Backpropagation for modular pattern classification

Rohitash Chandra[1,2], Sally Cripps[1,2]

[1] Centre for Translational Data Science
[2] School of Mathematics and Statistics
The University of Sydney, Sydney NSW, Australia

Abstract. Modular knowledge development in neural networks has the potential to feature robust decision given sudden changes in the environment or the data during real-time implementation. It can also provide a means to address robustness in decision making given certain features of the data are missing post training stage. In this paper, we present a modular backpropagation neural network that features developmental learning where the training takes into account several subgroups of features that constitute the bigger learning task. The learning employs multi-task learning where knowledge from previously trained neural network modules are used to guide knowledge developmental in future modules. The proposed method is tested on benchmark pattern classification datasets and the results show that it is possible to implement a modular network without losing training or generalization performance.

Key words: Backpropagation, modular network design, multi-task learning, modular pattern classification

1 Introduction

Embedded control systems guided with machine learning can encounter states that have disruptions in stream of data from sensors [1]. This can as a result provide poor decision making for actuators. Modular knowledge development in neural networks can provide features of robustness and scalability given the changes in the environment or the data [2]. Examples include block-based neural networks that have been implemented and reconfigured in digital hardware such as field programmable gate arrays [3]. Modular neural networks have been popular for hardware implementations [4]. Due to knowledge representation as modules, the system should be able to make decisions with some degree of uncertainty even if some of the modules are damaged or missing. Such motivations come from biological neural systems, i.e due to modular knowledge representation, one is able to see even if one eye is damaged [5].

Multi-task learning considers a shared knowledge representation that exploits knowledge learned from different but related tasks that can be helpful for each other during learning [6]. An example multi-task learning is a system that learns to recognise the face and facial expression recognition at the same time [7]. Transfer learning, on the other hand, considers a one-way transfer of knowledge from source tasks into target task [8] and recently have been popular for deep learning [9]. The main goal of transfer learning is to improve the the training

or generalization performance of the target task. In some problems, more than one source data would exist and hence the area is known as multi-source transfer learning [10]. In cases where the source task data has features that are not alighted directly or inconsistent with target task, the problem is knowledge as heterogeneous and multi-view transfer learning [11]. Ensemble learning considers a group of classifiers such as neural networks [12,13] in order to improve generalization performance. The notion of multi-task learning has been incorporated in the field of time series to address dynamic time series problems [14] and multi-step ahead prediction [15]. These methods incorporate elements of transfer, ensemble and multi-task learning. Developmental learning differs from conventional learning algorithms as the goal is not only concerned with building task specific knowledge and models, but to support the growth of learning for continuous development [16,17].

We incorporate the motivations from related learning techniques, in particular, multi-task, developmental and transfer learning for modular knowledge representation in neural networks. In this paper, we present a modular backpropagation neural network that takes into account several groups feature spaces that are partitioned from the data. The proposed method is tested on benchmark pattern classification datasets The method produces a modular network that provides decision making for the partitioned feature groups. The modular network would be functional even if some of the neurons and connections are perturbed from selected modules in the trained network.

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

2 Modular Backpropagation

As mentioned earlier, the proposed modular neural network features notions from multi-task, transfer and developmental learning. The the training takes into account several groups of overlapping features partitioned from the original dataset. The proposed modular backpropagation algorithm employs the theme of multi-task and transfer learning where knowledge from previously trained neural network modules are used to guide knowledge development for larger modules which feature the building blocks of knowledge. In order to implement this strategy, modular backpropagation with gradient decent is proposed that learns different network modules and transfer the knowledge from smaller to larger network modules. Essentially, the network architecture can be viewed as an ensemble of neural networks defined by cascading network architecture that increase in size with their corresponding feature space. Each feature space Ω in modular backpropagation is defined by the combination of feature groups. A feature group Φ_n is essentially a subset of features (f_i) from the dataset, for instance $\Phi_1 = [f_1, f_2, f_3]$. Therefore, the feature space is overlapped and defined as follows.

$$\begin{aligned} \Omega_1 &= [\Phi_1] \\ \Omega_2 &= [\Phi_1, \Phi_2] \\ \Omega_3 &= [\Phi_1, \Phi_2, \Phi_3] \\ \Omega_n &= [\Phi_1, \Phi_2, ..., \Phi_n] \end{aligned} \tag{1}$$

The size of the feature group and feature space needs to be defined experimentally or can be dependent on the problem given the contribution of the features. It is important to have the most contributing features in the first feature space in order to make use of the full potential of the algorithm. There is a need for appropriate feature selection algorithms as a pre-processing stage in finding the value of the features. Since feature selection is beyond the scope of the proposed method, the decomposition for feature groups that make the feature space would be done done arbitrarily as follows. For instance, feature space one (Ω_1) contains first 25 % of the features selected arbitrarily, while the rest (Ω_2, Ω_3, Ω_4) contain 50 %, 75 %, and 100 % of the features, respectively.

The number of input i, hidden h and output o neurons in an ensemble of cascaded network architectures defines a knowledge module $\theta_n = f(i_n, h_n, o)$, where the number of output neurons is same for the respective modules and $f()$ represents the feedforward network. Hence, n modules are defined by concatenation of knowledge modules which can also be viewed as network ensembles. Note that the input size for a given module is given by the size of the corresponding feature space, $i_n = size(\Omega_n)$. The number of hidden neurons for different modules can vary. In the proposed modular network architecture, we consider that the hidden neurons used in a given module is computed simply by considering the number of hidden neurons used in the previous module, $h_n = h_(n-1) + \tau$ where $\tau = h_1$. It refers to the number of hidden neurons in the base knowledge module θ_1.

Algorithm 1 gives an overview of the modular backpropagation which employs gradient descent. The training of the network modules are implemented in a incremental mode where each module is trained for a short period of time given by a fixed or adaptive depth d which is predetermined experimentally. The procedure repeats in a round-robin fashion until the termination condition is met which could be given by the number of epochs or the minimal training performance. This incremental training strategy can be viewed as developmental learning [16,17] while the transfer of knowledge from the smaller modules to larger ones could be seen as heterogeneous transfer learning [10]. Figure 1 shows the stage of the network with two modules after knowledge has been transfered from module 1 to module 2. The transfer of knowledge considers weights and bias in the respective layers as shown in the figure. The implementation of Algorithm 1 in Python is given online [3] .

3 Simulation and Analysis

This section presents experimental evaluation of the proposed modular backpropgation method for feature-based learning. We compare the results with canonical

[3] https://github.com/rohitash-chandra/modular-backpropagation-classification

Alg. 1 Modular Backpropagation

Step 1: Partition n groups of features spaces Ω_n from data.
Step 2: Define cascaded network ensembles of modules: $\theta_n = f(i_n, h_n, o)$
while until termination **do**
  for each Module θ_n **do**
    for Depth d **do**
      i. Forwardpropagate(θ_n, Ω_n)
      ii. Backpropagate using Gradient-descent(θ_n, Ω_n)
      iii. a) Calculate gradients
      b) Weight updates
    end for
    transfer-knowledge(θ_n, θ_{n+1})
  end for
end while

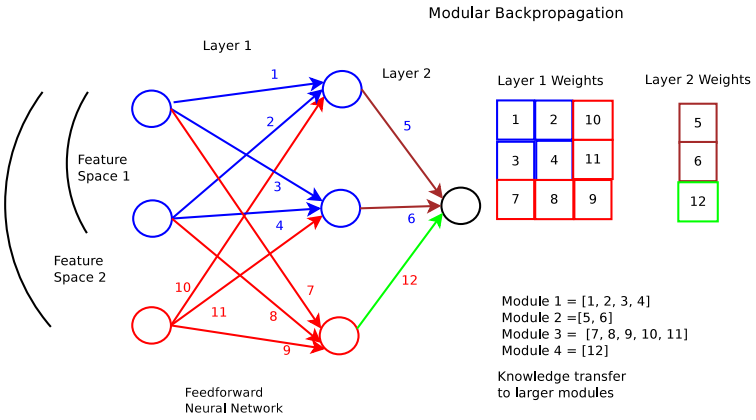

Fig. 1. Knowledge transfer through modular backpropagation. Note that the modules are represented by different colours. The neural network is implemented as an ensemble of cascaded network architectures linked with their corresponding feature space.

backpropagation that features stochastic gradient descent for pattern benchmark classification tasks.

3.1 Experimental Design

We selected twelve problems from the University of California, Irvine machine learning data repository [18] where the data was partitioned into 4 groups that contained overlapping features given the respective feature groups. The size δ of each feature space is given by the proportion η considering the total number of features for the given problem p.

$$\eta_n = [0.25, 0.5, 0.75, 1] \tag{2}$$

$$\delta_n = \eta_n * p \tag{3}$$

Table 1 gives details of the respective problems used with details about the number of features, number of classes, and number of instances. Note that the number of hidden neurons for foundation module that corresponds to Ω_1 is given by (λ). The rest of hidden neurons in the network modules for the respective feature spaces are incremented by $h_n = \lambda + h_(n-1) + \epsilon$, given that $h_1 = \lambda$. We use $\epsilon = 2$ as in our experiments.

The termination condition is also given. Note that the two major strategies in modular back-propagation approach deal with number of iterations used for each module while they are evolved or trained in a round-robin fashion fashion. In modular backpropagation, we investigate the performance of two instances where adaptive and fixed depth of search are used, respectively. The fixed depth of $f = 5$ is used while the adaptive depth ω considers the following for each module, $\omega_n = [10, 7, 4, 1]$. The motivation behind the adaptive depth is to use more training time for building blocks of knowledge given by the foundational modules and feature spaces. The termination condition is given by maximum epochs or minimum training performance (97 %) has been reached.

Table 2 show the classification performance results that mean (Train and Test) and standard deviation (std) for 30 independent experimental runs. The results for MBP for the instances of fixed and adaptive depth are further compared with canonical backpropagation that employs gradient descent (BP-GD). Note that the results of the different feature space for the respective problem has been shown (Ω_n) in Table 2. We evaluate the results where we compared the fixed and adaptive depth MBP in terms of of training and generalisation performance. The two strategies have similar performance for most of the problem. The major difference is shown for the Balloon problem where the fixed depth gives a much better training and test performance when compared to adaptive depth strategy. In the Heart problem, the fixed depth MBP - Ω_4 gives a much better training performance, howsoever, both methods give a similar test performance. The Tic-Tac-Toe problem also gets better training and test performance by fixed depth strategy. The Lenses problem gets better training performance by fixed depth strategy, howsoever, both methods have the same test performance. In general, we find that the fixed depth MBP provides better performance than adaptive depth MBP. Next, we compare the results given by fixed and adapt

Table 1. Configuration.

Dataset	Features	Classes	Instances	λ	Max. Time
Iris	4	3	150	6	500
Wine	13	3	178	6	500
Cancer	9	2	699	6	1000
Heart	13	2	270	16	2000
Credit	15	2	690	20	3000
Baloon	4	2	20	5	500
Tic-Tac-Toe	9	2	269	30	2000
Ionosphere	34	2	351	8	500
Zoo	17	7	101	6	300
Lenses	4	3	24	5	500
Balance	4	3	625	8	200
Robot (Four)	4	4	5456	14	2000
Robot(TwentyFour)	24	4	5456	14	2000

depth MBP with BP-GD. We choose to compare MBP-Ω_3 with BP-GD as since they use all the features in the feature space. We observe that fixed depth strategy (Ω_4) gives similar or outperforms in training performance than BP-GD in most of the problems except that of the Tic-Tac problem. In the test performance, we observe that fixed depth MBP gives better performance for most of the problems, except for the Tic-Tac problem. Similarly, adapt depth (Ω_4) also gives better test performance than GD in general.

3.2 Discussion

The results suggest that modularity enforced through backpropagation helped to retain the knowledge of the smaller network modules that are used as building blocks of knowledge. Heterogeneous form of transfer learning is employed to transfer knowledge from small network modules to larger network modules which link with different, but overlapping feature spaces. Hence, through transfer learning, the knowledge in foundational modules is projected through the learning of the entire network. This network architecture can be very useful in hardware implementations [4] given that real-time embedded systems may not have full information from sensors in some cases, the network will be able to provide a decision from knowledge in the foundational building blocks or modules. Modular back-propagation considered an ensemble of cascaded network modules that employ heterogeneous transfer learning for utilizing knowledge in smaller modules. This can also be viewed as a form of dynamic programming [19] as the problem is decomposed into modules and solution from the foundational modules are used in larger modules through transfer learning. The proposed modular backpropagation implements a neural network that is operational with a degree of error even when some of the neurons or links in the larger modules are damaged. Moreover, it would be operational with a degree of decision making when associated input features in the bigger feature space is unavailable during an event.

Table 2. Results of modular backpropgation (MBP) with feature space (Ω_n).

Problem	Domain	Adapt	Depth	MBP		Fixed	Depth	MBP		BP-GD
		Ω_1	Ω_2	Ω_3	Ω_4	Ω_1	Ω_2	Ω_3	Ω_4	
Iris	Train	55.73	60.52	74.79	80.18	52.70	59.27	89.76	95.09	93.30
	(std)	4.89	3.72	4.96	4.39	4.95	5.04	3.16	1.38	0.89
	Test	85.25	88.67	89.42	93.08	88.75	90.00	91.08	94.42	83.50
	(std)	5.38	5.07	3.14	4.46	5.31	4.38	4.36	2.56	2.47
Wine	Train	98.26	99.81	99.83	99.95	98.09	99.86	100.00	100.00	98.45
	(std)	0.91	0.42	0.55	0.26	1.25	0.34	0.00	0.00	0.87
	Test	99.75	100.00	100.00	100.00	99.75	99.83	100.00	100.00	83.25
	(std)	0.75	0.00	0.00	0.00	0.75	0.62	0.00	0.00	9.90
Cancer	Train	86.32	90.03	93.59	93.84	86.03	89.56	94.00	94.56	94.07
	(std)	0.75	0.90	0.65	0.81	0.53	0.85	0.48	0.73	0.54
	Test	97.73	97.49	98.37	98.44	97.46	97.67	98.22	98.41	96.24
	(std)	0.47	0.87	0.27	0.39	0.55	0.91	0.27	0.41	0.57
Heart	Train	49.61	56.76	68.39	77.03	47.89	55.21	71.99	88.20	90.92
	(std)	3.72	4.36	2.11	2.45	4.21	5.15	2.96	2.69	1.44
	Test	74.78	75.67	72.56	79.11	75.22	76.22	73.41	77.56	70.63
	(std)	1.84	1.63	2.65	2.19	1.70	1.74	1.88	2.42	3.14
Credit	Train	19.79	42.53	82.34	84.35	18.94	40.62	83.24	87.44	89.64
	(std)	2.91	4.46	1.86	2.26	4.15	5.18	1.97	1.39	1.07
	Test	56.97	61.82	82.22	82.64	55.23	62.40	82.71	82.06	78.79
	(std)	3.32	2.96	1.73	1.43	4.59	2.76	1.67	1.48	1.56
Balloon	Train	7.14	15.00	34.76	44.29	8.57	12.14	45.24	99.52	100.00
	(std)	10.10	9.82	12.61	6.23	10.50	11.97	10.49	1.78	0.00
	Test	50.00	50.00	58.33	81.67	50.00	50.00	62.78	100.00	91.11
	(std)	0.00	0.00	14.75	21.24	0.00	0.00	15.33	0.00	14.74
Tic-Tac	Train	33.05	42.18	61.81	71.14	36.07	40.06	62.10	79.65	97.40
	(std)	9.04	6.68	3.83	2.85	9.33	7.18	4.52	3.54	0.24
	Test	0.00	8.29	12.08	24.97	0.00	8.39	13.22	38.96	61.24
	(std)	0.00	3.40	2.33	6.65	0.00	3.39	3.38	12.18	6.90
Ionosph.	Train	85.03	91.07	94.39	94.80	80.98	88.98	93.58	95.97	94.88
	(std)	1.92	1.54	1.55	2.58	2.32	2.01	1.53	0.86	1.32
	Test	89.54	92.48	94.19	93.94	90.00	92.26	94.40	94.74	84.62
	(std)	4.42	2.12	1.43	2.07	4.70	1.47	2.34	1.72	1.99
Zoo	Train	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.81
	(std)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60
	Test	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.44
	(std)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.93
Lenses	Train	83.12	75.42	83.96	61.67	79.17	77.92	94.17	95.21	99.81
	(std)	15.15	17.60	21.51	24.72	14.99	15.46	9.12	5.97	0.60
	Test	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.44
	(std)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.93
Balance	Train	94.09	95.62	95.98	93.35	93.94	95.36	96.11	94.65	92.29
	(std)	0.00	0.72	0.63	1.05	0.57	0.97	0.53	1.60	7.85
	Test	100.00	100.00	99.98	100.00	100.00	100.00	100.00	100.00	88.33
	(std)	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	8.50
Robot	Train	99.95	100.00	100.00	100.00	99.93	100.00	100.00	100.00	95.95
(Four)	(std)	0.15	0.00	0.00	0.00	0.17	0.00	0.00	0.00	1.69
	Test	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.81
	(std)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.39
Robot	Train	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.69
(Twenty	(std)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.75
Four)	Test	100.00	100.00	100.00	100.00	1.17	100.00	100.00	100.00	94.35
	(std)	0.00	0.00	0.00	0.00	2.17	0.00	0.00	0.00	6.21

4 Conclusions and Future Work

We presented an ensemble of cascaded neural network architecture that employs modular backpropgation algorithm for feature-based pattern classification. The method incorporates heterogeneous transfer learning in order to utilize knowledge from smaller modules into larges ones. The major goal of the method was to provide modularity in knowledge representation and decision making so that it

can be applied to problems where real-time implementations may have cases where full feature space or all the features are available. The results shows that the proposed approach can deliver similar or better performance to that of canonical (non-modular) backpropagation network. Hence, we were able to obtained train neural networks that did not lose performance although their knowledge representation was implemented with modularity. Moreover, although unexpected, the modular approach outperformed canonical backpropagation in several cases.

In future work, the method can be adapted for other types of problems that include image classification. Moreover, the modular method needs to be adapted further so that the trained network is operational even when the foundational input feature space in unavailable during test phase. Uncertainty quantification through Bayesian inference methods for modular networks can also be explored. Furthermore, the method can be extended to the application of deep learning .

References

1. Hunt, K.J., Sbarbaro, D., Żbikowski, R., Gawthrop, P.J.: Neural networks for control systemsa survey. Automatica **28**(6) (1992) 1083–1112
2. Auda, G., Kamel, M.: Modular neural networks: a survey. International Journal of Neural Systems **9**(02) (1999) 129–151
3. Moon, S.W., Kong, S.G.: Block-based neural networks. IEEE Transactions on Neural Networks **12**(2) (Mar 2001) 307–317
4. Misra, J., Saha, I.: Artificial neural networks in hardware: A survey of two decades of progress. Neurocomputing **74**(13) (2010) 239 – 255 Artificial Brains.
5. Johnson, M.K.: A multiple-entry, modular memory system. Volume 17 of Psychology of Learning and Motivation. Academic Press (1983) 81 – 123
6. Caruana, R.: Multitask learning. In: Learning to learn. Springer (1998) 95–133
7. Zheng, H., Geng, X., Tao, D., Jin, Z.: A multi-task model for simultaneous face identification and facial expression recognition. Neurocomputing **171** (2016) 515 – 523
8. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. **22**(10) (October 2010) 1345–1359
9. Zeng, T., Ji, S.: Deep convolutional neural networks for multi-instance multi-task learning. In: Data Mining (ICDM), 2015 IEEE International Conference on. (Nov 2015) 579–588
10. Xu, Z., Sun, S.: Multi-source transfer learning with multi-view adaboost. In: International Conference on Neural Information Processing, Springer (2012) 332–339
11. Sun, S.: A survey of multi-view machine learning. Neural Computing and Applications **23**(7-8) (2013) 2031–2038
12. Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks. Technical report, DTIC Document (1992)
13. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. Artificial intelligence **137**(1-2) (2002) 239–263
14. Chandra, R., Ong, Y., Goh, C.: Co-evolutionary multi-task learning for dynamic time series prediction. CoRR **abs/1703.01887** (2017)
15. Chandra, R., Ong, Y., Goh, C.: Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction. Neurocomputing **243** (2017) 21–34

16. Geschwind, N., Behan, P.: Left-handedness: Association with immune disease, migraine, and developmental learning disorder. Proceedings of the National Academy of Sciences **79**(16) (1982) 5097–5100
17. Lee, M.H., Meng, Q., Chao, F.: Developmental learning for autonomous robots. Robotics and Autonomous Systems **55**(9) (2007) 750–759
18. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
19. Boutilier, C., Dearden, R., Goldszmidt, M.: Stochastic dynamic programming with factored representations. Artificial intelligence **121**(1) (2000) 49–107