Knowledge Transfer between Neural Networks

Kenneth McGarry and John MacIntyre
School of Computing, Engineering and Technology
University of Sunderland
St Peter's Campus, St Peter's Way
Sunderland, SR6 ODD, UK
ken.mcgarry@sunderland.ac.uk

Abstract

The goal of knowledge transfer is to take advantage of previous training experience to solve related but new tasks. This paper tackles the issue of transfer of knowledge between radial basis function neural networks. We present some preliminary work illustrating how a neural network trained on one task (the source) can be used to assist in the synthesis of a new but similar task (the target).

1 Introduction

The robustness and pattern matching characteristics of neural networks has enabled them to be applied to many real-world, large-scale problems of considerable complexity [Bishop, 1995]. They provide solutions to a variety of classification problems such as speech, character and signal recognition, as well as functional prediction and system modeling where the physical processes are not understood or are highly complex. Most of this neural network development effort has concentrated upon what has become known as the *tabula rasa* approach, i.e. each neural network is developed from scratch using the appropriate training data and does not take advantage of previous task-related work.

However, humans tend to perform better at learning new tasks after having been previously trained on a similar task. It has been argued for a long time that transfer of knowledge is an essential human capability [Ellis, 1965]. In most situations humans first try to rely on our experience and adapt knowledge or a strategy which has been successful before. Neural networks generally have difficulties sharing their task experience because each network is trained individually on a specific task that may involve the modeling of a complex function. The learned function is stored across the weights and thresholds in a distributed form. This difficulty hinders the isolation and transfer of desirable feature or activity learned by the neural network to another task [Pratt, 1993]. This may not appear to be a problem since it is a relatively simple matter to train a neural network given enough data. However, such a methodology for network development is clearly not biologically plausible and also creates severe difficulties for on-line adaptive learning. Sharkey describes the process of knowledge transfer as "adaptive generalisation" and argues the case for inserting prior knowledge into a neural network and is worth repeating here at length:

> *"If connectionist nets are to be able to exhibit adaptive behaviour, they need to be prestructured. Such prestructuring can be accomplished through training on related tasks... A net can be said to exhibit a degree of adaptive generalisation when training on one task results in positive transfer to another task. In such a case, information has been extracted that facilitates the performance of a second task. On the other hand, when negative transfer is obtained, prior experience interferes with subsequent learning. In this way, not only can previous knowledge be incorporated by means of postive transfer, but a net can be seen as having a predisposition to learn certain tasks rather than others."* [Sharkey and Sharkey, 1993]

The problem is also related to the difficulties encountered by neural networks in those situations that require sequential learning. If the network is presented with new training patterns without including the original training set, then "catastrophic interference" may occur [McCloskey and Cohen, 1989; McRae and Hetherington, 1993]. This interference manifests itself as a loss of accuracy as the network "forgets" the old patterns

The remainder of this paper is structured as follows: section two discusses the terminology and details of task transfer as applied to neural networks; section three highlights the architecture and characteristics of radial basis function neural networks; section four describes the experimental methodology; section five discusses the results and section six presents the conclusions.

2 Knowledge Transfer

In this section we discuss motivations, techniques and methodology for knowledge transfer between RBF networks.

2.1 Task Transfer Terminology

In the literature, some terms used in task transfer have several meanings which can be confusing. It will be useful to define the meanings of the various terms which are used in the following sections:
(i) Task. A task is the particular function of the RBF network to be transferred. In all the instances presented in this thesis it will refer to classification tasks of some type e.g. Vibration fault classification, Iris species classification or Vowel classification.
(ii) Data. Data refers to the training and test examples used to train the RBF networks. It is effectively what the tasks are performed on.
(iii) Network. The RBF network is trained on a particular classification task using data from a particular domain.
(iv) Activity. An activity is a specific instance of a task transfer operation e.g activity 10 in table 5 refers to task E being transferred over to task A.
(v) Domain. A domain is a general area of expertise and can refer to all the knowledge in a given area e.g. the Iris data set is a collection of three types of Iris in the domain of flowers.

Early research on task transfer by Ellis has provided some metrics to gauge the progress of transfer within humans [Ellis, 1965]. This research can equally be applied to neural network learning. Ellis identified three results of attempting task transfer:
(i) Positive transfer. Learning the first task aided in learning the second task.
(ii) Negative transfer. The first task has hindered learning on the second task. The two tasks were so dissimilar that the network parameters were initialized to unsuitable values. This would result in the second task not reaching an acceptable level of accuracy or taking far longer than normal to train.
(iii) Zero transfer. No overall effect was observed by learning the first task. This could be as a result of small but equal positive and negative effects canceling each other out.

2.2 Potential advantages of task transfer

Assuming positive transfer has occurred, the following characteristics should be present in the target network:
(i) Modeling tasks of increased complexity. The rationale for knowledge transfer is based upon the fact that humans are able to learn tasks that are of increasing difficulty. However, if a difficult task is presented before the simpler prerequisite tasks then it is possible that the learner may not be able to successfully complete or will at best finish the task by taking an inordinate amount of time.
(ii) Learning on fewer training examples. A good indication of the level of intelligence in humans is the ability of a learner to quickly understand how to accomplish a task without being repeatedly told how to do it. Assuming task transfer was successful then the previous task should have provided the network parameters with useful initial values (or better than random values).
(iii) Training speedup. Humans tend to perform related tasks faster, it may be possible for neural networks to benefit from a similar speedup in training time.

3 Radial Basis Function Networks

Radial basis function (RBF) neural networks are a model that has functional similarities found in many biological neurons [Moody and Darken, 1989]. RBF networks have been proved to be capable of universal function approximation. RBF networks have been applied to several real-world, large-scale problems of considerable complexity. They are excellent at pattern recognition and are robust classifiers, with the ability to generalize in making decisions about imprecise input data. They offer robust solutions to a variety of classification problems such as speech, character and signal recognition, as well as functional prediction and system modeling where the physical processes are not understood or are highly complex. Figure 1 shows the architecture of a typical RBF network.

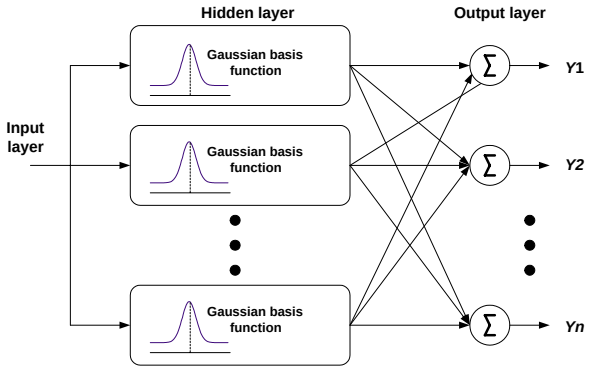

Figure 1: Radial basis function network

The RBF network consists of feedforward architecture with an input layer, a hidden layer of RBF units and an output layer of linear units. The input layer simply transfers the input vector to the hidden units, which form a localized response to the input pattern. This property appears to be very attractive for knowledge transfer in neural networks. The activation levels of the output units provide an indication of the nearness of the input vector to the classes. Learning is normally undertaken as a two-stage process. An unsupervised clustering technique is appropriate for the hidden layer while a supervised method is applied to the output layer units. The nodes in the hidden layer are implemented by kernel functions, which operate over a localized area of input space. The effective range of the kernels is determined by the values allocated to the centre and width of the radial basis function. While the Gaussian function is normally used as the receptive field other functions such as the thin-plate-spline function, multi-quadratic function and the inverse multi-quadratic functions have been used [Lowe, 1997].

4 Methodology

This section discusses the data sets used in the experimental work and the task transfer technique.

4.1 Data Sets

The data sets represent a variety of synthetic and real world problems of varying complexity (i.e. number of examples, input features and classes.).

Figure 2 gives the details of the data sets. The columns indicate the number of examples, the number of classes, the number of input features, if the data set contains continuous data, discrete data and the last column indicates if any data is missing.

4.2 Task specific constraints

Factors which must be considered are the differences between the source and target tasks. A source task consists of a pre-trained RBF network and/or the original data. A target task consists of the available training data (which may be insufficient) and information about the number of input features and output classes. A number of factors must be taken into account when judging the similarity between two tasks:

(i) Structural differences. For example, the number of inputs and outputs may not be the same for each task. If the source task has a greater number of inputs than the target task then the additional features may enable a better classifier to be built.

(ii) Symbolic differences. For example, the inputs and outputs present in the source task may not correspond to the same features on the target task. Even in strongly related domains such differences can occur.

(iii) Complexity differences. For example, either the source or the target task may be more complex. The complexity for each task is determined by the number of the degrees of freedom within the RBF network, training time, and the accuracy of the RBF network.

(iv) Spatial differences. For example, it is likely that the numerical values comprising the input space may differ to a great extent, this can be partially alleviated by scaling before the training the networks. Large numerical values would adversely affect the classification ability.

(v) Ordering differences. For example, related to the complexity difference as it may be easier to understand a simpler task before tackling a more complex task. Hence, the order in which task transfer occurs may be crucial.

Figure 3 shows the combinations of source task to target task input/output configurations.

The symbolic similarity measure can only be assessed manually. This required checking the input and output feature names for each data set and rating them accordingly as either "yes" or "no". For example, the vibration data sets were derived from a common data pool and several input features are common to both. As another example, the credit data sets (German, Japanese and Australian) have several input features named differently. However, certain relationships exist between these input features and can be taken to indicate the applicants economic situation

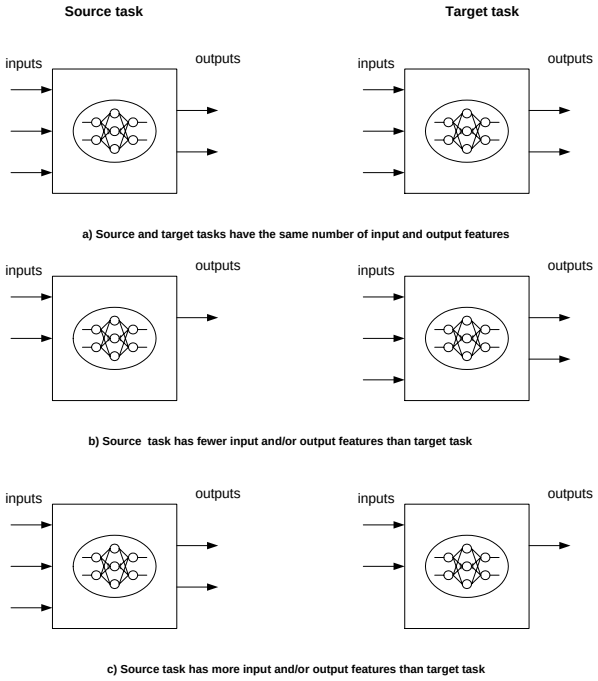

Figure 3: Source task to target task (structural similarity)

e.g. the Japanese input "bad region" has the same effect on classification as the German "Salary" input. The symbolic similarity measure was only useful for analyzing data sets from the same domain.

The first check was for structural similarity, the preferred situation was to have the same number of inputs and outputs for both tasks. However, a higher similarity rating was applied to target tasks with less input features than the source task (compared with the target task having more input features than the source task). This is because it is easier for the hidden units within the target task to "lose" a superfluous input than having to generate a "new" important input feature. Figure 3 shows the combinations of source task to target task input/output configurations.

4.3 Experimental approach for task transfer

We suggest it may be more appropriate to view task transfer within an RBF network as an analysis of the hidden units with the objective of recruiting those units that may be useful in representing the second task. The selected hidden units and weights are then copied and assigned to the new task. Figure 4 describes the transfer algorithm in detail.

The selection of RBF units deemed useful for transfer was based upon the activation levels of those units when presented with the second task training set. Those radial basis units that had consistently high (near 1) mean activation levels were selected for transfer. A variable set-point S for selecting the most active hidden units was used. A high value is initially

Figure 2: Composition of data sets used in experimental work

Data set	Cases	Classes	Attrib	Contin	Discrete	Missing
Xor(binary)	4	2	2	No	Yes	No
Xor(continuous)	100	2	2	Yes	No	No
Iris	150	3	4	Yes	No	No
Housing(see notes)	506	3	12	Yes	Yes	No
Vowell(Peterson)	1520	10	5	Yes	Yes	No
Vowell(Deterding)	990	11	11	Yes	Yes	No
Protein(yeast)	1484	10	8	Yes	No	No
Protein(ecoli)	336	8	8	Yes	No	No
Dna(splice)	3190	3	60	No	Yes	No
Credit(German)	1000	2	20	No	Yes	Yes
Credit(Japanese)	125	2	9	Yes	Yes	Yes
Credit(Australian)	690	2	15	Yes	Yes	Yes
Abalone(see notes)	4177	3	8	Yes	Yes	No
Diabetes(Pima)	768	2	8	Yes	No	No
Monks1	556	2	6	No	Yes	No
Sonar	208	2	60	Yes	No	No
Vibration 1	1028	3	9	Yes	No	No
Vibration 2	1862	8	20	Yes	No	No

assigned to S which can be reduced depending upon the strength of the task similarity.

S is used as a metric to judge the task similarity. It may be reduced where appropriate to include hidden units that may contribute towards a useful classification. Those hidden units that are selected are combined with the hiddden units generated from the appropriate second task training data. The hidden units are grouped with the appropriate output class units by calculating a new output weight matrix.

5 Experimental Results

The tasks were organized into seven combinations of training sets. Task G contains all three classes and therefore acts as a control to monitor the effects of transfer. Table 1 lists the contents of each task.

Table 1: Task training set composition

Task	Composition
Task A	Versacolor
Task B	Virginica
Task C	Setosa
Task D	Versacolor + Virginica
Task E	Setosa + Virginica
Task F	Setosa + Versacolor
Task G	Setosa + Virginica + Versacolor

Figure 5 shows the order in which the tasks were performed and the effects of the transfer process in terms of: classification accuracy, number of floating point operations required for training, number of hidden units involved in transfer and the overall result of transfer (positive, negative or zero).

Overall, the process of transfer worked quite well. The first six activities consisted of single class tasks. Activities 1, 2 and 4 had source tasks that were closely related to the target task and were able to contribute hidden units to the second task. Activities 5, 6 and 9 consisted of those source tasks that were too disimilar to the target task and were unable to contibute any hidden units. It would have been possible to reduce the setpoint value S and thus collect some hidden units. However, in practice the value of such units in contributing towards a useful classification is insignificant. Therefore the order in which the tasks are presented is also an important feature of neural network transfer, i.e. the zero transfer activities 5 and 6 are the reverse of positive activities 2 and 3.

Activities 7-12 are more complex consisting of one class task transferred to two class tasks and vice-versa. Activity 13 is a task trained on all three classes and acts as a control to measure the effects of transfer upon the other tasks. Activity 13 (TaskG) is the usual method of training a neural network, i.e. all the training examples were supplied on a single training run. Activity 3 is interesting because its classification accuracy is better than the control task G. This was due to the activity 3 consisting of two classes. The absent third class always causes mis-classification errors.

5.1 Inter-Task Transfer Experiments

This section describes the work performed on inter-task transfer i.e. transfer between entire data sets rather than a decomposed task (intra-task) as that performed on the Iris data set. Unfortunately, in most cases task transfer failed to obtain favorable results.

The task transfer algorithm described in figure 4 was then applied to the other problem domains. It was expected that previous learning on tasks within a related family would give significant training advantages. The tasks were organised into related tasks of training sets, table 2 identifies the contents of each task. Those tasks prefixed with a "U" are unrelated to all other tasks.

The first check was for structural similarity, the preferred situation was to have the same number of inputs and outputs for both tasks. However, a higher

Figure 5: Results of knowledge transfer on Iris dataset

Activity	Task Sequence	Classification Accuracy (%)	Complexity (MFlops)	RBF units Transferred	Total of RBF units	Overall Transfer
1	A → B	90	3.37	16	56	Positive
2	A → C	86	1.68	1	41	Positive
3	B → C	97	1.88	1	41	Positive
4	B → A	75	3.68	19	59	Negative
5	C → A	–	–	0	–	Zero
6	C → B	–	–	0	–	Zero
7	A → E	86	2.4	7	47	Positive
8	B → F	90	3.9	9	49	Positive
9	C → D	–	–	0	–	Zero
10	E → A	88	4.41	20	60	Positive
11	F → B	92	3.99	11	45	Positive
12	D → C	36	3.69	8	48	Negative
13	G	94	7.72	N/A	60	N/A

similarity rating was applied to target tasks with less input features than the source task (compared with the target task having more input features than the source task). This is because it is easier for the hidden units within the target task to "lose" a superfluous input than having to generate a "new" important input feature.

Table 2: Task naming convention and complexity rating

Task id	Domain	Complexity
A1	Xor(bin)	16
A2	Xor(continuous)	16
B1	Vowel(peterson)	347.00
B2	Vowel(deterding)	1509.80
C1	Protein(yeast)	764.57
C2	Protein(ecoli)	287.18
D1	Credit(german)	211.70
D2	Credit(japan)	149.73
D3	Credit(australian)	118.57
E1	Vibration(1)	96.93
E2	Vibration(2)	879.78
U1	Iris	22.23
U2	Housing	102.73
U3	Dna	380.51
U4	Monks1	56.82
U5	Sonar	115.90
U6	Diabetes	630.36

Modifications were made to the original task transfer algorithm. This involved developing a similarity checking algorithm which was used as a pre-processor to task transfer. This new algorithm checked several of the task criteria discussed earlier (structural and complexity similarities). The complexity measure was easily assessed by using equation 1:

$$Complexity = (Ni + Nh + No + Nw2)/(100/N_{acc}) \quad (1)$$

where: Ni is the number of input features, Nh is the number of hidden units, No is the number of output units and Nw2 is the number of hidden to output unit weights (W2). N_{acc} was the accuracy of the network and was given a greater role in determining the complexity than the other parameters.

5.2 Analysis of inter-task transfer

The disapointing results obtained from majority of the inter-task experiments could be traced down to a number of potential sources of error.

- The averaged spread σ values calculated for transferred hidden units were inappropriate. A hidden unit receiving a larger spread than it was trained on is apt to over generalize and give false positives. Conversely, a hidden unit receiving a smaller spread than it was trained on is unlikely to detect the appropriate input patterns and thus generate false negatives.
- The averaged input feature values (μ centres) calculated for the transferred hidden units with "missing" input features were inappropriate. No analysis was performed to verify this hypothesis. However, given the authors knowledge of how spread and centre position values can affect classification accuracy it is likely that this was a particulary damaging source of error.

6 Conclusions

The results of the initial experimental work on intra-task transfer were encouraging. Although it was suspected that the Iris domain may have been too simple to enable useful transfer of knowledge to occur. However, positive transfer did occur in a number of cases because of the decomposition of the Iris data. This enabled the formation of three tasks that had the same number of input features with RBF centre locations that were numerically similar. The main factor likely to prevent the uptake of knowledge transfer by the neural network community would concern to the practicalities of training a network afresh versus the tradeoff between the computational overheads of the transfer process.

Figure 6: Results of knowledge transfer for related tasks

Activity	Task Sequence	Acc (%)	Comp Diff	Symb Diff(%)	RBFs Trans	Total RBFs	Overall Transfer
1	A1 → A2	100:100	Equal	0.0	4	6	Positive
2	A2 → A1	100:100	Equal	0.0	4	4	Positive
3	B1 → B2	86:86	Greater	50.0	9	209	Zero
4	B2 → B1	62:62	Less	0.0	6	36	Zero
5	C1 → C2	87:87	Less	45.0	45	80	Zero
6	C2 → C1	57:57	Greater	35.0	11	131	Zero
7	D1 → D2	93:93	Less	50.0	0	50	Zero
8	D1 → D3	71:71	Less	N/A	0	50	Zero
9	D2 → D1	72:72	Greater	0.0	0	90	Zero
10	D2 → D3	71:71	Less	N/A	0	50	Zero
11	D3 → D1	72:72	Greater	N/A	0	90	Zero
12	D3 → D2	93:93	Greater	N/A	0	50	Zero
13	E1 → E2	94:85	Greater	68.0	12	112	Negative
14	E2 → E1	73:73	Less	0.0	23	46	Zero

References

[Bishop, 1995] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[Ellis, 1965] H. Ellis. *The Transfer of Learning*. The MacMillan Company, New York, 1965.

[Lowe, 1997] D. Lowe. Characterising complexity in a radial basis function network. In *Proceedings of the 5th International Conference on Artificial Neural Networks*, pages 19–23, Cambridge, UK, 1997.

[McCloskey and Cohen, 1989] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: the sequential learning problem. *The Psychology of Learning and Motivation*, 24:109–165, 1989.

[McRae and Hetherington, 1993] K. McRae and P. Hetherington. Catastrophic interference is eliminated in pre-trained networks. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 723–728, Hillsdale, NJ, 1993. Lawrence Erlbaum.

[Moody and Darken, 1989] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, pages 281–294, 1989.

[Pratt, 1993] L. Pratt. *Transfering Previously Learned Back-Propagation Neural Networks to New Learning Tasks*. PhD thesis, Rutgers, State University of New Jersey, May 1993.

[Sharkey and Sharkey, 1993] N. E. Sharkey and A. J. C. Sharkey. Adaptive generalization. *Artificial Intelligence Review*, 7:313–328, 1993.

Input:
- Source task A network parameters
- Source task A training data
- Target task B training data
- Set-point S
- Gaussian radius spread σ

Output:
- Target task B network
- Hidden units from task A

Procedure:

```
Train source task on A data
Set-point = upper value
Apply task B data to source network A
While set-point ≥ lower value
        If Task A hidden unit activations ≥ S
          Save hidden units
        Else
          Decrement S
If Hidden units found
  Extract hidden unit parameters
  Train task B network on task B data
  Merge extracted units with task B network
  Adjust σ for all RBF centers
  Compute new output unit weights
  Save final network
Else
  Exit program
```

Figure 4: Knowledge transfer algorithm