# Co-evolutionary multi-task learning for modular pattern classification

Rohitash Chandra

Centre for Translational Data Science
The University of Sydney, NSW 2006, Australia.
rohitash.chandra@sydney.edu.au

**Abstract.** Modularity in the learning process is a means by which effective decision making can be maintained when some of the input features are missing. In this paper, co-evolutionary multi-task learning algorithm is used for pattern classification which is robust to situations when some input features are unavailable during the deployment stage of decision support or control systems. The main feature of the algorithm is the ability to make decisions with some degree of error given misinformation. The results show that the method produces results comparable to non-modular methods while having modular features for dynamic and robust pattern classification

**Keywords:** Cooperative coevolution, neuroevolution, multi-task learning, modular pattern classification.

## 1 Introduction

In biological neural systems, due to a certain level of modularity in knowledge representation, the absence of a sensory input does not completely hinder the decision making process [9]. The knowledge perceived through different senses can be seen as a modular input to biological neural systems [13]. For instance, we can identify a person just by listening to their voice when we do not see their face given sudden darkness due to malfunction of light in a room. This is through the presence of modularity of knowledge representation in learning that gives effective decision making when some parts or input features are missing. Therefore, a person can make an identification through sense of hearing when visual signal is missing. We can identify a person or objects to a high degree of certainty given information from the the sense of sight and auditory information. However, we do not have complete loss of sight or hearing when we close an eye or have difficulty with one ear. This is possible through modular knowledge representation in the human brain [15].

Modular neural networks have been motivated from repeating structures in nature. They were introduced for visual recognition tasks that were trained by genetic algorithms [12]. More recently, a modular neural network was presented where the performance and connection costs were optimised through neuroevolution which achieved better performance when compared to fully connected neural networks [7] and had the feature to learn new tasks without forgetting old ones [9]. Developmental learning is an approach for learning new tasks with competence as they are encountered [10,14] which differs from conventional learning

algorithms. The goal is not only concerned with building task specific knowledge and models, but to support the growth of learning for continuous development. Multi-task learning exploits knowledge learned from related tasks through shared knowledge representation [2]. This can motivate the development of learning algorithms that can feature robust decision making given sudden misinformation from the input feature space.

Neuro-evolution refers to the use of evolutionary algorithms for training neural networks [1]. Cooperative coevolution [17] has been a prominent methodology for neuro-evolution where the neural network is decomposed into modules [18] that are implemented as sub-populations. This is also known as cooperative neuro-evolution [3]. The features of multi-task and ensemble learning have been incorporated to develop co-evolutionary multi-task learning algorithm to address dynamic time series problems [4] and multi-step ahead prediction [5].

In this paper, co-evolutionary multi-task learning is used for modular pattern classification where a group of features are used as building blocks of knowledge. This tackles dynamic and modular pattern classification using notions from multi-task and developmental learning. The effectiveness of the method is demonstrated using benchmark classification datasets.

The rest of the paper is organised as follows. Section 2 gives details of proposed method and problem formulation while Section 3 presents the results and discussion. Section 4 concludes the paper with directions for future research.

## 2 Modular Pattern Classification

We give an overview for the motivation for modular pattern classification using the case for a robot navigation task as shown in Figure 1. Consider a robot navigation problem that depends on input sensors that are controlled through a neural network [16]. In this case, if the robot is moving forward, the two front sensors (S1 and S2 in Figure 1) contributes the most. The goal is to develop a dynamic control system that can withstand malfunction or misinformation from the sensors at certain time. Howsoever, in a conventional classification-based control system, this could pose a serious problem if there is a sudden disruption of information from sensors. Therefore, a dynamic and robust control is needed that can guide the robot further given sudden misinformation from the respective sensors.

### 2.1 Co-evolutionary Multi-task Learning

We present modular pattern classification from the perspective of data that can be decomposed as feature groups with overlapping features, known as subtasks. A feature group, $X_m$, is a subset of features. The subtasks $\Omega_1, \ldots, \Omega_m$ are defined as the union of selected feature groups $X_1, \ldots X_m$, for $m = 1, \ldots, M$, where $M$ is the total number of feature groups, so that;

$$\begin{aligned} \Omega_1 &= [X_1] \\ \Omega_2 &= [X_1, X_2] \\ \Omega_3 &= [X_1, X_2, X_3] \\ \Omega_M &= [X_1, X_2, ..., X_M] \end{aligned} \tag{1}$$

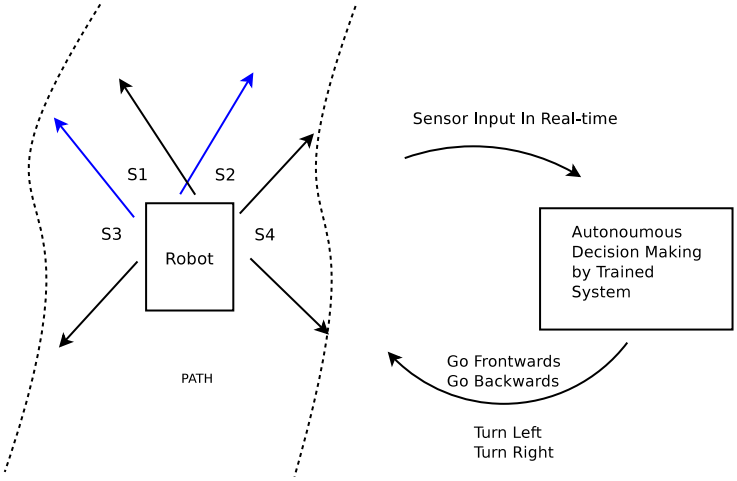


Fig. 1: Control problem for autonomous robots based on sensor input (S1, S2, S3, and S4). The decision making can be done through a control system that features modular pattern classification.

In the example of the robot navigation problem in Figure 1, the first feature group is defined to be $X_1 = \{S1, S2\}$ and the second to be $X_2 = \{S3, S4\}$. The overlapping subtasks are then $\Omega_1 = X_1$ and $\Omega_2 = \{X_1, X_2\}$. The input-hidden layer $\omega_m$ weights and the hidden-output layer $\upsilon_m$ weights are combined for the respective network module $\Phi_m$ as shown in Figure 2. The base module is given as $\Phi_1 = [\omega_1, \upsilon_1]$. Let $\theta_1 = \Phi_1$ be the set of weights in the base network module $\Phi_1$ which takes the features in $\Omega_1$ as input. Note that the cascaded network module $\theta_m$ of subtask $m$ is constructed by combining with current $\Phi_m$ and previous network module $\Phi_{m-1}$ as follows.

$$\begin{aligned} \Phi_1 &= [\omega_1, \upsilon_1]; \;\; \theta_1 = (\Phi_1) \\ \Phi_2 &= [\omega_2, \upsilon_2]; \;\; \theta_2 = [\theta_1, \Phi_2] \\ &\vdots \\ \Phi_M &= [\omega_M, \upsilon_M]; \;\; \theta_M = [\theta_{M-1}, \Phi_M] \end{aligned} \tag{2}$$

The list of network modules considered for training or optimisation is therefore $\mathbf{\Phi} = (\Phi_1, \ldots, \Phi_M)$.

The learning problem at hand is in optimising the respective knowledge modules for the given subtasks. Therefore, co-evolutionary multi-task learning (CMTL) initially proposed for dynamic time series is adapted for modular pattern classification [4]. The algorithm involves optimisation of sub-populations

**Alg. 1** Co-evolutionary multi-task learning

**Data**: Feature space $X_m$ and response $Y$ from data.
**Result**: Knowledge modules (weights) $\theta_m$ for the respective subtasks $\Omega_m$.
initialization
**for** *each subtask* $\Omega_m$ **do**
    1. Define different sub-populations $S_m$ using network module $\Phi_m$
    2. Initialize the values in sub-populations $S_m$
**end**
**while** *each phase until termination* **do**
    **for** *each sub-population* $S_m$ **do**
        **for** *each generation until depth* $\beta$ **do**
            **for** *each i individual in sub-population* $S_{m_{ij}}$ **do**
                **for** *each j in individual* $S_{m_{ij}}$ **do**
                    Assign individual $V_m = S_{m_{ij}}$
                **end**
                **if** $m == 1$ **then**
                    1. $Z_m = [V_m]$
                    2. Fitness evaluation by encoding $Z_m$ in cascaded network module $\theta_m$ given in Figure 2
                **end**
                **else**
                    1. Append to best individual $B_{m-1}$ of previous sub-population: $Z_m = [V_m, B_{m-1}]$
                    2. Fitness evaluation by encoding $Z_m$ in cascaded network module $\theta_m$ given in Figure 2
                **end**
            **end**
            **for** *each i individual in sub-population* $S_{m_{ij}}$ **do**
                * Select and create new offspring via evolutionary operators:
                1. Selection, 2. Crossover, and 3. Mutation
            **end**
        **end**
    **end**
**end**
**for** *each subtask* $\Omega_m$ **do**
    1. Get best solution $B_m$ from sub-population $S_m$
    2. Load test data for the respective cascaded network module $\theta_m$
    3. Report classification performance for the subtask
**end**

that feature a pool of solutions called individuals that consist of weights and biases for the respective modules. The decomposed modules in a cascaded network shown in Figure 2 are implemented as sub-populations. Algorithm 1 gives further details of the co-evolutionary multi-task learning algorithm. Each sub-population is given as $S_1, S_2, ..S_N$, where $N$ is number of sub-populations. The algorithm begins by creating sub-populations $S_{m_{ij}}$ that map the respective network modules

$\Phi_m$ for optimisation. $i$ refers to the individuals in the respective sub-population and $j$ refers to each element in the individual. The algorithm initialises the sub-populations by drawing from uniform distribution $U(-\alpha, \alpha)$ where $\alpha$ defines the range. Afterwards, the algorithm moves into the evolution phase where each sub-population is evolved for a fixed number for generations defined by the depth of evolution $\beta$. Here, each individual from a sub-population is assigned to a vector $V_m = S_{m_{ij}}$, that is further concatenated by the best individual $B_{m-1}$ from previous sub-population $Z_m = [V_m, B_{m-1}]$. Evaluation of $Z_m$ is done by encoding it to the respective neural network module given the reference to the subtask $m$ as shown in Figure 2. Note that given base case ( where $m == 1$), there is no concatenation from previous subtasks, therefore $Z_{m_j} = [V_{m_j}]$. Moving on, the loss is computed which is used as the fitness of the given individual. This is used further for creating new individuals with operators such as selection, crossover, and mutation that make the basis for co-evolutionary algorithms [6]. This procedure is done for every individual $i$ in the sub-population, $S_m$. All the sub-populations are evaluated and evolved with evolutionary operators such as selection, crossover, and mutation. Note that a phase is complete when all the subtask modules have been evolved for $n$ generations. This is repeated until the termination condition for evolution or optimisation is reached. The termination condition can be either the maximum number of function evaluations or a given fitness value (loss) from the training or validation dataset. Once the termination criteria has been reached, the algorithm moves into the testing phase where the best individuals from the respective sub-populations are mapped into the cascaded network. The classification performance for each subtask is then reported. The implementation code in Matlab has been given online [1].

## 3 Simulation and Analysis

In this section, CMTL is compared with conventional neuro-evolution methods such as cooperative neuro-evolution (CNE) and evolutionary algorithms (EA). The covariance matrix adaptation evolution strategies (CMAES) is used as the designated evolutionary algorithm in all the respective methods [11].

### 3.1 Experimental Design

The Wine, Iris, Ionosphere, Zoo, Cancer, Lenses problem are selected from the University of California, Irvine (UCI) Machine Learning Data repository [**?**]. The subtasks in CMTL are defined by a set of features which must be defined beforehand. In these experiments, the features are selected consecutively in portions that are defined as follows. Subtask one $\Omega_1$ contains first 50 % of the features, while subtasks two and three ($\Omega_2$ and $\Omega_3$) contain 75 % and 100 %, respectively. Table 1 gives details of the respective problems used with details about the number of features, number of classes, and number of instances used. Sigmoid units in hidden and output layers are used for all the problems. The number of hidden neurons for foundation module that corresponds to $\Omega_1$ is given by $\hat{h}$ in Table 1. The rest of hidden neurons in the respective network modules are incremented

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

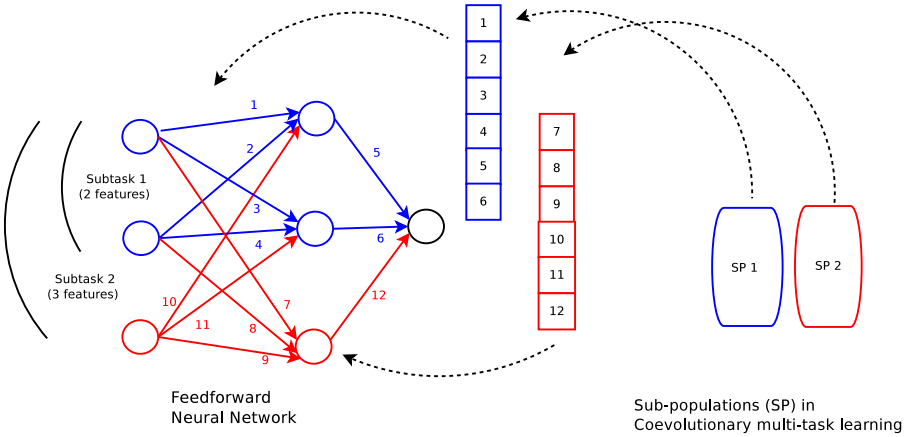


Fig. 2: This cascaded modular neural network which can also be viewed and implemented as an ensemble of neural networks. The colours associated with the synapses in the network are linked to their encoding that are given as different modules. Subtask 1 employs a network topology with 2 hidden neurons while the rest of the modules add extra input and hidden neurons.

by $h_n = \hat{h} + h_{n-1} + \epsilon$, given that $h_1 = \hat{h}$. $\epsilon = 2$ is used in all the experiments. The termination condition is when a total of 120 000 $\times M$ function evaluations have been reached for the respective problems where $M$ represents maximum number of subtasks. Note that all the sub-populations evolve for the same depth of search. The population size of the CMAES in all the respective methods is given by $P = 4 + floor(3 * log(\gamma))$, where $\gamma$ is the total number of weights and biases in the cascaded network architecture. A fixed depth of evolution of 6 generations is used in CMTL that was obtained from trail experiments. Note that all the sub-populations in CMTL and CNE evolve for the same depth of search.

Table 1: Configuration

| Dataset | Features | Classes | Instances | $\hat{h}$ |
|---|---|---|---|---|
| Wine | 13 | 3 | 178 | 8 |
| Iris | 4 | 3 | 150 | 5 |
| Ionosphere | 34 | 2 | 351 | 8 |
| Zoo | 16 | 7 | 102 | 6 |
| Cancer | 9 | 2 | 699 | 6 |
| Lenses | 4 | 3 | 24 | 5 |

Table 2 shows the mean and confidence interval for classification performance given on the train and test datasets. A discussion is given about the comparison with non-modular pattern classification (EA and CNE) in terms of training and generalisation performance on the test dataset. The results show that the Ionosphere, Zoo and Cancer classification problems seem to have little effect on the performance of the different subtasks. This shows that the module from the first subtask $\Omega_1$ contains all the relevant information to develop the basic knowledge modules where not much is addressed from the rest of the features. Hence, in a way, the rest of the features have little information for the classification problem and would be eliminated by feature selection algorithms. Moving on, for the rest of the problems, the knowledge from first subtask has been further improved by additional subtasks ($\Omega_2$ and $\Omega_3$) which is shown for the Wine, Iris and Lenses problems.

Overall, it is reasonable to compare $\Omega_3$ with CNE and EA since they use all the features. Hence, $\Omega_3$ gives the best generalisation (test) performance for Wine, Ionosphere, Zoo, and Lenses datasets. In the rest, i.e Iris and Cancer datasets, CMTL performance is close to either CNE or EA. Note that in general, EA gives the weakest performance when compared to CNE and CMTL. Similar trend can also be observed for the training performance. Overall, the results show that CMTL can perform similar or better when compared to the non-modular pattern classification methods (EA and CNE).

Table 2: Comparison with related methods

| Prob. | | EA | CNE | $\Omega_1$ | $\Omega_2$ | $\Omega_3$ |
|---|---|---|---|---|---|---|
| Wine | Train | 66.43 ± 10.10 | 75.00 ± 7.58 | 75.00 ± 5.66 | 87.33 ± 3.25 | 91.75 ± 2.19 |
| | Test | 74.92 ± 9.71 | 86.17 ± 3.55 | 74.88 ± 5.57 | 83.96 ± 4.54 | 93.99 ± 2.74 |
| Iris | Train | 69.79 ± 8.49 | 91.39 ± 2.91 | 54.83 ± 5.11 | 76.00 ± 4.17 | 88.17 ± 2.34 |
| | Test | 71.00 ± 8.08 | 90.00 ± 2.27 | 65.03 ± 3.88 | 78.36 ± 5.66 | 87.45 ± 5.02 |
| Ionos. | Train | 95.06 ± 0.60 | 90.44 ± 1.14 | 93.70 ± 0.52 | 94.74 ± 0.73 | 95.26 ± 0.73 |
| | Test | 94.65 ± 1.04 | 90.73 ± 1.80 | 98.23 ± 0.24 | 98.69 ± 0.23 | 99.01 ± 0.19 |
| Zoo | Train | 100.00 ± 0.00 | 100.00 ± 0.00 | 90.63 ± 0.00 | 90.31 ± 0.45 | 90.00 ± 0.45 |
| | Test | 90.42 ± 0.50 | 91.25 ± 0.80 | 100.00 ± 0.00 | 100.00 ± 0.00 | 100.00 ± 0.00 |
| Cancer | Train | 95.45 ± 0.29 | 91.60 ± 0.90 | 97.52 ± 0.13 | 97.83 ± 0.29 | 97.81 ± 0.33 |
| | Test | 97.13 ± 0.51 | 95.10 ± 0.63 | 95.30 ± 0.12 | 96.20 ± 0.23 | 96.41 ± 0.23 |
| Lenses | Train | 55.15 ± 3.62 | 52.62 ± 2.87 | 44.58 ± 9.37 | 39.58 ± 7.89 | 62.50 ± 6.43 |
| | Test | 67.67 ± 1.06 | 62.75 ± 2.26 | 2.92 ± 2.25 | 11.67 ± 4.65 | 79.17 ± 8.42 |

## 4 Discussion

The results have shown that the subtasks have important characteristics that contribute to overall decision decision making through the network modules. In some cases, the problem can be solved by the first subtask alone while in others, the other subtasks are needed. This highlights a further advantage of of the proposed methodology which highlights strength or feature contribution during

and at the end of the learning process. Apart from this, the main feature is the ability of the algorithm to make decision with some degree of error based on the base subtask in cases the features from the rest of the subtasks are missing.

CMTL differs from cooperative coevolutionary methods [18,3] by the way the problem is decomposed and the way the fitness for each individual is calculated. In CMTL, the fitness of an individual from a sub-population generally depends on the knowledge from the modules. This is different for the case of cooperative coevolution as the fitness of an individual is calculated when it is concatenated with the best individuals from all the sub-populations. This difference makes CMTL particularly useful for developmental and multi-task learning.

The depth of evolution determines how much training is needed for the particular subtask before knowledge is utilised in the larger cascaded network module. Although a fixed depth of evolution was used for all the sub-populations, there needs to be an investigation between the depth of evolution and transfer of knowledge for different types of problems. Feature selection methods [8,19] in a prior stage could help in depending better feature groups that consider interdependencies, relevance and importance for the constriction of various subtasks. This could be effective for real-world applications.

A limitation of the algorithm is timely convergence since it is based on neuroevolution which is a black-box non-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.

## 5 Conclusions and Future Work

The application of co-evolutionary multi-task learning for modular pattern classification incorporated feature-based knowledge development. It provided information about contribution of the respective subtasks during and after the learning process. In results show that in some cases, the problem can be solved by the first subtask alone while in others, the other subtasks are needed. Moreover, the results comparable to non-modular methods while having modular features for dynamic and robust pattern classification. The main feature of the algorithm is the ability to make decisions with some degree of error given the base subtask is present and others are missing.

In future work, it would be worthwhile to study if and modular pattern classification can be a way for developing robust control systems. The proposed algorithm can be extended to datasets to address missing values or attributes in the test stage. Heterogeneous transfer learning that considers various streams of data is a major challenge of data science. This can be addressed by an extension of the proposed method where the modules identified could be seen as different sources of data with heterogeneous feature space. Moreover, 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.

## References


1. Angeline, P., Saunders, G., Pollack, J.: An evolutionary algorithm that constructs recurrent neural networks. Neural Networks, IEEE Transactions on 5(1), 54 –65 (1994)
2. Caruana, R.: Multitask learning. In: Learning to learn, pp. 95–133. Springer (1998)
3. Chandra, R., Frean, M., Zhang, M.: On the issue of separability for problem decomposition in cooperative neuro-evolution. Neurocomputing 87, 33–40 (2012)
4. Chandra, R., Ong, Y.S., Goh, C.K.: Co-evolutionary multi-task learning for dynamic time series prediction. CoRR abs/1703.01887 (2017), `http://arxiv.org/abs/1703.01887`
5. Chandra, R., Ong, Y.S., Goh, C.K.: Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction. Neurocomputing 243, 21–34 (2017)
6. Chandra, R., Zhang, M.: Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. Neurocomputing 186, 116 – 123 (2012)
7. Clune, J., Mouret, J.B., Lipson, H.: The evolutionary origins of modularity. Proceedings of the Royal Society of London B: Biological Sciences 280(1755) (2013)
8. Dash, M., Liu, H.: Feature selection for classification. Intelligent data analysis 1(3), 131–156 (1997)
9. Ellefsen, K.O., Mouret, J.B., Clune, J.: Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Comput Biol 11(4), 1–24 (04 2015)
10. Geschwind, N., Behan, P.: Left-handedness: Association with immune disease, migraine, and developmental learning disorder. Proceedings of the National Academy of Sciences 79(16), 5097–5100 (1982)
11. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
12. Happel, B.L., Murre, J.M.: Design and evolution of modular neural network architectures. Neural Networks 7(6-7), 985 – 1004 (1994)
13. Johnson, M.K.: A multiple-entry, modular memory system. Psychology of Learning and Motivation, vol. 17, pp. 81 – 123. Academic Press (1983)
14. Lee, M.H., Meng, Q., Chao, F.: Developmental learning for autonomous robots. Robotics and Autonomous Systems 55(9), 750–759 (2007)
15. Meunier, D., Lambiotte, R., Bullmore, E.T.: Modular and hierarchically modular organization of brain networks. Frontiers in neuroscience 4, 200 (2010)
16. Miller, W.T.: Real-time application of neural networks for sensor-based control of robots with vision. IEEE Transactions on Systems, Man, and Cybernetics 19(4), 825–831 (1989)
17. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. Parallel problem solving from naturePPSN III pp. 249–257 (1994)
18. Potter, M.A., De Jong, K.A.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evol. Comput. 8(1), 1–29 (2000)
19. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. bioinformatics 23(19), 2507–2517 (2007)