Context-sensitive MTL Networks for Machine Lifelong Learning

Daniel L. Silver and **Ryan Poirier***
Jodrey School of Computer Science
Acadia University
Wolfville, NS, Canada B4P 2R6
danny.silver@acadiau.ca

Abstract

Context-sensitive Multiple Task Learning, or *cs*MTL, is presented as a method of inductive transfer that uses a single output neural network and additional contextual inputs for learning multiple tasks. The *cs*MTL method is tested on three task domains and shown to produce hypotheses for a primary task that are significantly better than standard MTL hypotheses when learning in the presence of related and unrelated tasks. A new measure of task relatedness, based on the context input weights, is shown to have promise. The paper also outlines a machine lifelong learning system that uses *cs*MTL for sequentially learning multiple tasks. The approach satisfies a number of important requirements for knowledge retention and inductive transfer including the elimination of redundant outputs, representational transfer for rapid but effective short-term learning and functional transfer via task rehearsal for long-term consolidation.

Introduction

Multiple task learning (MTL) neural networks are one of the better documented methods of inductive transfer of task knowledge (Caruana 1997; Silver & Mercer 1997). An MTL network is a feed-forward multi-layer network with an output for each task that is to be learned. The standard back-propagation of error learning algorithm is used to train all tasks in parallel. Consequently, MTL training examples are composed of a set of input attributes and a target output for each task. Figure 1 shows a simple MTL network containing a hidden layer of nodes that are common to all tasks. The sharing of internal representation is the method by which inductive bias occurs within an MTL network (Baxter 1996). The more that tasks are related, the more they will share representation and create positive inductive bias.

Formally, let X be a set on $\Re^n$ (the reals), Y the set of $\{0, 1\}$ and $error$ a function that measures the difference between the expected target output and the actual output of the network for an example. Then for single task learning (STL), the target concept is a function f that maps the set X to the set Y, $f : X \rightarrow Y$, with some probability distribution P over $X \times Y$. An example for STL is of the form $(\mathbf{x}, f(\mathbf{x}))$, where $\mathbf{x}$ is a vector containing the input values $x_1, x_2, \ldots, x_n$ and $f(\mathbf{x})$ is the target output. A training set S_{STL} consists of all available examples, $S_{STL} = \{(\mathbf{x}, f(\mathbf{x}))\}$. The objective of the STL algorithm is to find a hypothesis h within its hypothesis space H_{STL} that minimizes the objective function, $\sum_{x \in S_{STL}} error\,[f(\mathbf{x}), h(\mathbf{x})]$. The assumption is that $H_{STL} \subset \{f | f : X \rightarrow Y\}$ contains a sufficiently accurate h.

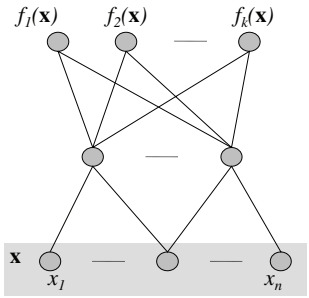

Figure 1: A multiple task learning (MTL) network with an output node for each task being learned in parallel.

MTL can be defined as learning a set of target concepts $\mathbf{f} = \{f_1, f_2, \ldots f_k\}$ such that each $f_i : X \rightarrow Y$ with a probability distribution P_i over $X \times Y$. We assume that the environment delivers each f_i based on a probability distribution Q over all P_i. Q is meant to capture some regularity in the environment that constrains the number of tasks that the learning algorithm will encounter. Q therefore characterizes the domain of tasks to be learned. An example for MTL is of the form $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$, where $\mathbf{x}$ is the same as defined for STL and $\mathbf{f}(\mathbf{x}) = \{f_i(\mathbf{x})\}$, a set of target outputs. A training set S_{MTL} consists of all available examples, $S_{MTL} = \{(\mathbf{x}, \mathbf{f}(\mathbf{x}))\}$. The objective of the MTL algorithm is to find a set of hypotheses $\mathbf{h} = \{h_1, h_2, \ldots, h_k\}$ within its hypothesis space H_{MTL} that minimizes the objective function $\sum_{x \in S_{MTL}} \sum_{i=1}^{k} error\,[f_i(\mathbf{x}), h_i(\mathbf{x})]$. The assumption is that H_{MTL} contains sufficiently accurate h_i for each f_i being learned. Typically $|H_{MTL}| > |H_{STL}|$ in order to represent the multiple hypotheses.

*This research has been funded in part by the Government of Canada through NSERC.

Limitations of MTL for Lifelong Learning

Previously, (Silver & Mercer 2002; Silver & Poirier 2004; O'Quinn, Silver, & Poirier 2005) have investigated the use of MTL networks as a basis for developing a machine lifelong learning (ML3) system and have found them to have several limitations related to the multiple outputs of the network. First and foremost, MTL requires that training examples contain corresponding target values for each task; this is impractical for lifelong learning systems as examples of each tasks are acquired at different times and with unique combinations of input values. We have examined methods of generating corresponding target values but have found weaknesses related to the differences in the distribution of examples over the input space for various tasks.

With MTL, shared representation is limited to the hidden node layer and is not possible at the output nodes. The theory is that optimal inductive transfer occurs when related tasks share the same hidden nodes. This perspective does not consider the sharing of knowledge at the example level and in the context of unrelated tasks. Consider two concept tasks where half of the MTL training examples have identical target values. From a MTL task-level perspective, using most statistical and information theoretic measures, these sets of training examples would be considered unrelated and of little value to each other for inductive transfer.

There is also the practical problem of how a MTL based lifelong learning agent would know to associate an example with a particular task. Clearly, the learning environment should provide the contextual queues, however this suggests additional inputs and not outputs. A related problem is managing the redundant representation that can develop for the same task in an ML3 system based on MTL. A lifelong learning system should be capable of practising a task and improving its model with new examples over time. However, because there are is no task context queue, an MTL based ML3 system requires a separate output for each new set of training examples. It is unclear how this build up of redundant task outputs over time can be handled.

In response to these problems, we have developed *context-sensitive* MTL, or *cs*MTL. *cs*MTL is based on standard MTL with two major differences; only one output is used for all tasks and additional inputs are used to indicate the example *context*, such as the task to which it is associated. The following section describes the *cs*MTL network. The remaining sections present a ML3 system based on a *cs*MTL network, discuss its theoretical benefits and limitations and report on experiments that test its performance.

*cs*MTL

Figure 2 presents the *cs*MTL network. It is a feed-forward network architecture of input, hidden and output nodes that uses the back-propagation of error training algorithm. The *cs*MTL network requires only one output node for learning multiple concept tasks (more outputs could be used for predicting a vector of values for each task). Similar to standard MTL neural networks, there is one or more layers of hidden nodes that act as feature detectors. The input layer can be divided into two parts: a set of *primary* input variables for

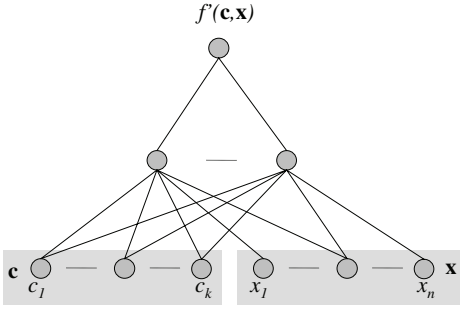

Figure 2: Proposed system: *cs*MTL

the tasks and a set of inputs that provide the network with the *context* of each training example. The context inputs can simply be a set of task identifiers that associate each training example to a particular task. Alternatively, they can offer more specific environmental information (such as location and light level) and in this way index over a continuous domain of tasks. Related work on context-sensitive machine learning can be found in (Turney 1996).

Formally, let C be a set on $\Re^n$ representing the context of the primary inputs from X as described for MTL. Let $\mathbf{c}$ be a particular instance of this set where $\mathbf{c}$ is a vector containing the values $c_1, c_2, \ldots, c_k$; where $c_i = 1$ indicates that the example is associated with function f_i. *cs*MTL can be defined as learning a target concept $f' : C \times X \rightarrow Y$; with a probability distribution P' on $C \times X \times Y$ where P' is constrained by the probability distributions P and Q discussed in the previous section for MTL. An example for *cs*MTL takes the form $(\mathbf{c}, \mathbf{x}, f'(\mathbf{c}, \mathbf{x}))$, where $f'(\mathbf{c}, \mathbf{x}) = f_i(\mathbf{x})$ when $c_i = 1$ and $f_i(\mathbf{x})$ is the target output for task f_i. A training set S_{csMTL} consists of all available examples for all tasks, $S_{csMTL} = \{(\mathbf{c}, \mathbf{x}, f'(\mathbf{c}, \mathbf{x}))\}$. The objective of the *cs*MTL algorithm is to find a hypothesis h' within its hypothesis space H_{csMTL} that minimizes the objective function, $\sum_{x \in S_{csMTL}} error\,[f'(\mathbf{c}, \mathbf{x}), h'(\mathbf{c}, \mathbf{x})]$. The assumption is that $H_{csMTL} \subset \{f | f : C \times X \rightarrow Y\}$ contains a sufficiently accurate h'. Typically, $|H_{csMTL}| = |H_{MTL}|$ for the same set of tasks because the number of additional context inputs under *cs*MTL matches the number of additional task outputs under MTL.

With *cs*MTL, the entire representation of the network is used to develop hypotheses for all tasks, $f'(\mathbf{c}, \mathbf{x})$, following the examples drawn according to P'. The focus shifts from learning a subset of shared representation for multiple tasks to learning a completely shared representation for the same tasks. This presents a more continuous sense of domain knowledge and the objective becomes that of learning internal representations that are helpful to predicting the output of similar combinations of the primary and context input values. Therefore, the important concept of relatedness shifts from the task level to the example level. During learning, $\mathbf{c}$ selects an inductive bias over H_{csMTL} relative to the secondary tasks being learned in the network. Once f' is learned, if $\mathbf{x}$ is held constant, then $\mathbf{c}$ indexes over the hypothesis base H_{csMTL}. If $\mathbf{c}$ is a vector of real-valued inputs and from the environment, it provides a grounded sense of task relatedness. If $\mathbf{c}$ is a set of task identifiers, it differentiates

between otherwise conflicting examples and selects internal representation used by related tasks.

In the following section we propose how *cs*MTL can be used to overcome the limitations of standard MTL for construction of a ML3 system. The proposed ML3 is described so as to provide addition motivation for and useful characteristics of *cs*MTL.

*cs*MTL and Machine Lifelong Learning

Figure 3 shows the proposed *cs*MTL ML3 system. It has two components; a temporary *short-term learning network* and a permanent *long-term consolidation cs*MTL *network*. The long-term *cs*MTL network is the location in which domain knowledge is retained over the lifetime of the learning system. The weights of this network are updated only after a new task has been trained to an acceptable level of accuracy in the short-term learning network. The short-term network can be considered a temporary extension of the long-term network that adds representation (several hidden nodes and a output node, fully feed-forward connected) that may be needed to learn the new task. At the start of short-term learning the weights associated with these temporary nodes are initialized to small random weights while the weights of the long-term network are frozen. This allows representational knowledge to be rapidly transferred from related tasks existing in the long-term network without fear of losing prior task accuracies.

Once the new task has been learned, the temporary short-term network is used to consolidate knowledge of the task into the permanent long-term *cs*MTL network. This is accomplished by using a form of functional transfer called *task rehearsal* (Silver & Mercer 2002). The method uses the short-term network to generate *virtual examples* for the new tasks so as to slowly integrate (via back-propagation) the task's knowledge into the long-term network. Additionally, virtual examples for the prior tasks are used during consolidation to maintain the existing knowledge of the long-term network. Note that it is the functional knowledge of the prior tasks that must be retained and not their representation; the internal representation of the long-term network will necessarily be updated as the new task in integrated.

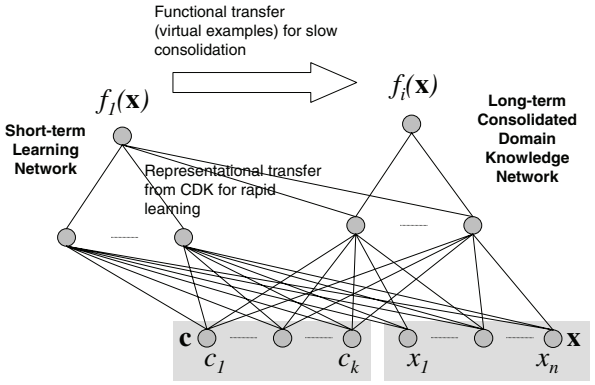

Figure 3: Proposed system: *cs*MTL

Benefits and Limitations

The following discusses the benefits and limitations of the proposed *cs*MTL method of life-long learning.

Long-term Retention of Learned Knowledge.

Knowledge retention in a MTL network is the result of consolidation of new and prior task knowledge using task rehearsal (Silver & Mercer 2002). Task rehearsal overcomes the stability-plasticity problem originally posed by (Grossberg 1987) taken to the level of learning sets of tasks as opposed to learning sets of examples (Robins 1995; French 1997). Consolidation of new task knowledge without loss of existing task knowledge is possible given sufficient number of training examples, sufficient internal representation for all tasks, slow training using a small learning rate and a method of early stopping to prevent over-fitting and therefore the growth of high magnitude weights (Silver & Poirier 2004).

In the long-term *cs*MTL network there will be an effective and efficient sharing of internal representation between related tasks, as in the case of an MTL network, without the disadvantage of having duplicate representation of identical or near identical task outputs. Over time, more detailed practice sessions for the same task will contribute to the development of a more accurate long-term hypothesis. Learning closely related tasks will fill in useful knowledge of the domain. In fact, the *cs*MTL network can represent a fluid domain of tasks where subtle differences between tasks can be represent by small changes in the context inputs.

Our conjecture is that *cs*MTL does not require an explicit method of indexing into domain knowledge for related tasks. Instead, the internal representation of all tasks saved in the long-term network is used (held fixed) as a portion of the new hypothesis. Indexing occurs as the connection weights between the long-term network and the temporary short-term network are trained.

The *cs*MTL ML3 approach does have its limitations. It suffers from the scaling problems of similar neural network systems. The computational complexity of the standard back-propagation algorithm is $O(W^3)$, where W is the number of weights in the network. Long-term consolidation will be computationally more expensive than standard MTL because the additional contextual inputs will increase the number of weights in the network at the same rate as MTL and it may be necessary to add an additional layer of hidden nodes for certain task domains. The rehearsal of each of the existing domain knowledge tasks requires the creation and training of $m \cdot k$ virtual examples, where m is the number of virtual training examples per task and k is the number of tasks. An important benefit from consolidation is an increase in the accuracy of related hypotheses existing in the *cs*MTL network as a new task is integrated.

Short-term Learning with Inductive Transfer

An inductive transfer system should produce a hypothesis for the primary task that meets or exceeds the generalization performance of hypotheses developed strictly from the training examples. The proposed *cs*MTL ML3 system uses a temporary short-term learning network with representational transfer from the long-term consolidation network. This form of transfer should be efficient and effective. If the current task has been previously learned and retained, then the weights between the long-term network and the short-term network will train quickly to produce the desired output. If the new task is different but related to a prior task, the long-term to short-term network weights will select the most appropriate features of domain knowledge and the supplemental hidden nodes of the short-term network will play only a partial role in the hypothesis. If the new task is unrelated to any prior learning, the supplemental internal representation of the short-term network will play the major role in the new hypothesis.

Although the computational cost of long-term consolidation is high, the benefit is that a hypothesis for a new but related task can be quickly developed in the short-term network. The reasons for this are: the existing internal representation of the long-term network can be used to develop the hypothesis, only the new task training examples are required, and there are relatively few weights in the temporary short-term network to be trained.

Experimentation

This section reports on a set of initial experiments that compares the ability of *cs*MTL to transfer knowledge with MTL and ηMTL, a variant of MTL that selects the most related task knowledge based on correlation of the target outputs (Silver & Mercer 2002). All experiments use a *cs*MTL network as described in the csMTL section. Empirical studies of a lifelong learning system with short-term and long-term components, as described in the previous section, will be reported in a future article.

Three domains have been studied using *cs*MTL. The *Band domain*, described in (Silver & Mercer 2002), consists of seven synthetic tasks. Each task has a band of positive examples across a 2-dimensional input space. The tasks were synthesized so that the primary task T_0 would vary in its relatedness to the other tasks based on the band orientation. The *Logic domain*, described in (McCracken 2003) consists of six synthetic tasks. Each positive example is defined by a logical combination of 4 of the 10 real-valued inputs of the form, $T_0 : (I_0 > 0.5 \vee I_1 > 0.5) \wedge (I_2 > 0.5 \vee I_3 > 0.5)$. The tasks of this domain are more or less related in that they share zero, one or two features such as $(I_0 > 0.5 \vee I_1 > 0.5)$ with the other tasks. The Band and Logic domains have been designed so that all tasks are non-linearly separable; each task requires the use of at least two hidden nodes of a neural network to form an accurate hypothesis. The *fMRI domain* challenges the learning systems to develop models that can classify 24 features extracted from fMRI images as a subject reading a sentence or viewing a picture[1]. Inductive transfer between two subject models is examined; from subject T_1 for which good models could be developed to a second subject T_0 for which only poor models could be developed.

Method

A *cs*MTL network was configured for each domain with one output node, a layer of hidden nodes (30 for the Band, 20 for the Logic and 10 for the fMRI domain) and a layer of input nodes. The Band domain has 9 inputs, 2 represent the coordinates of the 2-dimensional input space and the remaining 7 provide the *context*, that is, they indicate the task to which each example belongs. The Logic domain has 16 inputs, 10 represent the primary values for logical expression and the remaining 6 provide the task *context*. The fMRI domain has 26 inputs, where 24 are used to represent the activity level of a region of interest in the subjects brain and the remaining 2 indicate subject T_0 or T_1.

For all three domains, the objective is to learn task T_0 using an impoverished training set of examples (10 for the Band, 30 for the Logic and 48 for the fMRI domain) for which single task learning (STL) does poorly. Each of the other tasks of the domain have 48 or more training examples that have been demonstrated to develop models with accuracies greater than .75 using a STL network. A tuning set of examples (10 for the Band, 20 for the Logic and 8 for the fMRI domain) is used to prevent over-fitting on each domain. An independent test set (200 for the Band, 1000 for the Logic and 24 for the fMRI domain) was used to determine hypothesis performance.

The mean accuracies reported are from repeated studies (10 for the Band, 30 for the Logic domain and 5 for the fMRI domain).

Results

Figure 4 shows the results for the three domains. It compares the mean accuracy of the *cs*MTL hypotheses developed for the T_0 tasks to hypothesis developed with no inductive transfer under STL, with transfer under standard MTL, and selective transfer under ηMTL. The MTL and ηMTL results demonstrate the advantage of knowledge transfer with mean accuracies that are significantly better for all domains of tasks. *cs*MTL does significantly better than STL and MTL on all domains, equal in performance to ηMTL on the Band and Logic domains and significantly better than ηMTL on the fMRI domain. The results indicate that *cs*MTL is able to selectively transfer knowledge from the shared internal representation to a new task when training on examples of *all* prior tasks without the aid of a measure of task relatedness.

Discussion

A brute force study of learning T_0 in an *cs*MTL network with each of the secondary tasks has shown that the most accurate hypotheses are developed when inductive transfer occurs from T_1 or T_2 for the Logic domain. We therefore consider these tasks to be the most related to the primary T_0 task.

[1]Courtesy of the Brain Image Analysis Research Group and CALD, Carnegie Mellon University.

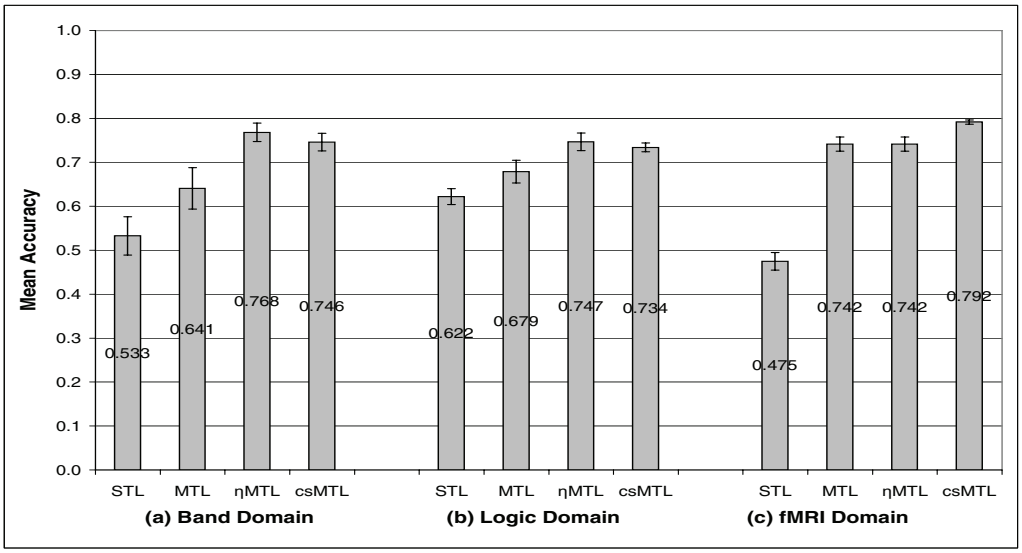

Figure 4: *cs*MTL compared to STL and previous MTL methods. Shown is the mean test set accuracy for T_0 hypotheses for the three domains of tasks.

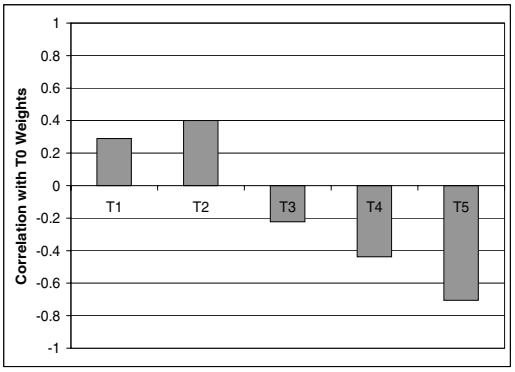

Figure 5: Correlation similarity of context input-to-hidden node weights in *cs*MTL networks developed for Logic (left) and Band (right) domains.

In the *cs*MTL section we conjecture that relatedness between tasks can be measured by the similarity of the input to hidden node weights for each c_i, if $\mathbf{c}$ is a set of task identifiers. To test this we examined the *cs*MTL network weights developed for the tasks of the Logic domain. Specifically, we computed the linear correlation of the weight vector associated with the T_0 context input to the corresponding weight vectors for the other tasks. Figure 5 shows the correlation values in the form of bar graphs. The left graph of the Logic domain shows that T_1 and T_2 weights are the only two tasks that are positively correlated with the T_0 weights. This agrees with the brute force sense of relatedness.

Conclusion

This paper has presented *cs*MTL as a method of inductive transfer that uses a single output neural network and additional *context* inputs for learning multiple tasks. The method was developed in response to problems we had encounter in using MTL networks for developing machine lifelong learning (ML3) systems. The question of how an example is associated with one task versus another is solved by the context inputs. The operator (or the environment) can provide these contextual queues with each example. The method eliminates the build-up of redundant task representation that can frustrate the search for related prior knowledge. Lastly, and perhaps most importantly, the *cs*MTL approach shifts the focus from learning a subset of shared representation for multiple tasks to learning a completely shared representation for the same tasks. The context inputs can be seen as indexing over that domain at the example level as opposed to the task level. Our conjecture is that this approach avoids the issue of having to measure the relatedness between tasks in order to ensure a positive inductive bias. Similar examples of the primary task will collaborate with similar examples of related tasks to build mutually beneficial internal representation. Dissimilar examples will work to develop unique representations that capture the subtleties of the individual tasks. We are currently examining the theory of *Hints* (Abu-Mostafa 1995) for direction on formalizing the notion that each separate task can be seen as a Hint that reduces the VC dimension for learning the internal representation of related tasks within the domain.

Experimentation on three different domains of tasks has demonstrated that *cs*MTL can produce hypotheses that are significantly better than standard MTL hypotheses when learning a primary task in the presence of related and unrelated tasks. The results indicate that *cs*MTL is able to

selectively transfer knowledge from shared internal representation to a new task without the aid of a measure of task relatedness. An analysis of the *context* input-to-hidden node weight vectors indicates that similarity between weight vectors provides a promising measure of task relatedness should it be needed.

The paper also describes a ML3 system based on *cs*MTL that is capable sequential knowledge retention and inductive transfer. The system is meant to satisfy a number of ML3 requirements including the effective consolidation of task knowledge into a long-term network using task rehearsal, the accumulation of task knowledge from practice sessions, effective and efficient inductive transfer during new learning, and the tradeoff between using inductive transfer and the available training examples during new learning. We are currently developing a *cs*MTL ML3 system capable of sequential learning and plan to conduct experiments on informative synthetic and real-world domains so as to more fully explore the approach.

References

Abu-Mostafa, Y. S. 1995. Hints. *Neural Computation* 7:639–671.

Baxter, J. 1996. Learning model bias. In Touretzky, D. S.; Mozer, M. C.; and Hasselmo, M. E., eds., *Advances in Neural Information Processing Systems*, volume 8, 169–175. The MIT Press.

Caruana, R. A. 1997. Multitask learning. *Machine Learning* 28:41–75.

French, R. M. 1997. Pseudo-recurrent connectionist networks: An approach to the sensitivity-stability dilemma. *Connection Science* 9(4):353–379.

Grossberg, S. 1987. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science* 11:23–64.

McCracken, P. 2003. *Selective Representational Transfer Using Stochastic Noise*. Wolfville, NS, Canada: Honurs Thesis, Jodrey School of Computer Science, Acadia University.

O'Quinn, R.; Silver, D. L.; and Poirier, R. 2005. Continued practice and consolidation of a learning task. In *Proceedings of the Meta-Learning Workshop, 22nd International Conference on Machine Learning (ICML 2005)*.

Robins, A. V. 1995. Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science* 7:123–146.

Silver, D. L., and Mercer, R. E. 1997. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Learning to Learn* 213–233.

Silver, D. L., and Mercer, R. E. 2002. The task rehearsal method of life-long learning: Overcoming impoverished data. *Advances in Artificial Intelligence, 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI'2002)* 90–101.

Silver, D. L., and Poirier, R. 2004. Sequential consolidation of learned task knowledge. *Lecture Notes in Artificial Intelligence, 17th Conference of the Canadian Society for Computational Studies of Intelligence (AI'2004)* 217–232.

Turney, P. D. 1996. The identification of context-sensitive features: A formal definition of context for concept learning. In *13th International Conference on Machine Learning (ICML96), Workshop on Learning in Context-Sensitive Domains*, volume NRC 39222, 53–59.