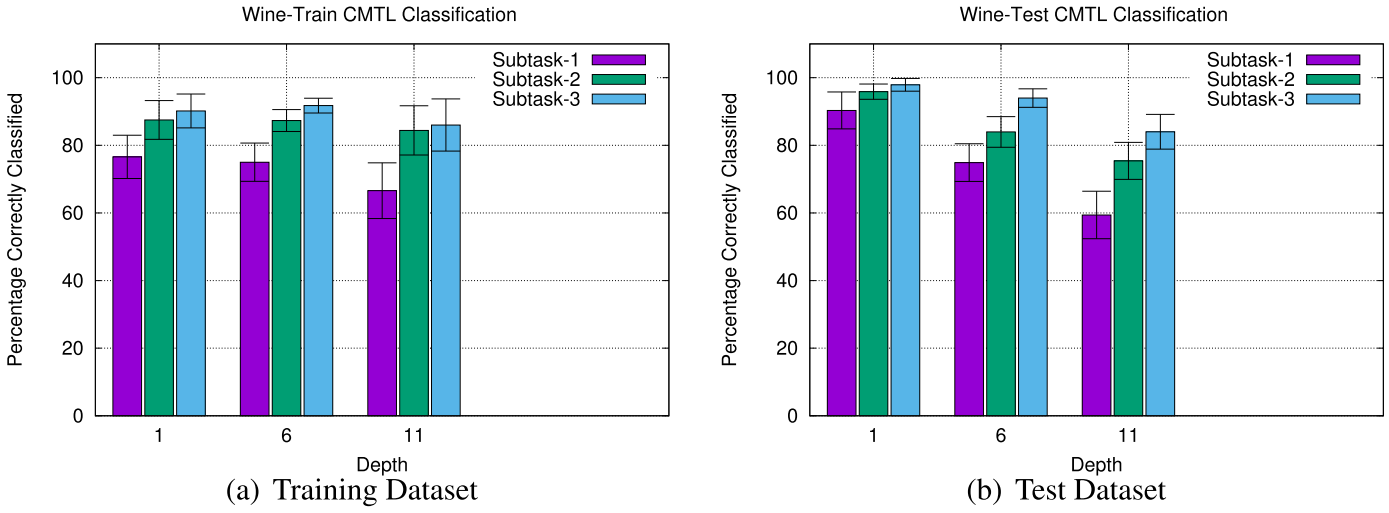

Fig. 3. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

Table 1
Configuration.

Dataset	Features	Classes	Instances	Hid.	Max. FE	Max. Epoch
Wine	13	3	178	8	120000	500
Iris	4	3	150	5	120000	500
Ionosphere	34	2	351	8	120000	500
Zoo	16	7	102	6	120000	300
Cancer	9	2	699	6	120000	1000
Lenses	4	3	24	5	120000	500
Balloon	4	2	20	5	120000	500
Balance	4	3	625	8	120000	200

4. Simulation and analysis

In this section, we present an experimental study that evaluates the proposed methodology that features co-evolutionary multi-task learning for modular pattern classification. We further compare the performance with related methods such as cooperative neuro-evolution (CNE), neuroevolution (NE), and backpropagation that features gradient descent (BP). We use covariance matrix adaptation evolution strategies (CMAES) as the designated evolutionary algorithm in the sub-populations of CNE and CMTL and population of NE [49]. In the respective methods, the population size is fixed as $P = 4 + \text{floor}(3 * \log(W_s))$, where W_s is the total number of weights. Since CNE also features modularity via the sub-populations, the comparision with CMTL will help in understanding knowledge representation amongst the different forms of modularity. We investigate the effect of depth of search d for a set of values, $d = [1, 6, 11]$.

4.1. Experimental design

We selected the Wine, Iris, Ionosphere, Zoo, Cancer, Lenses, Balloon, Balance datasets and the Robot navigation problem from the University of California, Irvine (UCI) Machine Learning Data repository [50]. For all examples, we define the first feature space ($\mathcal{F}_1$) to be that space containing 50% of the features. The second and third feature spaces, $\mathcal{F}_2$ and $\mathcal{F}_3$ contain 75 % and 100 % of the features, respectively. Note that the features were allocated to the given feature space based on how they were arrange in the dataset, with no adjustment as to what may be an optimal grouping of features.

Table 1 gives details of the respective data sets, including the number of features, the number of classes, and the number of instances along with the number of hidden neurons used in the FNN. Note that the number of input and output neurons in the FNN corresponds to the number of features and classes of the respective datasets. The chosen number of hidden neurons (Hid.) in Table 1 were determined in trial experiments. The termination condition is when a total of $40000 \times m$ function evaluations (FE) have been reached for the respective problems, m refers to the maximum number of subtasks.

4.2. Results

We report the classification performance for the respective datasets using 30 independent experimental runs. We first present the results for modular pattern classification using CMTL where we investigate the effect of depth of evolution d for the respective subtasks (θ_1, θ_2, θ_3). The results are shown in Figs. 3–9, where each figure corresponds to a different dataset. The figures contain the percentage of observations that are correctly classified in the training data, (left panel), and in the test data (right panel). Within each panel, we report the impact of increasing the number of subtasks. As expected, the percentage of observations correctly classified is higher for the training data than the test data. The classification performance increases as we increase the number of subtasks. The mean and 95% confidence interval of the classification performance for 30 experimental runs is reported as histogram with error bars for the respective scenarios (Figs. 3–9).

The results show that Zoo, Cancer, and Ionosphere problems do not have much effect from the varied depth of evolution. The problems also do not have much effect from the different subtasks, which shows that the basic subtask (θ_1) contains all the necessary information to reach the maximum classification performance in both training and test datasets. Therefore, if a feature selection method is used, the additional feature groups in these problems would be dropped. Hence, the proposed modular pattern classifi-

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

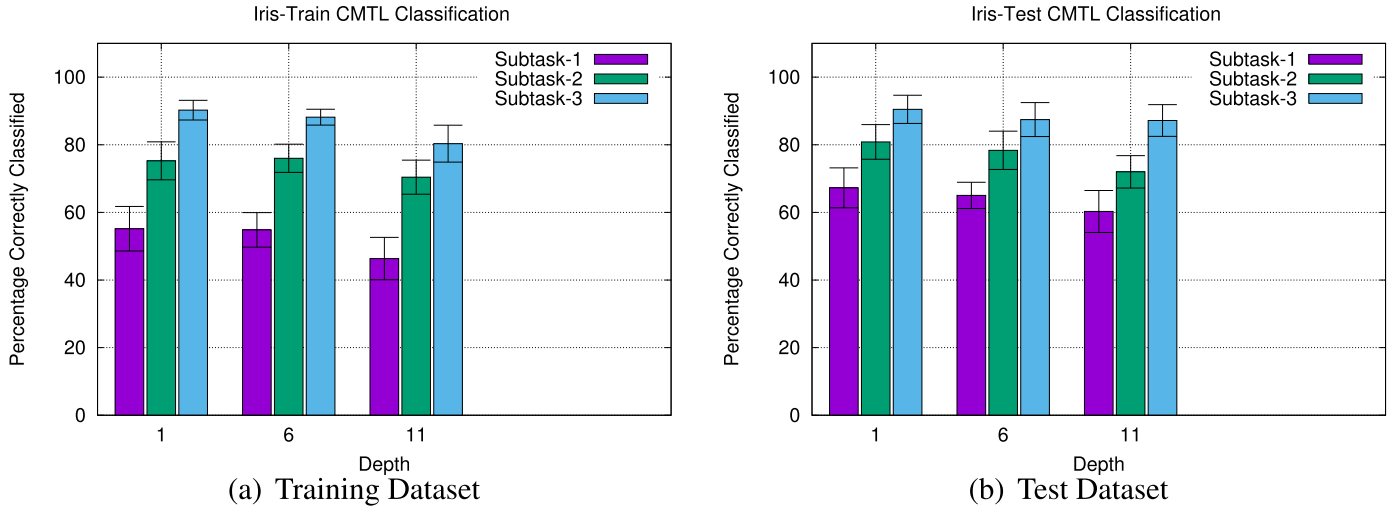

Fig. 4. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

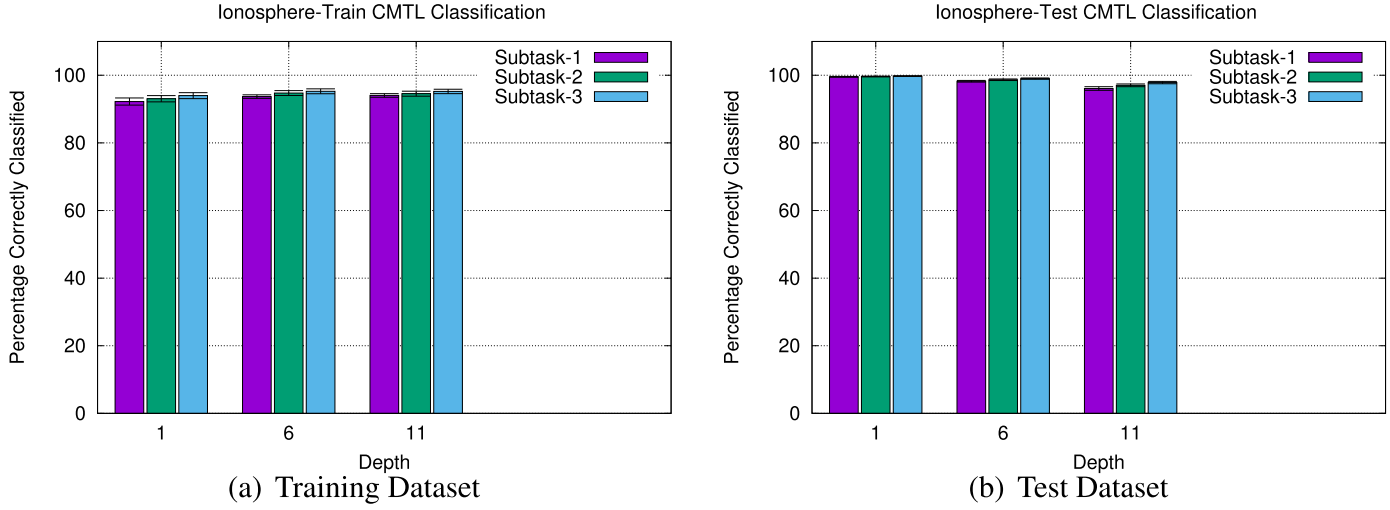

Fig. 5. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

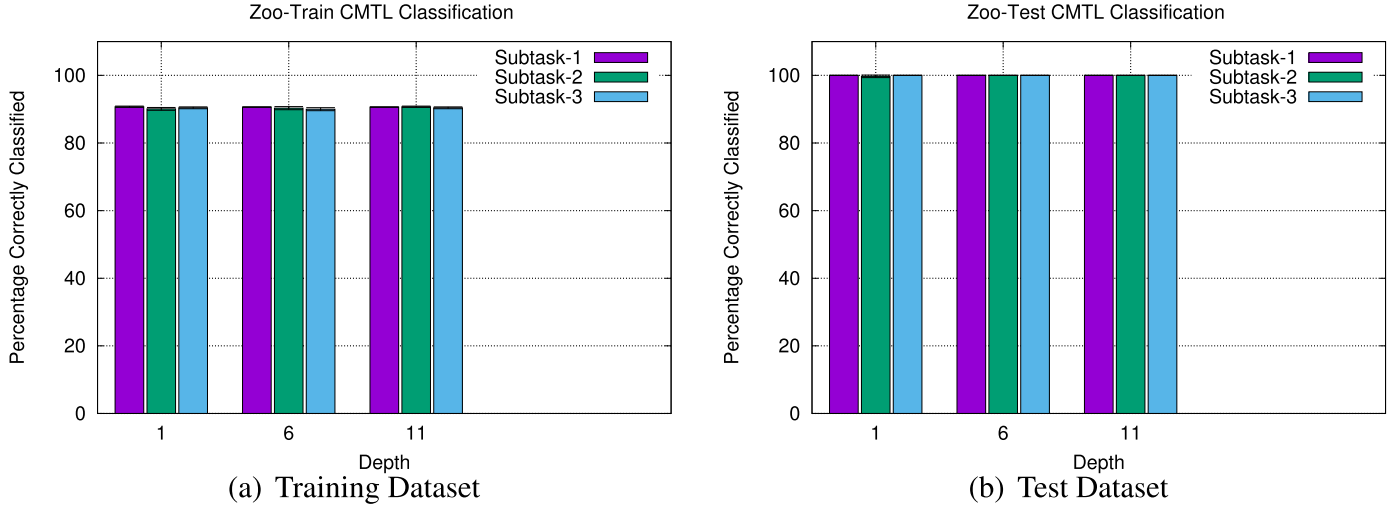

Fig. 6. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

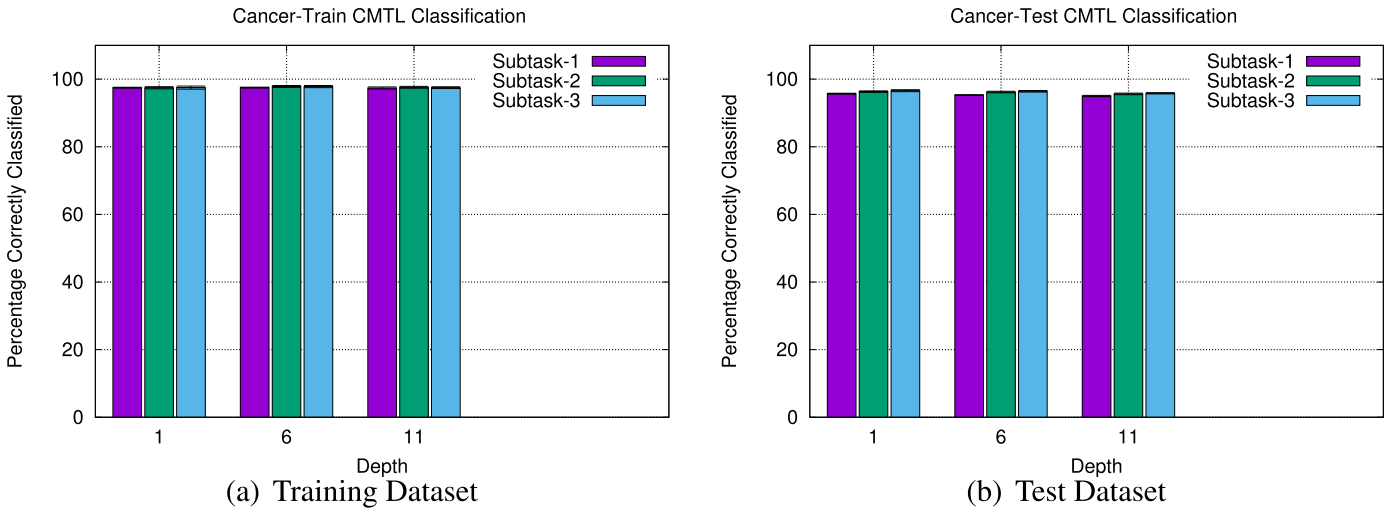

Fig. 7. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

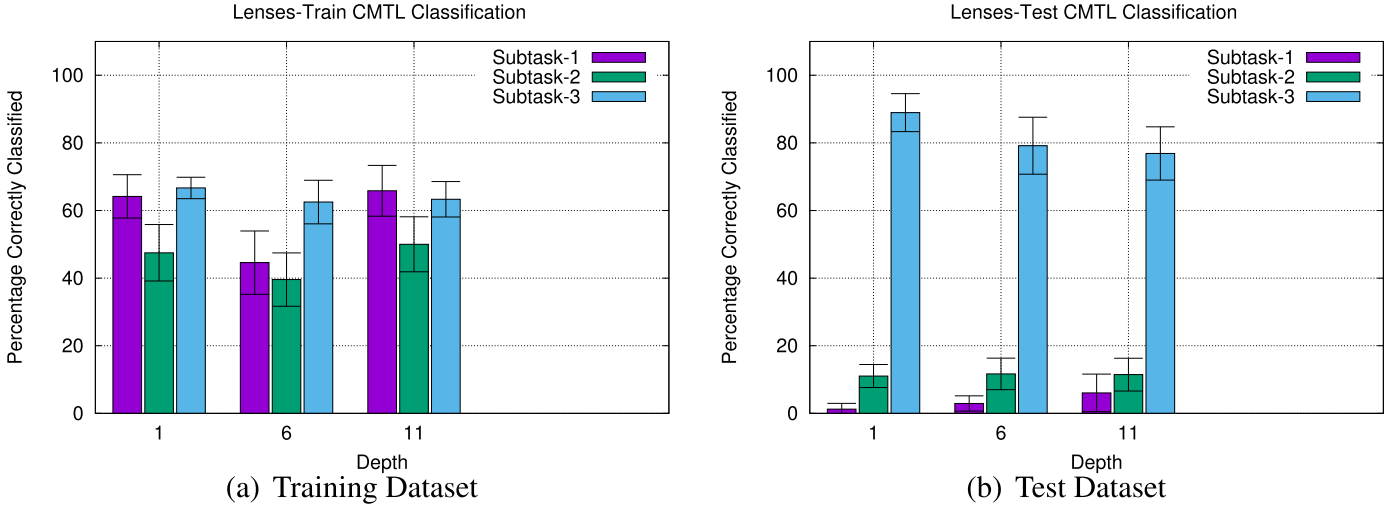

Fig. 8. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

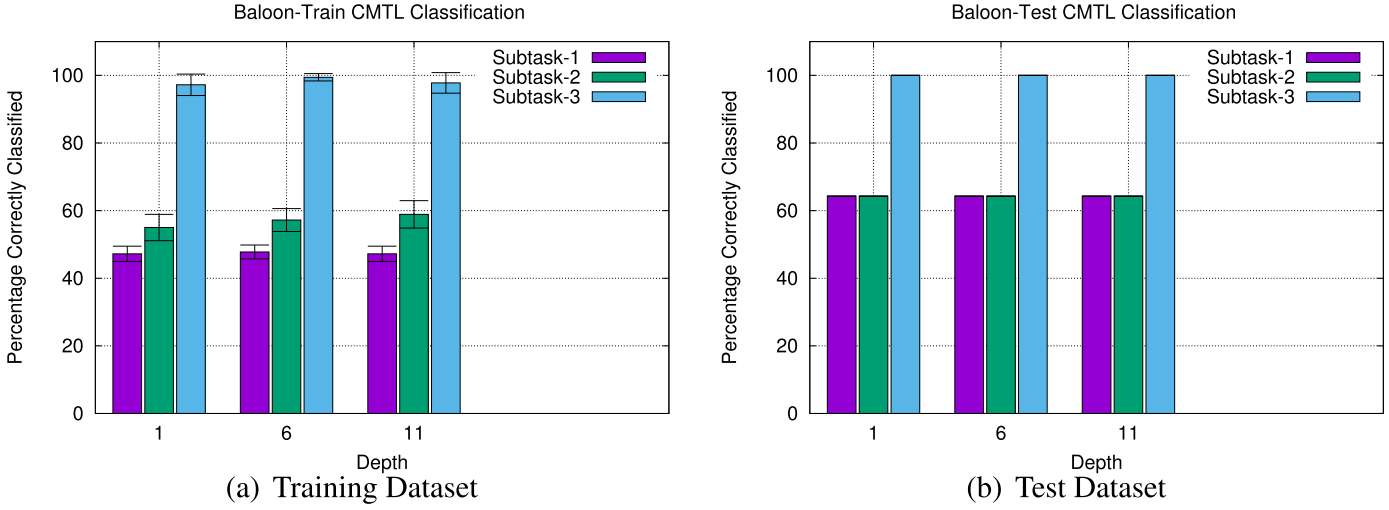

Fig. 9. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

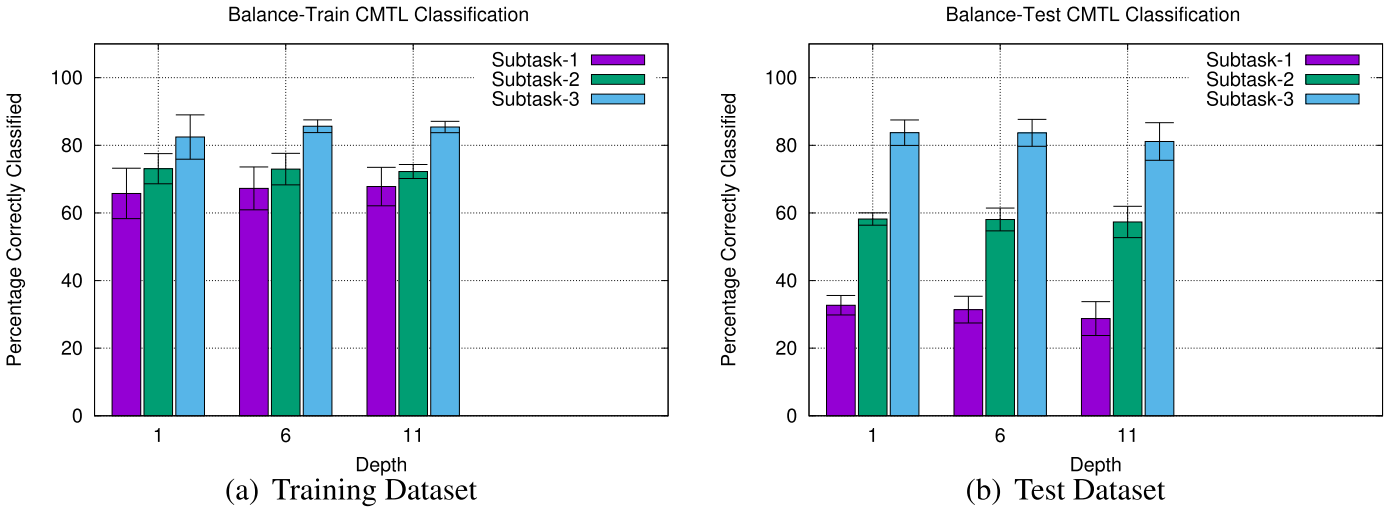

Fig. 10. Performance evaluation of CMTL depth given in terms of the number of generations (1, 6, 11) across different subtasks (θ_1, θ_2, θ_3).

Table 2
Comparison with related methods.

Problem	Dataset	BP	NE	CNE	CMTL-θ_1	CMTL-θ_2	CMTL- θ_3
Wine	Train	97.25 ± 0.34	66.43 ± 10.10	75.00 ± 7.58	75.00 ± 5.66	87.33 ± 3.25	91.75 ± 2.19
	Test	95.25 ± 2.91	74.92 ± 9.71	86.17 ± 3.55	74.88 ± 5.57	83.96 ± 4.54	93.99 ± 2.74
Iris	Train	92.15 ±0.82	69.79 ± 8.49	91.39 ± 2.91	54.83 ± 5.11	76.00 ± 4.17	88.17 ± 2.34
	Test	83.67 ±1.79	71.00 ± 8.08	90.00 ± 2.27	65.03 ± 3.88	78.36 ± 5.66	87.45 ± 5.02
Ionosphere	Train	95.18 ±1.33	95.06 ± 0.60	90.44 ± 1.14	93.70 ± 0.52	94.74 ± 0.73	95.26 ± 0.73
	Test	83.33 ±2.53	94.65 ± 1.04	90.73 ± 1.80	98.23 ± 0.24	98.69 ± 0.23	99.01 ± 0.19
Zoo	Train	99.72 ± 0.56	100.00 ± 0.00	100.00 ± 0.00	90.63 ± 0.00	90.31 ± 0.45	90.00 ± 0.45
	Test	98.54 ± 3.30	90.42 ± 0.50	91.25 ± 0.80	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Cancer	Train	93.46 ± 0.38	95.45 ± 0.29	91.60 ± 0.90	97.52 ± 0.13	97.83 ± 0.29	97.81 ± 0.33
	Test	96.22 ± 0.64	97.13 ± 0.51	95.10 ± 0.63	95.30 ± 0.12	96.20 ± 0.23	96.41 ± 0.23
Lenses	Train	97.50 ±5.25	55.15 ± 3.62	52.62 ± 2.87	44.58 ± 9.37	39.58 ± 7.89	62.50 ± 6.43
	Test	85.83 ±5.33	67.67 ± 1.06	62.75 ± 2.26	2.92 ± 2.25	11.67 ± 4.65	79.17 ± 8.42
Balloon	Train	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	47.78 ± 2.06	57.22 ± 3.39	99.44 ± 1.09
	Test	90.00 ± 15.27	97.78 ± 3.03	95.00 ± 3.88	64.29 ± 0.00	64.29 ± 0.00	100.00 ± 0.00
Balance	Train	95.56 ±0.90	85.20 ± 3.61	84.46 ± 1.55	67.26 ± 6.33	72.97 ± 4.67	85.66 ± 1.88
	Test	99.89 ± 0.29	87.30 ± 1.05	86.27 ± 1.06	31.40 ± 3.95	58.06 ± 3.37	83.69 ± 3.98

cation method can be seen as a way to feature selection and machine learning in parallel. The rest of the problems seem to follow a certain pattern where the classification performance improves from subtasks θ_1 to θ_3. This is because more feature groups for learning the respective feature space are added as the subtask increases. The only peculiar trend is in the training performance of the Lenses dataset where subtask θ_2 has lower classification performance than subtask θ_1. Later in θ_3, the performance increases. As we review the generalization performance in the test dataset, we find that there is a very large increase in the performance in subtask θ_3 which indicates that its correspondence feature group is important for the particular subtask. The importance of the third feature group in subtask θ_3 can also be observed for the Balloon dataset where the effect is visible both for the training and test datasets. The largest depth of evolution (11 generations) seems to give a lower quality performance for the Iris and Wine problems.

Table 2 compares the performances of CMTL for the different number of subtasks with respective methods (BP, NE, and CNE) on both the training and test datasets. Note that the CMTL subtasks ($\theta_1 - \theta_3$) with a depth of evolution of 6 generations is used for comparison.

Table 2 shows that, for the Ionosphere, Zoo, and Cancer datasets, the percentage of correctly classified observations is relatively independent of the number of subtasks used, indicating that the first feature group contains those features which account for most of the variability in the outcome variable. However, for the Wine, Ionosphere, Zoo, Balloon and most notably the Lenses datasets, the inclusion of additional subtasks improves the classification performance.

Overall, it would be justifiable only to compare CMTL-subtask θ_3 with CNE and NE since they use all the features. Hence, we observe that CMTL-subtask θ_3 gives the best generalization (test) performance for Wine, Ionosphere, Zoo, Lenses and Balloon datasets. In the rest, i.e Iris, Cancer, and Balance datasets, CMTL performance is comparable to either CNE or NE. We note that in general, NE gives the weakest performance when compared to CNE and CMTL. Similar trends can also be observed for the training performance. Overall, we can infer that CMTL can perform similar or better when compared to the non-modular pattern classification methods (NE and CNE).

In the comparison of *BP* with CMTL-θ_3, we observe that CMTL-θ_3 gives much better generalization performance apart from the Lenses and the Balance problem. The difference in generalization performance by the two strategies could be due to overfitting or underfitting as defined by the training performance. Moreover, multi-task and developmental learning characteristics from CMTL could have contributed to improved generalization performance in

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

a certain way, howsoever, further analysis would be needed to have a better understanding.

5. Discussion

We note that CMTL was proposed in previous work specifically targeted to dynamic time series problems [27]. Dynamic time series was identified as a category of robust time series prediction that required decision making given varied set of input features or observations. Hence, it was formulated as a multi-task learning problem and CMTL was propose for training neural networks that retained modularity for robust prediction. Such problems emerge in real-world applications such as cyclone prediction, where robust prediction is needed in cases when limited data or observations are available. Moreover, CMTL was also used for multi-step time series prediction [28]. The method presented in this paper borrows the aspect of modularity and incorporates it with knowledge representation and decision making for pattern classification problems. Hence, the contribution of this paper is mainly towards problem decomposition for modular pattern classification.

In CMTL, the depth of evolution evaluated the inter-dependencies among the groups of features utilised as building blocks for the subtasks. In some problems (Zoo, Cancer and Ionosphere), we observed that there is not much effect across the different subtasks when considering the different values for the depth of evolution. These problems are those that can be decomposed easily, often called separable problems as they do not have inter-dependencies amongst the subtasks. In the cooperative neuro-evolution literature, the effect of inter-dependencies amongst the knowledge modules implemented as sub-populations has been experimentally studied [22,45]. Therefore, given a high level of inter-dependencies amongst the sub-populations, lower depth of search (eg. $d = 1$ generation) is the most appropriate. Problem decomposition for partially separable problems has also been a major challenge for large-scale and black-box global optimization problems [51–53]. We further note that the knowledge module from subtask 1 is independent or stands on its own to make a decision. However, other subtasks use it as building blocks of knowledge or are fully dependent for decision making. This is a typical dynamic programming approach, whereas CNE features modularity through a divide and conquer approach. There are two different forms of inter-dependencies in CMTL, one in the data or feature group space, and the other in the weight space implemented as knowledge modules.

We note that with higher values for the depth of evolution in CMTL, there is faster convergence as a lower depth of search requires more computational load in transferring knowledge from one subtask to another. The transfer of knowledge takes place after CMTL evolves all the sub-populations for the depth of evolution in a round-robin fashion. Once this is done, then the knowledge from the foundational subtasks are transferred to bigger subtasks. The higher the depth of evolution, the more time is required for the transfer of knowledge. Hence, there needs to be a balance between the depth of evolution and transfer of knowledge for different types of problems depending on the inter-dependencies amongst the features and the feature weighting or redundancy and relevance for the overall problem. The depth of evolution of 6 generations has been ideal for the problems studied. The results show that they have close performance when compared to depth of search of 1 generation. Howsoever, there will be a higher frequency of transfer of knowledge with lower depth of search which would take a further computational load. At this stage, it would be reasonable to justify that feature section methods [54,55] in a prior stage of the proposed work could help in depending better subtasks that consider interdependencies, relevance and importance. This has not been done in this study and could be useful for real-world applications. Note that in most of the cases of the results given by backpropgation, the classification results are generally better for the training dataset when compared to the test dataset. This is not observed for most of the cases of CMTL as the training performance is slightly poorer. A reason could be due to the way the algorithm develops knowledge from building blocks of knowledge. In cases where there is more emphasis on the earlier subtasks, there seems to be a slight difficulty in learning when the rest of the subtasks are presented for training. This could be due to knowledge developed when learning from incomplete information and the update of knowledge when further information is presented.

Developmental learning feature of the proposed method has been motivated by biological neural systems and perception and cognitive learning and adaptive behaviour. The human eye has a similar vision system as given by the sensors (S1 to S4) of the robot navigation example in Fig. 1. In very extreme situations, we rely on information from the peripheral vision which is a part of the vision that occurs outside the very center of gaze to make a decision [56,57]. Mostly, we focus our attention or gaze to the frontal visual system. In some animals such as horses, sudden change experienced through visual system can be damaging to their attention, especially while on a journey. This is due to the way their eyes are positioned which has more side visual view when compared to humans. This is why traditionally vision guides have been used for horses that pulled carts. Similar ways of attention and focus can be used for auditory systems and would be helpful for advanced speech recognition systems. This is especially when one needs to give attention to a specific voice in a noisy and dynamic environment. We naturally adjust our hearing to everyday situations when some parts of sensory inputs are either unavailable or are too noisy as trying to understand what someone is saying in an environment with sudden background noise. In humanoid robotics applications, such development of cognitive systems that need to be dynamic and have features of modularity will be beneficial. Therefore, this study can further motivate to develop systems that need to autonomously focus their attention at different groups of input features in real-time scenarios when certain feature inputs are unavailable or contain noise.

The problem of “missing values” in a dataset intensifies for the problems with availability of limited data especially in the applications where data is scarce or cannot be reproduced [58]. If we consider the proposed method to tackle the problem of “missing values” in a limited dataset problem, the major constraint is that the proposed method needs all the data from the foundational subtasks during training. The additional feature groups could have missing values. Note that the proposed method considers knowledge from previous subtasks as building blocks for decision making. Hence, the proposed method must be adapted in future work where this property does not hold. Hence, in order to develop a variant of the proposed method for “missing” values in the training data, the final decision-making process of the neural network should not be dependent on knowledge from previous subtasks. Howsoever, modularity in knowledge representation would be needed. A Bayesian approach can become helpful when the missing features in the base subtask can be drawn from a probability distribution [59].

A limitation of the algorithm is timely convergence since it is based on neuroevolution which is not a gradient-based learning method. Hence, big data problems could not be considered in this work. Howsoever, there is scope for future work that will derive a gradient-based learning method that will have much faster convergence and the method would be applied to big data problems. Moreover, the way the problem is decomposed into the feature groups that make up the feature space and the subtasks would be of interest to the future investigation. The proposed method

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

can be seen as a new approach to pattern classification that provides a synergy for feature selection and machine learning. It can be applied to datasets with a large number missing values or attributes. Transfer learning and heterogeneous domain adaptation are some of the current challenges in data science and the proposed methods could be extended for these subtasks. In this case, the subtasks identified could be seen as different domains or sources of data with different set of features which some are relevant [60].

6. Conclusions and future work

We extended a coevolutionary multi-task learning algorithm for modular pattern classification that featured concepts from developmental and multi-task learning. The problem was formulated by decomposition of subtasks through specific feature groups for incremental knowledge development. The goal was to have a certain level of modularity in pattern classification problems while ensuring that the quality of decision making in classification performance was not substantially deteriorated. We conducted experiments with pattern classification datasets and investigated the effects of combining different feature groups into feature space that defined the subtasks. The results showed that the proposed method produced similar results when compared to related methods. In certain problems, the proposed method improved the performance. The proposed method provides a way for learning while evaluating the feature groups which is helpful to quantify feature contribution.

In future work, it would be worthwhile to apply the proposed modular pattern classification approach for decision making for selected autonomous systems. Moreover, the development of gradient-based learning can further speed up the learning process and the approach could be used for other neural network architectures such as recurrent neural networks. There is also scope for heterogeneous transfer learning and domain adaptation. Furthermore, an extension of the proposed approach through Bayesian inference would provide a principled way for uncertainty quantification in the decision making process.

References

[1] D. Meunier, R. Lambiotte, A. Fornito, K.D. Ersche, E.T. Bullmore, Hierarchical modularity in human brain functional networks, Front. Neuroinform. 3 (2009). https://doi.org/10.3389/neuro.11.037.2009.
[2] C. Nicolini, A. Bifone, Modular structure of brain functional networks: breaking the resolution limit by surprise, Sci. Rep. 6 (2016). https://doi.org/10.1038/srep19250.
[3] C. Prince, N. Helder, G. Hollich, Ongoing emergence: a core concept in epigenetic robotics, in: Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, 2005, pp. 63–70.
[4] A.F. Morse, J. De Greeff, T. Belpeame, A. Cangelosi, Epigenetic robotics architecture (Era), IEEE Trans. Auton. Mental Develop. 2 (4) (2010) 325–339.
[5] N. Geschwind, P. Behan, Left-handedness: Association with immune disease, migraine, and developmental learning disorder, Proc. Nat. Acad. Sci. 79 (16) (1982) 5097–5100.
[6] M.H. Lee, Q. Meng, F. Chao, Developmental learning for autonomous robots, Robot. Auton. Syst. 55 (9) (2007) 750–759.
[7] A. Arsenic, Developmental learning on a humanoid robot, in: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks, 4, IEEE, 2004, pp. 3167–3172.
[8] M.K. Johnson, A multiple-entry, modular memory system, in: Psychology of Learning and Motivation, 17, Academic Press, 1983, pp. 81–123. https://doi.org/10.1016/S0079-7421(08)60097-3.
[9] D.G. Stork, Is backpropagation biologically plausible? in: Neural Networks, 1989. IJCNN., International Joint Conference on, 1989, pp. 241–246vol.2, doi:10.1109/IJCNN.1989.118705.
[10] J. Yang, P. Li, Brain networks of explicit and implicit learning, PLOS ONE 7 (8) (2012) 1–9, doi:10.1371/journal.pone.0042993.
[11] D.D. Cox, T. Dean, Neural networks and neuroscience-inspired computer vision, Current Biol. 24 (18) (2014) R921–R929. https://doi.org/10.1016/j.cub.2014.08.026.
[12] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[13] J.L. Elman, Finding structure in time, Cognit. Sci. 14 (1990) 179–211.
[14] M.J. Pazzani, Influence of prior knowledge on concept acquisition: Experimental and computational results., J. Exper. Psychol. Learn. Memory Cognit. 17 (3) (1991) 416.
[15] J.L. Elman, Learning and development in neural networks: the importance of starting small, Cognition 48 (1) (1993) 71–99.
[16] J.L. McClelland, B.L. McNaughton, R.C. O'Reilly, Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory., Psycholog. Rev. 102 (3) (1995) 419.
[17] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[18] 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.
[19] 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, in: ICML '09, ACM, New York, NY, USA, 2009, pp. 137–144, doi:10.1145/1553374.1553392.
[20] 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.
[21] 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, Lecture Notes in Computer Science, 866, Springer Berlin Heidelberg, 1994, pp. 249–257.
[22] R. Chandra, M. Frean, M. Zhang, On the issue of separability for problem decomposition in cooperative neuro-evolution, Neurocomputing 87 (2012) 33–40.
[23] M.A. Potter, K.A. De Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, Evol. Comput. 8 (2000) 1–29.
[24] N. García-Pedrajas, D. Ortiz-Boyer, A cooperative constructive method for neural networks for pattern recognition, Pattern Recogn. 40 (1) (2007) 80–98.
[25] F. Gomez, R. Mikkulainen, Incremental evolution of complex general behavior, Adapt. Behav. 5 (3–4) (1997) 317–342. https://doi.org/10.1177/105971239700500305.
[26] 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.
[27] R. Chandra, Y.S. Ong, C.K. Goh, Co-evolutionary multi-task learning for dynamic time series prediction, Appl. Soft Comput. 70 (2018) 576–589.
[28] R. Chandra, Y.S. Ong, C.K. Goh, Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction, Neurocomputing 243 (2017) 21–34.
[29] R. Little, et al., Calibrated Bayes, for statistics in general, and missing data in particular, Stat. Sci. 26 (2) (2011) 162–174.
[30] B.P. Carlin, T.A. Louis, Bayes and empirical Bayes methods for data analysis, Stat. Comput. 7 (2) (1997) 153–154.
[31] Z. Ghahramani, M.I. Jordan, Learning from incomplete data, Artificial Intelligence Laboratory, MIT, 1994, pp. 1–11. A.I. Memo No. 1509.
[32] G. Haffari, Y. Wang, S. Wang, G. Mori, F. Jiao, Boosting with incomplete information, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 368–375.
[33] M.A. Potter, K.A. De Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, Evol. Comput. 8 (1) (2000) 1–29. https://doi.org/10.1162/106365600568086.
[34] R. Chandra, Co-evolutionary multi-task learning for modular pattern classification, in: Proceedings of the 24th International Conference on Neural Information Processing, 2017, pp. 692–701.
[35] M.L. Anderson, Neural reuse: A fundamental organizational principle of the brain, Behav. Brain Sci. 33 (4) (2010) 245–266.
[36] C.I. Bargmann, Beyond the connectome: how neuromodulators shape neural circuits, Bioessays 34 (6) (2012) 458–465.
[37] C. Eliasmith, A unified approach to building and controlling spiking attractor networks, Neural Comput. 17 (6) (2005) 1276–1314.
[38] F. Donnarumma, R. Prevete, A. de Giorgio, G. Montone, G. Pezzulo, Learning programs is better than learning dynamics: a programmable neural network hierarchical architecture in a multi-task scenario, Adapt. Behav. 24 (1) (2016) 27–51.
[39] K.A. Thoroughman, R. Shadmehr, Learning of action through adaptive combination of motor primitives, Nature 407 (6805) (2000) 742.
[40] K.A. De Jong, W.M. Spears, A formal analysis of the role of multi-point crossover in genetic algorithms, Annals Math. Artif. Intel. 5 (1) (1992) 1–26.
[41] D. Ashlock, S. Willson, N. Leahy, Coevolution and tartarus, in: Proceedings of the Congress on Evolutionary Computation, 2, IEEE, 2004, pp. 1618–1624.
[42] V.R. Khare, X. Yao, B. Sendhoff, Credit assignment among neurons in co-evolving populations, in: Proceedings of the International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 882–891.
[43] L. Bull, On coevolutionary genetic algorithms, Soft Comput. 5 (3) (2001) 201–207.
[44] F. Gomez, J. Schmidhuber, R. Miikkulainen, Accelerated neural evolution through cooperatively coevolved synapses, J. Mach. Learn. Res. 9 (2008) 937–965.
[45] R. Chandra, M. Frean, M. Zhang, Adapting modularity during learning in cooperative co-evolutionary recurrent neural networks, Soft Comput. Fusion Foundat. Methodol. Appl. 16 (6) (2012) 1009–1020.

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011

[46] 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.
[47] N. Garca-Pedrajas, D. Ortiz-Boyer, C. Hervas-Martne, Cooperative coevolution of generalized multi-layer perceptrons, Neurocomputing 56 (2004) 257–283. https://doi.org/10.1016/j.neucom.2003.09.004.
[48] N. Garca-Pedrajas, C. Hervas-Martinez, J. Munoz-Perez, Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks), Neural Networks 15 (2002) 1259–1278.
[49] 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.
[50] A. Asuncion, D. Newman, UCI machine learning repository, 2007.
[51] Y. Mei, M.N. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2) (2016) 13:1–13:24, doi:10.1145/2791291.
[52] K.K. Bali, R. Chandra, Scaling up multi-island competitive cooperative coevolution for real parameter global optimisation, in: Proceedings of the Advances in Artificial Intelligence - 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30 - December 4, 2015, Proceedings, 2015, pp. 34–48, doi:10.1007/978-3-319-26350-2_4.
[53] R. Salomon, Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms, Biosystems 39 (3) (1996) 263–278.
[54] M. Dash, H. Liu, Feature selection for classification, Intel. Data Anal. 1 (3) (1997) 131–156.
[55] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.
[56] J.M. Henderson, Human gaze control during real-world scene perception, Trends Cognit. Sci. 7 (11) (2003) 498–504. https://doi.org/10.1016/j.tics.2003.09.006.
[57] A. Burgess, R. Wagner, R. Jennings, H.B. Barlow, Efficiency of human visual signal discrimination, Science 214 (4516) (1981) 93–94.
[58] R.J. Little, D.B. Rubin, The analysis of social science data with missing values, Sociolog. Methods Res. 18 (2–3) (1989) 292–326.
[59] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach. Learn. 9 (4) (1992) 309–347.
[60] 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.

Dr. 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) from the University of the South Pacific. Dr. Rohitash Chandra is currently USyd Research Fellow at the Centre for Translational Data Science and School of Geosciences at the University of Sydney. His research interests are in areas of deep learning, neuro-evolution, Bayesian methods, solid Earth Evolution, reef modelling and mineral exploration. Currently, he is developing novel learning algorithms for robust and dynamic decision making given misinformation and uncertainty from the environment. This provides a synergy of deep learning methods with Bayesian inference and multi-task learning. Furthermore, he is involved in projects that employ machine learning methods and Bayesian inference via parallel tempering for solid Earth evolution, mineral exploration, and reef modelling.

Prof. Sally Cripps is a Co-Director at the Centre for Translational Data Science, University of Sydney. She is also Professor in Statistics at the School of Mathematics and Statistics, University of Sydney. Her research interests include areas of Bayesian inference and computational statistics with applications to Earth and environmental sciences.

Please cite this article as: R. Chandra, S. Cripps, Coevolutionary multi-task learning for feature-based modular pattern classification, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.08.011