

Learning Mappings with Neural Network

Yefei Peng, Paul W. Munro
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15206 USA
{ypeng, pmunro}@mail.sis.pitt.edu

Abstract

The authors extended the idea of training multiple tasks simultaneously on a partially shared feed forward network. A shared input subvector was added to represent common inputs of all tasks. A “cross training” mechanism was used to specify corresponding components between the two tasks. By examining output of one network in response to stimulus from the other network, we can test if the network can learn the correspondence which was not cross-trained. Two kinds of studies on binary tree mapping were conducted. The shared input was used to represent “relation” between tree nodes. The result shows the network can fill in the missing correspondence with sufficient training data.

Keywords: shared weights, transfer, analogy

1. Introduction

A fundamental operator for high-level cognition is the establishment of correspondences between analogous components of structurally similar tasks. Thorndike & Woodworth [13] put forward a theory that two different “mental functions” may share cognitive structures in their processing. A system that includes common representations of features in different domains can conceivably establish mappings between them [4].

The intrinsic pattern-matching properties of the connectionist framework can be brought to bear on feature mapping [5, 7, 8] Approaches using hybrid symbolic-connectionist models have addressed the processing of structural mapping in the context of language processing [9] Skill transfer between similar, if not strictly analogous, tasks has been the subject of connectionist models that transfer knowledge by reusing weights learned on one task on a second task [12]. Caruana [1] devised a multitask learning (MTL) architecture that exhibits faster convergence and better performance learning multiple tasks simultaneously than on one of the constituent tasks alone.

An early implementation of IENN appeared at Munro’s work [10], which used feedforward network with two hidden layers and trained on three simple analogous tasks: three squares with different orientation. Dienes et al. [2] proposed a very similar network architecture which uses a modification of Elman’s [3] simple recurrent network (SRN) with two hidden layers. The network was trained on artificial grammar. Both of the works demonstrate that knowledge transfer can significantly benefit network learning.

2. Network Architecture

A partially shared network (PSN) [10, 11] is defined here as a feed-forward network that processes several input-output tasks with some input and output units dedicated to individual tasks, and other intermediate units and connections shared among the tasks.

The strictly layered PSN has an architecture in which the input vector is partitioned into K ($K > 1$) subvectors with the output vector is partitioned into K corresponding subvectors. There are 2 or more hidden layers. The connectivity from the input banks to the first hidden layer is parallel; hidden layers project from one to the next in serial fashion; the final step from the last hidden layer to the output banks is again in parallel.

Input could be fed from any input subvector, output can be read at any output subvector depending definition of task. Input subvector and output subvector do not have to have same index number. This will allows a task to be a cross task.

There is an optional input subvector which is shared among tasks. It represents common inputs of all tasks. For example, in the tree mapping task, this subvector of input layer could be used to represent different connection types between nodes of the tree. Specifically, there are four types of relations: identity, parent, child, and sibling. So there will be four nodes in this subvector, represent these four types of relationships respectively. At any time, only one node in this subvector will be activated.

3. Methodology

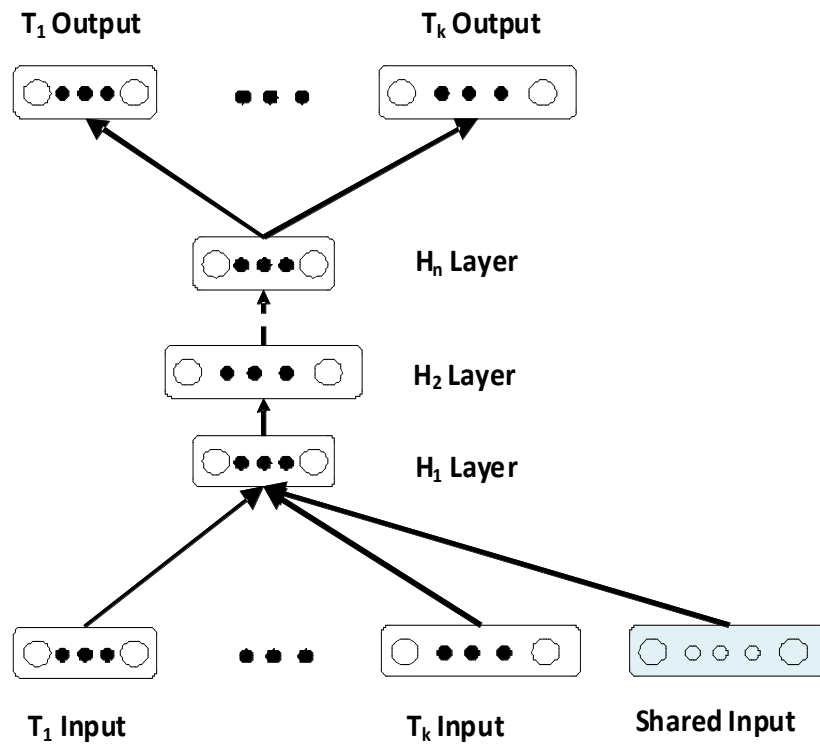


Fig. 1. A Strictly Layered PSN with shared input. Several banks of input units, T_j , project to a common hidden layer H_1 , which in turn projects to a second hidden layer H_2 , which projects to a subsequent layer H_3 , and so on to H_N , ($N > 1$). In the final stage, layer H_N projects to several output banks corresponding to the input banks. Arrows indicate full unit-to-unit connectivity between connected layers. S is shared input to all tasks.

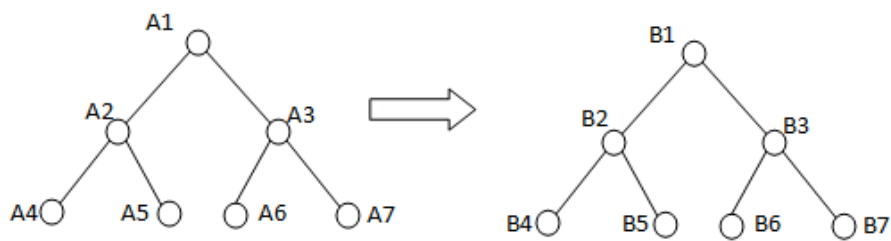


Fig. 2. A tree mapping example. Each 3 level binary tree was trained as a task. Mappings between the two networks were cross-trained and cross-tested.

3.1 Tasks

The networks in the study were trained on binary tree mapping tasks. We'll use example in Fig 2 to show how

the tasks were generated. Fig 2 shows a tree mapping example between two 3 level binary trees. Each tree itself is represented as one task. Each node is represented as a single unit in input bank. For example, node #1 will be represented as 1000000.

The shared input S represents relationships between nodes. There are four relationships: identity (self), parent, child, and sibling. Each relationship is represented as a single unit in S.

Combing one input in input bank and one relationship, the target at output bank is determined by the tree topology. For example, node #1's children are node #2 and #3 will be translated to: T_1 input (1000000) + S (0100) => target at T_1 output (0110000).

When learning correspondence between two trees, it's possible that there exists more than one valid mapping configuration depending on which nodes were cross trained. For example, if we know $A1 \leftrightarrow B1$, $A2 \leftrightarrow B2$, $A4 \leftrightarrow B4$, there are two valid mapping configurations: $A1 \leftrightarrow B1$, $A2 \leftrightarrow B2$, $A3 \leftrightarrow B3$, $A4 \leftrightarrow B4$, $A5 \leftrightarrow B5$, $A6 \leftrightarrow B6$, $A7 \leftrightarrow B7$; and $A1 \leftrightarrow B1$, $A2 \leftrightarrow B2$, $A3 \leftrightarrow B3$, $A4 \leftrightarrow B4$, $A5 \leftrightarrow B5$, $A7 \leftrightarrow B6$, $A6 \leftrightarrow B7$. The ambiguity is due to we cannot infer the mapping between A6, A7 and B6, B7.

3.3 Cross-testing

During training and testing, patterns are only presented on one input bank and shared bank. This restriction maintains a consistent net input to the units of H_1 . There is no reason for such a constraint among the output layers during testing. Thus output patterns across all banks can be examined in response to input on a single input bank. This makes possible a correspondence analysis between items in different tasks. That is, the output in T_2 generated by a T_1 input (A) can be compared to the outputs in T_2 generated by the set of T_2 inputs. Any T_2 input (A') that generates a T_2 output that is sufficiently close to that generated by A is a candidate for the image of A in the input space of T_2 .

Another approach to correspondence analysis is to compare representations of inputs from different input banks at the H_1 level. If a T_1 pattern and a T_2 pattern have identical or nearly identical H_1 representations, then they must give the same (or nearly the same) output patterns on all output banks.

As we stated in previous sections, it's possible that there exists more than one valid mapping configuration. In this case, we treat all these configurations as correct. When calculating error, we'll choose minimum error across all valid configurations.

3.4 Cross-training

The network can be explicitly trained to form a correspondence between a given pattern A in T_1+S and a given pattern A' in T_2+S . The back propagation algorithm

3.2 Interlaced Training

Networks were initialized by setting the weights to small random values from a uniform distribution. The networks were trained with two similar tasks (T_1+S and T_2+S) by presenting a random input-output pair from T_1 to input bank 1 and shared input S, training the net with back propagation, then training on a pair from T_2 to input bank 2 and shared input S, and proceeding by alternating between T_1 and T_2 . When training T_1 , the weights from T_1 input to H_1 , from S to H_1 , from H_1 to T_1 output, and all weights between hidden layers will be changed.

If there is cross training, it will be applied with a probability after each cycle of training T_1 and T_2 .

is applied by first presenting A' at the T_2+S input, the output at T_2 is then used as a target pattern for A on the T_1+S input.

4. Experiments

4.1 Experiment 1: Cross-training and Testing on Identity Relationship Only

The two tasks are 3 level binary trees with four relationships: identity, parent, child, and sibling. Only records with identity relationship were cross-trained and cross-tested. Totally there are 7 such records.

We used leave-one-out method and leave-two-out method. For leave-one-out method, 6 out of 7 records were cross-trained, with 1 cross-tested. For a 3 level binary tree, there are three cases: testing on node #1(root), #2(one of 2nd level nodes), and #4(one of 3rd level nodes) respectively. For leave-two-out method, 5 out of 7 records were cross-trained, with 2 cross-tested. For a 3 level binary tree, there are seven cases.

We also did leave-one-out experiment on a 4 level binary tree. There are four cases: testing on node #1(root), #2(one of 2nd level nodes), #4(one of 3rd level nodes), and #8(one of leave nodes) respectively.

In each case, we run the experiment for 10 trials. Number of successful trials is reported.

Table 1. Experiment Results for Experiment 1: Cross-training and Testing on Identity Relationship Only

| Tree level | Training | Testing | success(max10) |
|------------|-----------------------------------|---------|----------------|
| 3 | 2;3;4;5;6;7 | 1 | 10 |
| 3 | 1;3;4;5;6;7 | 2 | 10 |
| 3 | 1;2;3;5;6;7 | 4 | 5 |
| 4 | 2;3;4;5;6;7;8;9;10;11;12;13;14;15 | 1 | 10 |
| 4 | 1;3;4;5;6;7;8;9;10;11;12;13;14;15 | 2 | 10 |
| 4 | 1;2;3;5;6;7;8;9;10;11;12;13;14;15 | 4 | 6 |
| 4 | 1;2;3;4;5;6;7;9;10;11;12;13;14;15 | 8 | 3 |

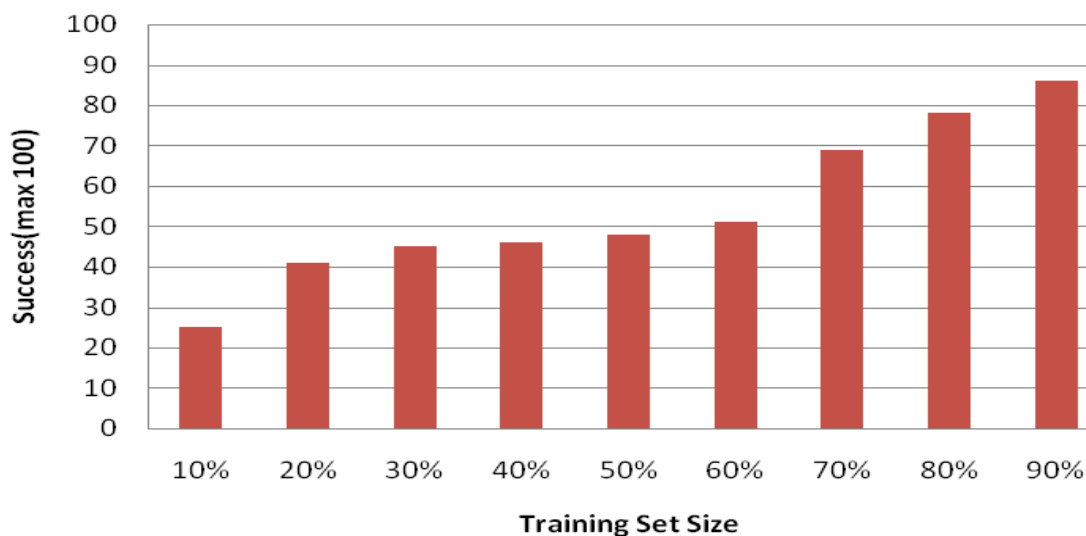


Fig. 3. Experiment Results for Experiment 2: Randomly Dividing Training and Testing with All Relationships

4.2 Experiment 2: Randomly Dividing Training and Testing with All Relationships

The two tasks are 3 level binary trees with four relationships: identity, parent, child, and sibling. All records were cross-trained and cross-tested. Totally there

are 22 training records. We randomly chose training records with different percentage, from 10% to 90%.

For each probability, we randomly divided data into training and testing set in 10 different ways, and run 10 trials for each dividing. Totally there are 100 trials for each probability. Number of successful trials is reported.

5. Results

5.1 Results of Experiment 1

Result is shown in Table 1. We can see that for 3 level binary tree, testing on #1 and #2 yield 10 out of 10 success. Testing on #4 yields 5 out of 10 successes. Similar trend is also observed in 4 level binary tree experiments. When testing node goes from root to leaf, success rate drops.

5.2 Results of Experiment 2

Result is shown in Fig 3. It clearly shows the trend that success rate increases with more training data. This experiment demonstrated that the network is generalized well to have shared relationship inputs.

6. Discussion

The ability to establish correspondences between similar situations is fundamental to intelligent behavior. Here, a network has been introduced that can identify corresponding items in analogous spaces. A key feature of this approach is that there is no need for the tasks to use a common representation. Essentially the first hidden layer provides a common representational scheme for all the input spaces.

It should be noted that the partially shared network architecture used here is virtually identical to the network used in Hinton's [6] classic "family trees" example. The network in that paper also had independent inputs and shared hidden units, but only briefly addresses the notion of generalization.

A possible application of this approach is for ontology mapping. Here, the tasks would be to learn the structure of various ontologies with relationship represented in shared input. Explicit cross-training could be used for specific "known" correspondences.

7. References

- [1] Caruana, R. (1997) Multitask learning. *Machine Learning*, 28:41-75.
- [2] Dienes, Z., Altman G. T. M., and Gao, S.-J. (1999) Mapping across domains without feedback:
- [3] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179--211.
- [4] Gentner, D., (1983) Structure-mapping: A theoretical framework for analogy, *Cognitive Science* 7:155-170.
- [5] Halford, G., Wilson, W., Guo, J., Gayler, R., Wiles, J., Stewart, J. (1994). Connectionist implications for processing capacity limitations in analogies. In: K. Holyoak & J. Barnden (eds.) *Advances in connectionist and neural computation theory, vol. 2, Analogical Connections*, pp. 363--415. Norwood, NJ: Ablex.
- [6] Hinton, G. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1-12, Amherst, Lawrence Erlbaum, Hillsdale.
- [7] Holyoak, K. & Thagard, P. (1989) Analogical mapping by constraint satisfaction. *Cognitive Science* 13, 295-355.
- [8] Hummel, J. & Holyoak, K. (1997) Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104, 427- 466.
- [9] Mitchell, M. (1993) *Analogy-making as Perception: A computer model*. Cambridge, MA: MIT Press.
- [10] Munro, P. (1996) Shared network resources and shared task properties. In: *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*. Mahwah NJ: Erlbaum
- [11] Munro, P. (2008) Learning Structurally Analogous Tasks. In: *Proceedings of the Eighteenth Conference of Artificial Neural Networks*. Prague, Czech Republic
- [12] Pratt, L. Y., Mostow, J., and Kamm, C. A. (1991) Direct transfer of learned information among neural networks. In: *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)* Anaheim CA
- [13] Thorndike, E. L., & Woodworth, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions. *Psychological Review*, 8, 247-261.