Memory Organization and Knowledge Transfer

David J. Stracuzzi STRACUDJ@CSLI.STANFORD.EDU
Center for the Study of Learning and Information, Computational Learning Laboratory,
Stanford University, CA 94305 USA

Abstract

An important aspect of learning is the ability to transfer knowledge from one domain to another. Recent transfer research has focused on the basic problem of how knowledge structures may be transferred and reused. In this paper, we consider the larger problem of how a learner can select the appropriate knowledge structures to transfer when many are available. We propose that previously acquired knowledge must be organized, and demonstrate one possible approach.

1. Introduction

The ability to transfer knowledge from one domain to another is an important aspect of learning. Knowledge transfer helps to increase learning efficiency by freeing the learner from duplicating past efforts. Transfer also allows the learner to acquire knowledge that may otherwise be unattainable or intractable. A learner can use previously acquired knowledge to bootstrap itself toward increasingly complex knowledge.

Recent research into knowledge transfer focuses on the basic problem of how to transfer knowledge structures from one domain to another. In this paper, we consider a longer view. Suppose an agent has many existing knowledge structures from which it can draw while learning in a new domain. The problem then becomes one of deciding which existing knowledge is relevant.

Moreover, suppose the learner is to perform on many interrelated learning tasks. The transfer problem then expands to selecting relevant knowledge from an expanding knowledge base. The learner must consider all previously acquired knowledge as potential sources for transfer. The number of possible transfer combinations considered by the learner grows, and the transfer learning problem becomes more computationally expensive, with each new task. Learning based on this method cannot scale up. In this paper, we propose that the results of prior learning must be organized in order for the agent to take full advantage of its acquired knowledge.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. The Memory Organization Problem

The purpose of memory organization is to reduce the complexity of learning without reducing the learner's ability transfer knowledge among domains. The goal then is to provide a set of candidate knowledge dependencies for the learner. In this paper, dependency refers to a "uses" relationship between two concepts, and knowledge structure refers to a concept hierarchy. Transfer of a knowledge structure implies discovering a dependency between source and target concepts.

To allow for useful redundancy in representation and flexibility in learning, the definition of *related* should be relatively broad. In other words, we are not concerned with finding a minimal set of knowledge structures for transfer. This places two constraints on memory organization methods.

The first constraint stems from the goal of designing a system capable of long-term learning. The number of existing knowledge structures may become quite large, so the organizational method must scale up. Specifically, the set of candidates available for transfer is equal to the set of all existing knowledge structures times the number of ways that the learner can manipulate these structures to fit the new domain. This size of this set can quickly become quite large. Explicit comparisons between all possible candidates and the new target becomes impractical.

The second constraint arises from the learning and organizational process itself. The set of candidate knowledge structures is drawn from a variety of sources and changes regularly as the learner adds new knowledge. Thus, there may be little common information among the individual instances of the knowledge transfer problem. This rules out or renders inefficient many popular organizational strategies, such as clustering (see Dempster et al., 1977) and feature selection (see Kohavi & John, 1997).

3. A Memory Organization Method

In this section we present the SCALE algorithm, which is designed to learn a hierarchy of relational concepts from an online stream of data (Stracuzzi, 2005). Each training point in the stream includes one supervised example of one concept, such that examples of all concepts in the hierarchy arrive in random order. The goal then, is to learn this concept hierarchy, including definitions for and dependencies among the concepts.

A key aspect of SCALE is that each concept must be representable as a simple function (for example, a linear threshold unit) of its inputs. Here, *inputs* refers to both input variables and to the other concepts on which a given concept depends. This restriction to simple functions forces the learner to acquire concepts in a bottom-up fashion, from simple to complex (Utgoff & Stracuzzi, 2002). With respect to knowledge transfer, this means that knowledge is always transferred vertically in the concept hierarchy.

3.1. Learning Environment

We state the specific transfer learning problem considered by SCALE as follows. The agent is situated in an environment described by m input variables $\alpha_1 \cdots \alpha_m$. The task is to learn n relational Boolean concepts $\beta_1 \cdots \beta_n$. Supervised training data $\mathcal{X}$ arrives in an online fashion. Each $X_t \in \mathcal{X}$ provides values for the input variables $\vec{\alpha}$ and the desired output of exactly one target concept $\beta_i(\vec{\alpha})$. Examples for the n target concepts arrive in random order. From this stream of data, the learner must produce a representation of the concept hierarchy which may be used to predict accurately the values of novel, unlabeled examples.

3.2. Algorithm

We include here a brief summary of the SCALE algorithm. A more complete description may be found in Stracuzzi (2005). We begin by defining two important terms. A relational concept (predicate) P_i is considered *learned* if it has successfully acquired its target concept β_i. Likewise, P_i is *unlearnable* if it cannot acquire β_i given its current set of dependencies (one or more necessary dependencies is missing). In practice, the metrics that support these definitions have been determined empirically, and depend on the number of P_i's dependencies, and the number of examples seen.

Table 1. The SCALE algorithm.

```
Given:
  Example $X_t$ for concept $\beta_i$

Algorithm: SCALE-UPDATE($X_t$)
  $P_i \leftarrow$ predicate for target concept $\beta_i$
  while $P_i \notin \mathcal{L}$ do
    if $P_i$ not learned then
      update $P_i$ using $X_t$
      if updated $P_i$ learned then
        $\mathcal{L} \leftarrow \mathcal{L} + \{P_i\}$
      else if updated $P_i$ unlearnable then
        select new dependency from $P_{j<i} \in \mathcal{L}$
      else
        infer all concepts $P_{j<i} \in \mathcal{L}$
        update conditional probabilites for $P_{j<i}$
    else if $P_i$ not done removing inputs then
      ELIMINATE-IRRELEVANT-INPUTS($P_i, X_t$)
    else
  end while
```

SCALE begins by initializing the set of *learned* predicates $\mathcal{L} = \emptyset$, and then receiving the training data $\mathcal{X}$. For each training example X_t (which describes some β_i) SCALE updates the corresponding predicate P_i. Initially P_i has no dependencies on other predicates, and learns from only the input variables $\vec{\alpha}$. After P_i is updated, the output values of all predicates $P_{j<i} \in \mathcal{L}$ are inferred based on X_t. Using these inferred values, a simple memory organization metric, computed as conditional probability measures $\Pr[P_i = \mathit{true} | P_{j<i} = \mathit{true}, \text{Data}]$ and $\Pr[P_i = \mathit{true} | P_{j<i} = \mathit{false}, \text{Data}]$, are updated for each $P_{j<i}$.

If P_i is *unlearnable*, SCALE selects one or more new dependencies according to the conditional probability measures (larger values preferred). P_i then continues to train using the new, expanded set of dependencies. This process continues until P_i is *learned.* At this point, P_i is added to $\mathcal{L}$. Subsequent training examples for concept β_i are used to remove any unnecessary or redundant dependencies from P_i using the randomized variable elimination algorithm (Stracuzzi & Utgoff, 2004). Removing unnecessary dependencies improves P_i's generalization, and makes subsequent inference computations run faster.

Training in SCALE stops when all predicates are both *learned* and pruned, or if every predicate that is not *learned* is also *unlearnable.* The first condition signifies that both learning and memory organization are complete, while the second condition signifies that the learner is unable to find a viable representation for all

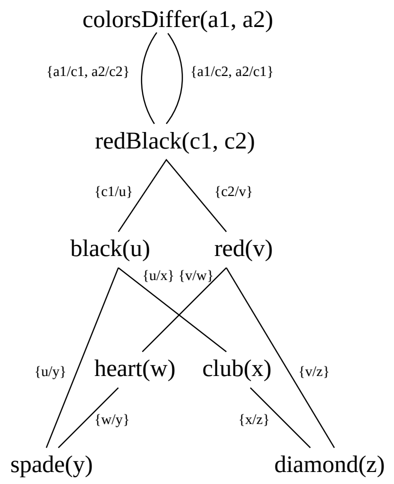

Figure 1. Partial concept hierarchy for solitaire.

of the concepts. The SCALE update algorithm for an individual training example is shown in Table 1.

Note that SCALE attempts to learn all target concepts simultaneously. Simpler concepts are acquired first, and become available as possible dependencies for more complex concepts. The algorithm therefore explores possible transfer options between all learned predicates and all unlearned predicates.

4. Experiments

The objective of the following experiments is to demonstrate the importance and effects of memory organization on knowledge transfer. Using a card stackability domain based on freecell solitaire, we demonstrate that even the simple organizational methods used by SCALE improve transfer of knowledge from one concept to another by reducing dimensionality and increasing the accuracy of acquired knowledge.

Toward this end, the performance of SCALE is compared against an alternate version of the algorithm, called CUMULUS, which considers all learned predicates as possible sources for transfer. CUMULUS therefore attempts to perform the same transfer learning task as SCALE, but without any memory organization.

4.1. Card Solitaire Domain

The card solitaire domains consists of 19 concepts. The two high-level concepts test whether one card may be stacked on top of another in freecell solitaire under two different circumstances. Other concepts in the hierarchy play supporting roles by testing for relations among the suit, color and rank of card pairs.

The low-level input variables are rudimentary. Each card in the deck is represented by a single integer in the range 0–51. The cards are indexed such that the spades ace through king are indexed 0–12, the hearts ace through king are indexed 13–25, and likewise for clubs and diamonds. The learner must therefore learn concepts to map this integer representation into suits, colors and ranks. We choose this representation because it adds depth to the concept hierarchy and makes the learning problem more complex. Figure 1 shows a subset of the solitaire concept hierarchy.

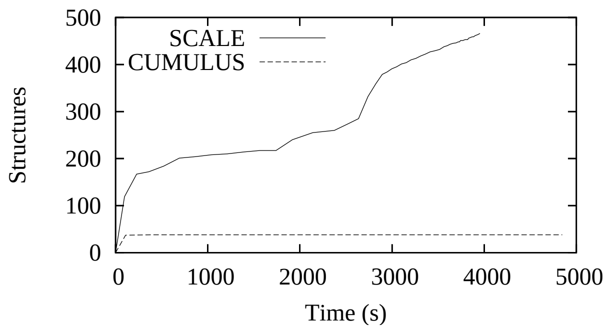

Figure 2. Number of knowledge structures considered.

There are a total of 34788 unique examples for the 19 solitaire concepts. Recall that each example is a supervised instance of exactly one of the 19 target concepts. We set aside approximately five percent of this data (1751 examples) for training, and reserved the remainder for testing purposes. The training examples were given to SCALE and CUMULUS one at a time, in random order. We then tested the accuracy of the algorithms on all 19 concepts at regular intervals during training to establish learning curves. We also kept track of the number of distinct knowledge structures (hypotheses) considered by each algorithm, along with the complexity of each knowledge structure.

4.2. Results

Figure 2 plots the number of knowledge structures considered by SCALE and CUMULUS against time. SCALE considers 466 unique hypotheses over the course of training, while CUMULUS considers just 38. This difference follows directly from the absence of memory organization in CUMULUS. All *learned* predicates are immediately added as dependencies to any *unlearnable* predicate. This means that CUMULUS does not consider many hypotheses in which a predicate depends on only a subset of the learned predicates.

Although a smaller search space can be enviable in machine learning, the lack of breadth is detrimental here. CUMULUS only manages to learn 15 of the 19 concepts; the remaining four are determined to be unlearnable. The learner is bogged down by the large number of (mostly irrelevant) dependencies in

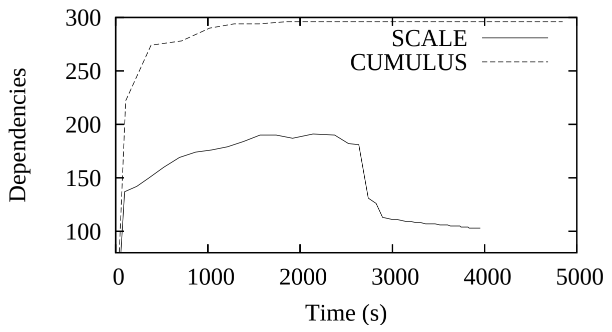

Figure 3. Knowledge structure complexity.

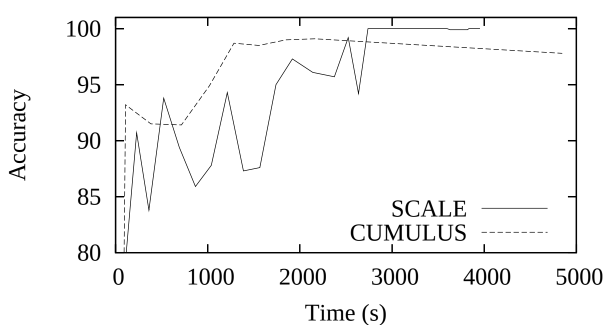

Figure 4. Comparison of learning curves.

the knowledge structure.

Figure 3 illustrates this point by showing the number of dependencies in the knowledge structures used by the two algorithms. Notice how CUMULUS works with very large and complex structures almost immediately. In contrast, SCALE adds dependencies slowly, and ultimately manages to remove almost half of the dependencies added.

Generalization results were somewhat surprising. Figure 4 suggests that memory organization may slow down the learning process in the short term. SCALE takes more time (and more examples) to reach the early performance levels achieved by CUMULUS. However, in the longer term, SCALE reaches higher accuracy rates when CUMULUS becomes bogged down by the combinatorics of its learning problem. We expect that the importance of memory organization increases with the number of target concepts.

5. Discussion and Future Work

Flexibility is a critical aspect of memory organization. The structure of a learner's knowledge must permit a broad range of relationships among target problems. The alternative is redundant learning and recreation of existing knowledge. In SCALE, knowledge structures are not explicitly clustered or physically configured, although dependencies among concepts are explicitly represented. This approach has the advantage of being both sufficiently structured to provide learning efficiency, yet unrestrictive with respect to opportunities for knowledge transfer.

One important direction for future research concerns SCALE's inability to revise the representation of acquired knowledge. After a concept is learned, its representation remains fixed. Although this approach does not preclude SCALE from learning any needed structure, this lack of flexibility may prevent the algorithm from finding the most compact representation. Alternate dependency structures may be possible and may lead to more efficient structures, or a wider range of knowledge transfer.

6. Conclusion

Memory organization plays a critical role in knowledge acquisition. A learner must be able to transfer existing knowledge to new learning problems. However, distinguishing related and unrelated knowledge is a challenging task. In the absence of memory organization, every piece of existing knowledge must be considered as a possible source for transfer. The result is a situation in which each new learning problem becomes more complex than the previous. The SCALE algorithm uses a combination of implicit and explicit organizational techniques to improve the learner's ability to use existing knowledge to acquire new knowledge.

References

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum liklihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B, 39*, 1–38.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Stracuzzi, D. J. (2005). Scalable knowledge acquisition through memory organization. *International and Interdisciplinary Conference on Adaptive Knowledge and Reasoning (AKRR 2005)* (pp. 57–64). Espoo, Finland: Helsinki University of Technology.

Stracuzzi, D. J., & Utgoff, P. E. (2004). Randomized variable elimination. *Journal of Machine Learning Research*, *5*, 1331–1364.

Utgoff, P. E., & Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation*, *14*, 2497–2529.