Co-evolutionary multi-task learning for modular pattern classification

Dr. Rohitash Chandra

1. Centre for Translational Data Science
http://sydney.edu.au/data-science
2. Discipline of Business Analytics
School of Business

International Conference on Neural Information Processing
Ramada Pearl Hotel, Guangzhou, China
www.iconip2017.org

Overview

Background

- Neuro-evolution refers to the use of evolutionary algorithms for training. Cooperative coevolution (CC) is an evolutionary computation method that decomposes a problem into subcomponents and employs standard evolutionary algorithms in order to gradually solve the bigger problem. The subcomponents are also known as species or modules and are represented as subpopulations.
- The subpopulations are evolved separately and the co-operation only takes place for fitness evaluation for the respective individuals in each subpopulation.

Background and Motivation

- Modular neural networks are motivated from repeating structures in nature. The goal is to store knowledge as modules so that disruption to certain modules do not disrupt the entire network.
- Multi-task learning employs shared representation knowledge for learning multiple instances from the same problem. In the case of time series, multi-task learning can consider different a set of embedding dimensions as tasks that have shared knowledge representation.
- Although neuro-evolution has been successfully applied for training neural networks, multi-task learning for enhancing neuro-evolution has not been fully explored.

Background and Motivation

- 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.

Method: Robot Motivation

Consider a robot navigation problem that depends on input sensors that are controlled through a neural network.

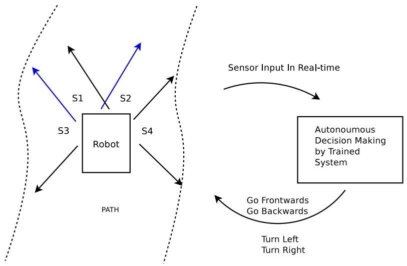
S1
S2
S3
S4
Robot
PATH
Sensor Input In Real-time
Autonoumous Decision Making by Trained System
Go Frontwards
Go Backwards
Turn Left
Turn Right

Figure: 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.

Proposed Method: Co-evolutionary multi-task learning

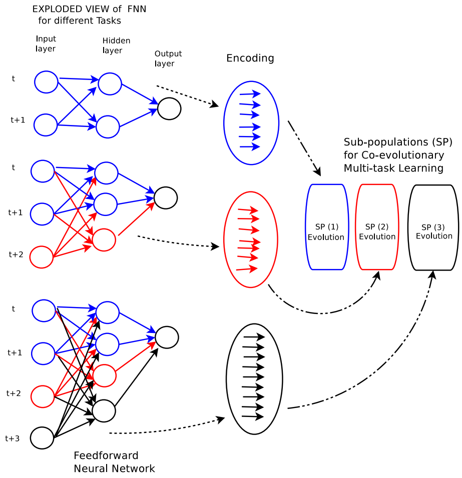
EXPLODED VIEW of FNN for different Tasks
Input layer
Hidden layer
Output layer
Encoding
t
t+1
t+2
t+3
Sub-populations (SP) for Co-evolutionary Multi-task Learning
SP (1) Evolution
SP (2) Evolution
SP (3) Evolution
Feedforward Neural Network

Figure: Subtasks in co-evolutionary multi-task learning. Subtask 1 employs a network topology with 2 hidden neurons while the rest of the subtasks add extra input and hidden neurons.

Co-evolutionary multi-task learning

The base module is given as $\Phi_1 = [\omega_1, \upsilon_1]$. Multi-task learning is used via coevolution to update the respective network module Φ_m given subtask Ω_m. Note that the cascaded network module θ_m of subtask m is constructed by combining with current Φ_m and previous network module Φ_{m-1} as follows.

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

Co-evolutionary multi-task learning

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

$$\begin{aligned} y_1 &= f(\theta_1, \Omega_1) \\ y_2 &= f(\theta_2, \Omega_2) \\ &\vdots \\ y_M &= f(\theta_M, \Omega_M) \end{aligned} \tag{2}$$

CMTL

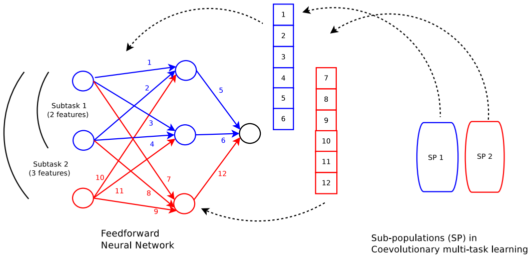

Figure: This cascaded modular neural network which can also be viewed and implemented as an ensemble of neural networks. The colours associated with the synapses in the network are linked to their encoding that are given as different modules. Subtask 1 employs a network topology with 2 hidden neurons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

Algorithm - CMTL

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
```

Design of Experiments

- An experimental study that compares the performance of CMTL with conventional evolutionary (single task learning) methods such as cooperative neuro-evolution (CNE) and an evolutionary algorithm (EA).
- Pattern classification problems from UCI machine learning repository is considered.
- 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 Ω_1 contains first 50 % of the features, while subtasks two and three (Ω_2 and Ω_3) contain 75 % and 100 %, respectively.

Design of Experiments

- The number of hidden neurons for foundation module that corresponds to Ω_1 is given by $\hat{h}$ in Table 1. The rest of hidden neurons in the respective network modules are incremented 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 γ 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.

Dataset

Table: 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

South Indian Ocean

Table: Comparison with related methods

Prob.		EA	CNE	Ω_1	Ω_2	Ω_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

Discussion

- 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.
- The key 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.

Conclusions and Future Work

- 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.
- A limitation of the algorithm is timely convergence since it is based on neuro-evolution which is a black-box non-gradient based learning method. Hence, big data problems could not be considered in this work.
- 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.

Thank You

More information: https://rohitash-chandra.github.io