Compositional Classification

Joshua Jones
College of Computing
Georgia Institute of Technology
Atlanta, USA 30332
jkj@cc.gatech.edu

March 25, 2008

Abstract

An intelligent system must routinely deal with massive information processing complexity. The research discussed in this document is concerned with finding representations and processes to deal with a part of this complexity. At a high level, the proposed idea is that a synthesis between the symbolic reasoning of classic artificial intelligence research and the statistical inference mechanisms of machine learning provides answers to some of these issues of complexity. This research is specifically concerned with a subset of classification problems that we call "compositional classification", where both the class label and values produced at internal nodes in the classification structure entail verifiable predictions. This research specifies and evaluates a technique for compositional classification. This investigation will consist of (i) implementing a framework for the construction of supervised classification learning systems that codifies the technique, (ii) instantiating a number of learning systems for various specific classification problems using the framework, (iii) using a synthetic problem setting to systematically vary the problem characteristics and system parameters and assess the impact on performance, and (iv) formally analyzing the properties of the technique. A central problem addressed by this technique is how diverse techniques for representation, reasoning and learning that arise from differing viewpoints on intelligence can be reconciled to form a consistent and effective whole. For example, how can neural network backpropagation and knowledge-based diagnosis be combined to achieve an effective structural credit assignment technique for a hybrid knowledge representation?

1 Introduction

An intelligent system must routinely deal with massive information processing complexity. Though research under the broad umbrella of artificial intelligence (AI) has made *some* progress in theorizing mechanisms that deal with portions of this complexity, many questions remain unresolved. The body of research described and proposed here is motivated by the view that various schools of thought within AI all contain valuable insights that tend to provide answers to simulating some portion of intelligence. Thus, as we wish to build progressively more complete techniques for simulating intelligence as a whole, one approach is to build hybrid systems that retain the power of the diverse techniques upon which they are based, making (explicit) tradeoffs where necessary. The research outlined here is the result of applying this perspective to classification problems.

The ubiquity and importance of classification in intelligence is well recognized (Chandrasekaran & Goel, 1988) (Duda et al., 2000) (Weiss & Kulikowski, 1991) (Winston, 1992). One of the primary reasons that classification is so important is that it allows diverse situations, objects, and so forth to be handled similarly by grouping them into equivalence classes. Without the use of equivalence classes, each new situation encountered would need to be treated separately from all those previously experienced. Thus, classification is a fundamental strategy for organization and practical use of knowledge. At some level, classification is also required for learning to be meaningful, since in practical terms no two situations are ever exactly the same. Use of classification for knowledge organization is pervasive in the sciences – for example, biological taxonomies, or the organization of organic compounds in terms of chemical families and functional groups. The engineering disciplines also facilitate the use of knowledge through classification. For instance, note the use of design patterns in computer science and architecture to prescribe action in classes of situations. These patterns could not be applied if classification was not used to identify a situation with an equivalence class associated with a particular pattern. Because of the importance of classification to intelligence, it is not surprising that it has played a significant role, explicitly or implicitly, in many knowledge-based systems (Clancey, 1985) (Porter et al., 1990) (Bylander et al., 1989), and has also been studied extensively in the machine learning community (Mitchell, 1997) (Wasserman, 2004).

However, though much work on classification has been done in the knowledge-based AI community, there has been little focus within this community on how to automatically repair the knowledge structures used for classification when knowledge engineering is faulty. Thus, though previous work has focused on knowledge representation and reasoning, learning has been conspicuously absent. This gap is significant for at least two reasons. First, systems that depend entirely upon handcrafted knowledge are brittle in the face of changing or incompletely understood environments, and may therefore require significant intervention and tuning in order to achieve satisfactory performance. Second, theories dealing with intelligence must address knowledge representation as well as the use *and acquisition* of that knowledge in order to avoid leaving apparent questions unanswered. This suggests the need for development of classification methods that not only make use of knowledge structures, but also have the capability to modify those structures.

On the other hand, ML approaches to classification take a data-centric view, relying on very general and thus very flexible mechanisms to inductively infer classification knowledge from large datasets. While these methods make only minimal assumptions about the data, they also tend to have resource requirements (e.g. time, dataset size) that are not always realistic given the constraints under which intelligent systems must operate, or practically speaking, within the constraints of a problem that we wish a system to solve. Research on tree-structured bias (Russell, 1988), also called structured determinations (Tadepalli & Russell, 1998), addresses this problem by introducing knowledge structures in the form of tree structures into ML classification techniques. This research has demonstrated a significant positive impact on learning efficiency. These knowledge structures subdivide large classification problems into a series of smaller problems. The work described here also makes use of tree-structured knowledge to bias learners. However, our work addresses a more general class of problems than does past work on tree-structured bias. We handle problem domains with multivalue features, in contrast to past work on tree-structured bias, which handled only binary features. In addition, this work takes a broader view of the kinds of interactions that a situated classification learner may have with its environment. Interactions anticipated by past work on tree-structured bias form a subset of the interactions considered here. Finally, this work explicitly represents abstraction from directly observable characteristics of the environment to the features represented by nodes in the knowledge structure, making it possible for the learner to modify these abstractions. This concept does not appear within past work on tree-structured bias. A more complete comparison with previous work on learning with tree-structured bias is presented in the related research section.

The goal of this research is to produce a family of hybrid systems that incorporate some of the flexibility of learning that comes from ML approaches to classification, as well as some of the scalability and efficiency that comes from the representation and reasoning of knowledge-based AI approaches. The result will be a system that makes a unified statement about representation, reasoning and learning. In the following section, the specific kind of classification problem that the technique proposed here addresses is described in more detail.

2 Learning Task

2.1 FreeCiv

In order to describe the general problem and classification technique clearly, we will first introduce a concrete example from the turn-based strategy game FreeCiv (www.freeciv.org). FreeCiv is an open-source variant of a class of Civilization games with similar properties. The aim in these games is to build an empire in a competitive environment. The major tasks in this endeavor are exploration of the randomly initialized game environment, resource allocation and development, and warfare. Winning the game is achieved most directly by destroying the civilizations of all opponents, though there are alternative paths to victory. We chose FreeCiv as a domain for research because playing the game provides many opportunities for the use of classification, and is a large enough problem to require efficient classification techniques in order to make decision making tractable.

In this example, the focus is on the specific task of constructing a city in FreeCiv. The game is played on a board that is divided into a grid. Each square in this grid can be characterized by the type of terrain, presence of any special resources, and proximity to a source of water such as a river. One of the fundamental actions taken while playing FreeCiv is the construction of cities on this game map, an action that requires resources. In return for this expenditure, each city produces resources on subsequent turns that can then be used by the player for other purposes, including but not limited to the production of more cities. The quantity of resources produced by a city on each turn is based on various factors, including the terrain and special resources surrounding the city's location on the map, and the skill with which the city's operations are managed. As city placement decisions are pivotal to success in the game, a player must make reasoned choices about where to construct cities, and a major factor in this decision is the level of resource production over time that can be expected from the city to be built. Thus, an important classification task that is a subproblem in the game of FreeCiv is the categorization of a given map location in terms of the expected future resource production of a potential city built in that location.

Figure 1 displays the structure of a hierarchical classifier designed for the classification of a potential city location on the game map in terms of expected future resource production. This representation of classification knowledge fits a pattern of classification common in knowledge-based AI, termed ”structured matching”, that is shown to require time and space linear in the number of input parameters for inference (Bylander et al., 1991). This computational efficiency makes structured matching a powerful classification technique, leading to our choice of hierarchical classification as the knowledge-based representation underpinning the proposed family of hybrid classification techniques. As described in detail later, we propose pairing this symbolic-level representation with numerical-level representations by placing statistical learners within each of the nodes in the hierarchy. We call the resulting hybrid representation an ”Abstraction Network” (AN). Given this representational choice, we are then left with the problem of specifying coherent reasoning and learning procedures that operate over our hybrid representation. These procedures are described in detail after a further discussion of the specific problem addressed by this work.

To return to our running example, there is one important characteristic of the city production estimate task that need not be true of classification tasks in general. Note that the city production estimate classification is *predictive*. That is, the class label that is produced is interpreted as a prediction about some future occurrence. It can be argued that all useful classifications have this property; if a classification cannot be used to draw *some* inferences, what is its purpose? The characteristic that sets this problem apart from classification in general is that here the intermediate abstractions in the classification hierarchy are also predictive classifications. That is, there is a way of expressing knowledge used in the city production estimate task as a composition of ”smaller” classification problems, each of which is also predictive. For this reason, we call the task an instance of a ”compositional classification” problem. The predictivity of the intermediate classification tasks is important because this characteristic enables diagnosis at the symbolic, knowledge-based level of the representation, which we will

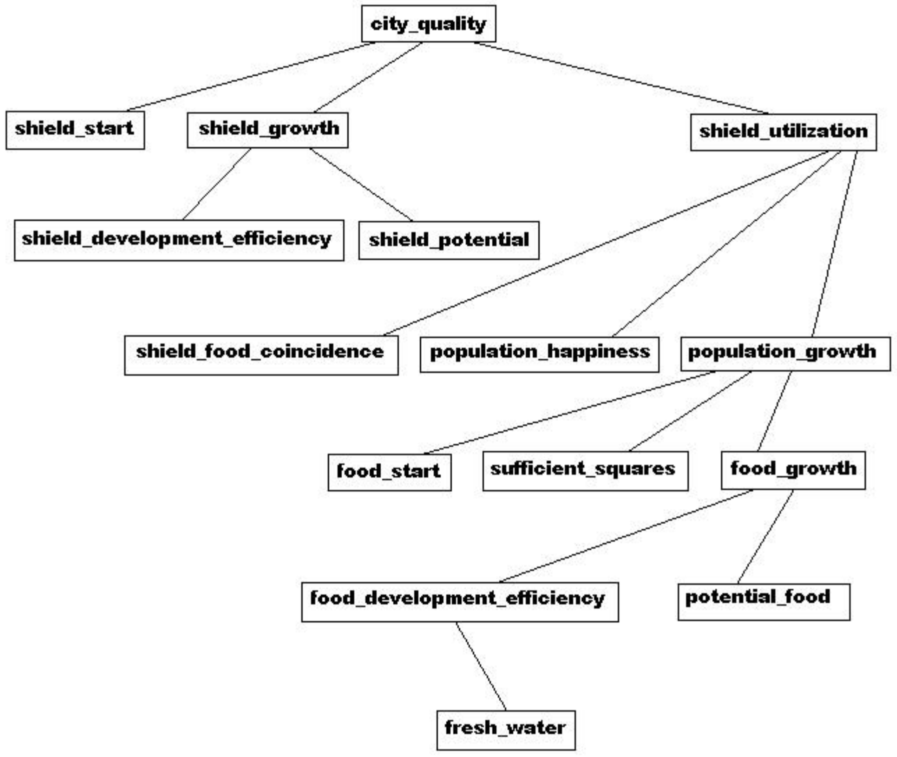


Figure 1: City Production Estimate Classifier

pair with techniques for structural credit assignment at the numerical level of the representation in order to form a complete technique for diagnosis and repair (learning). In order to make appropriate modifications to the structure, the elements of that structure that are most directly responsible for error must be identified. Accurate structural credit assignment allows faulty knowledge to be repaired while leaving correct knowledge untouched. Because the problems we consider can be broken into sub-problems that all have a predictive nature, the results of inference can be verified once time passes and actual relevant occurrences can be observed. This then affords an opportunity for learning, since errors in classification can be automatically detected and diagnosed. Effectively, the compositional nature of the classification tasks we consider allows the environment to reasonably provide oracles for each of the nodes in the hierarchy.

2.2 Compositional Classification

Let us now step back and more formally describe the class of learning problems, *compositional classification* tasks, that we intend to address with the research. In the case of the FreeCiv example outlined above, we wish to predict whether a potential city location will yield either average, above average, or below average future resource production if the city is built. We formally define compositional classification as follows.

Let T be a discrete random variable representing the class label. Let $S = \{s : s$ is empirically determinable and $h[T] > h[T|s]\}$, where $h[x]$ denotes the entropy of x. S is a set of discrete random variables that have nonzero mutual information with the class label and are "empirically determinable." Each member s of S represents a related set of equivalence classes, where each value taken by s is a unique equivalence class. In the case of FreeCiv, things like the future population growth of the potential city and the amount of food provided by terrain squares around the city location constitute S. A task instance is generated by jointly sampling the variables in $S \cup T$. In FreeCiv, the game engine handles this for us by randomly generating a game map and handling game dynamics that govern the relationships among the variables in S. Empirical determinability captures the notion of predictivity, indicating that each equivalence class represents some falsifiable statement about the world. In the simplest case, empirical determinability means that the value taken by the variable in a given task instance is directly observable. In general, some experiment (a branching sequence of actions and observations) may need to be performed in order to observe the value of some $s \in S$. The simple case can be seen as a trivial experiment consisting of zero actions and a single observation. In FreeCiv, all of the values can be directly observed, though some can be observed only after classification has occurred.

Each experiment has some nonnegative cost. We denote by $C_b(s)$ the cost of the experiment required to determine s before predicting the class label. The task is constrained by limited resources; only a fixed cost R_b may be incurred before the decision about the class label must be produced[1]. For this reason, the values of only a proper subset of S will in general be known when the prediction must be produced. Let $K \subseteq S$ with $\sum_{k \in K} C_b(k) \leq R_b$ be the information available at the time that classification must be performed. In the FreeCiv task, the resource constraint is time. In order to be useful, the prediction of city resource production must be made before the city is actually constructed and its resource production rate can be observed. Thus, we cannot directly observe the proper prediction value at inference time, but can obtain the true value later in order to learn.

Learning is required in part because the distributions $\mathcal{P}(s|K), s \in S \cup T, K \subseteq S$ are not assumed to be given, but must be inferred from experience. In this way, we are able to relax the requirements on the knowledge engineer; if knowledge about the distributions is available a priori, it is possible to initialize the classification knowledge accordingly and decrease the demands on learning. But, when this knowledge is not available, not complete, or not correct, we require the system to learn the correct values.

After the predictive class label is produced and some time passes, the correct class label is determined and some additional quantity of resources R_a is allotted to the learner. These resources are then used to determine the values of other variables empirically before the next task instance is presented. The costs of performing experiments before predicting the class label may not be the same as the costs of performing ex-

[1]Actually, these costs and resources are better represented as vectors, as there may be multiple dimensions of cost, where each dimension has its own constraint.

periments afterwards. For this reason, we denote by $C_a(s)$ the cost of performing experiment s after class label prediction. For some domains we may have $C_a = C_b$, but this need not be true in general. In the case of FreeCiv, there is a proper subset of S that is always available before classification and the remainder of S is never available until after classification. This is a special case of the general domain, where $R_b = 0, R_a = \sum_{s \in S \cup T} C_a(s)$ and there is some (proper) subset of S s.t. $C_b(x) = 0$ for all x in the subset. This characteristic is important because it makes the value of information problem trivial. Initially, we focus exclusively on problems with this characteristic in order to avoid the need to incorporate strategies for determining information value at this time. Because this situation arises commonly in predictive classification problems, where the constraining costs are temporal and either 0 or ∞ before classification[2], this subset of problems is far from empty and is interesting in its own right. However, it is worth identifying the more general class of problems with arbitrary cost values on experiments because it is likely that techniques proposed for the problem subclass considered here can be extended to cover the general case by drawing from work on information value theory (Howard, 1966) (Jensen & Liang, 1994) and budgeted learning (Madani et al., 2004). Information cost is also taken into account in active learning (Tong & Koller, 2002), though the setting in active learning is different from that considered here.

Success at this learning task can be measured in terms of the final classification accuracy achieved, the rate at which accuracy improves as examples are presented, and the resources saved. Alternatively, resources saved during one instance could be made available for use during subsequent instances. In this case, the resources remaining at the end of the sequence contribute to the success measure. For the subclass of problems we consider here, we consider resource conservation to be a less important metric.

This research is specifically concerned with two simultaneous problems. First is the problem of enhancing the efficiency in terms of time and/or number of required examples of existing numerical techniques used for compositional classification problems. Second is the problem of making efficient knowledge-based techniques for compositional classification more flexible by adding the capacity for automatic diagnosis and repair.

3 Existing Work

The issues and hypotheses that form the core of this document arise from an ongoing research program (Jones & Goel, 2004) (Jones & Goel, 2005), (Jones & Goel, 2007), which has led to the creation of a framework codifying the idea of Abstraction Networks. This framework is used to build structured knowledge representations for the compositional classification task described above, and has to date been integrated with two numerical learning techniques, rote table-based learning and artificial neural networks (ANNs). This framework has undergone preliminary evaluation in three

[2]Of course, there is also a class of problems where the constraints are temporal and take arbitrary real values. For instance, this occurs when the experiments that must be performed to determine values before classification are non-trivial, but there is some fixed time horizon in which a decision must be made. Such problems once again require judgements about value of information, and thus are not examined here.

problem domains – the FreeCiv problem discussed above, as well as an economic forecasting problem and a synthetic domain. Current research is focused on extending and modifying the AN framework as understanding of the issues is increased through experimentation, and as the framework is applied to larger problems. The framework is based on a few observations and assumptions :

Observations:

- In general, evaluation and learning are interleaved, i.e. on-line learning must be possible. This means that components cannot be trained separately, but must rather be trained as a complete classification system.
- Saving resources is desirable, so resources should be expended only when the acquired information will be useful, either contributing to accurate class label prediction or allowing meaningful learning to occur. In the subset of the general compositional classification task considered in this paper, only the latter consideration is significant, as a fixed set of variables will be (freely) available for use in classification.

Assumptions:

- Although the specific dependencies among the relevant variables are not given, we have to this point assumed that the *structure* of the dependencies among the variables is correctly given. For example, in our FreeCiv network (see Figure 1), population growth is directly dependent upon food growth, but shield utilization is dependent upon food growth only insofar as it effects population growth (i.e. is independent of food growth given population growth). We assume that correct background knowledge about these relationships is available. We have considered extensions to the framework to relax this assumption, but the work described here is based on the availability of correct dependency structure knowledge.
- We further assume that the structure of the dependencies is hierarchical, and that for all leaf variables l in this structure, we have $C_b(l) = 0$, while for all other variables v we have $C_b(l) \neq 0$. This assumption simplifies the learning method.

3.1 Abstraction Networks

3.1.1 Representation

Informally, we begin by establishing a node for each $s \in S \cup T$. These nodes are connected according to the given dependency structure, which we know will result in a hierarchy based on the given assumptions. This structuring follows the pattern of structured matching. A structure used for experimentation in the previously discussed FreeCiv problem is depicted in Figure 1. Each node will handle the subproblem of learning to predict the value of the variable with which it is associated given the values of its children, which are the variables upon which the variable to be predicted has direct (forward) dependency. Organizing the structure of the knowledge to be learned in this fashion has the benefit of making full use of the dependency structure knowledge to limit the hypothesis space while being certain not to eliminate any hypothesis that could be correct, and also yields the proven efficiency benefits of hierarchical classification (Bylander et al., 1991).

A more formal definition follows.

Definition 3.1 *Here, we will define a* supervised classification learner *as a tuple $< I, O, F, U >$, where I is a set of input strings (input space), O is a set of output symbols (output space), F is a function from I to O, and U is a function from $(i, o) : i \in I, o \in O$ to the set of supervised classification learners that share the same input space I and output space O.*

Definition 3.2 *An* empirical verification procedure *is a tuple $< E, O, C_b, C_a >$ where O is a set of output symbols (output space) and E is an arbitrary, possibly branching sequence of actions in the environment and observations from the environment concluding with the selection of an $o \in O$. C_b and C_a are the costs of procedure E before and after classification, respectively.*

We can now be more specific about what makes a set of equivalence classes empirically determinable, a term used more informally above. Any output space O of an empirical verification procedure is an empirically determinable set of equivalence classes. So, viewed from the other direction, a set of equivalence classes is empirically determinable if an empirical verification procedure can be defined with an output space equal to that set of classes. Note that this definition is in terms of the actions and observations available to the agent that learns and reasons with the knowledge, making a commitment about the way that interaction with the environment is expected to justify and give meaning to knowledge in this system.

Definition 3.3 *An* Abstraction Network *is a tuple $< N, O, L, P >$, where N is a (possibly empty) set of Abstraction Networks, O is a set of output symbols, L is a supervised classification learner, and P is an empirical verification procedure. Let I be the set of strings formable by imposing a fixed order on the members of N and choosing exactly one output symbol from each $n \in N$ according to this order. The supervised classification learner L has input space I and output space O, and the empirical verification procedure P has output space O.*

Notice that when N is empty, L is trivial and has no use as the input space is empty. In these cases (the leaves of the AN), the only way to make a value determination is to invoke P. Because I am restricting the current work to cases where AN leaves are always determined empirically before classification, this is not an issue[3]. Having described the AN representation, we next turn to reasoning (performing predictive classification) using an AN.

3.1.2 Reasoning

In a given task instance, the values of the leaf nodes are fixed by observation. As described above, in the problem settings considered here, obtaining the values of the

[3]That is, in the current work, whenever $N = \emptyset, F.C_b = 0$. If the technique is generalized, provisions will have to be made to deal with undetermined leaf values.

leaf nodes has zero cost, and no other values are available before classification. Each node with fixed inputs then produces its prediction. This is repeated until the value of the class label is predicted by the root of the hierarchy. This procedure will produce the most likely class label based on the current state of knowledge.

More formally, the reasoning procedure over an arbitrary AN a can be described as follows:

```
begin AN-reasoning($a$)
  1. If $a.N = \emptyset$, execute $a.P$ and return the result.
  2. Else, recursively execute this procedure for each $n \in N$ to generate an input
     string $i$ for $a.L$, then return $a.L.F(i)$ and store this value and $i$ for the purpose
     of the learning procedure (called $a.last\_value$ and $a.last\_input$ below).
end
```

3.1.3 Learning

After classification, the true value of the class label is obtained. If it is correct, no further action is taken. This preserves the maximum quantity of resources, as the current knowledge has produced the correct overall result in this instance. On the other hand, if the value is found to be incorrect, the following credit assignment procedure is followed, setting the root node as the ”current node”:

1. The true value of each child of the current node is obtained and compared to the relevant predictions.
2. If the predictions of all children were correct, modify local knowledge at the current node.
3. Otherwise, recursively repeat this procedure for each child node that was found to have produced an incorrect prediction.

More formally, for an AN a (note that $last\_value$ and $last\_input$ are explained under ”Reasoning” above):

```
begin AN-learning($a$)
  1. If $a.P == a.last\_value$ then return $true$.
  2. $\forall n \in a.N$, call AN-learning($n$). If $\exists n \in a.N$ s.t. AN-learning($n$)== $false$
     then return $false$.
  3. $a.L <- a.L.U((a.last\_input, a.P))$, return $false$.
end
```

Notice that this procedure has a base case when the leaves are reached, as their true values were obtained before classification, and thus cannot be found to be incorrect during learning.

Depending upon the relative weighting of resource preservation and learning speed in the evaluation metric, this procedure may be suboptimal, because it implements the policy of only obtaining information when it is certain that the information will lead to learning (with the exception of the class label, which is always determined after prediction). It is likely to be a good policy when resource preservation is weighted highly against learning speed, and a poor policy when the reverse is true. Some tuning of the balance between obtaining the correct values of more nodes and preserving resources could be a useful generalization of the technique. However, note that this choice of policy will not prevent convergence. If we are to consider some piece of knowledge stored within the hierarchy as incorrect, there must be at least one situation occurring with nonzero probability where that knowledge will lead to an incorrect overall result (we take this as the definition of incorrect knowledge). When this situation arises, the incorrect knowledge will be identified by credit assignment, and the knowledge will be modified.

Another point to notice here is that the specific procedure for the modification of local knowledge is not specified. Any supervised classification leaner that satisfies the definition given in the ”Representation” section is acceptable. A closely related point is that the representation of the knowledge, and thus the procedure for knowledge application within each node is similarly unspecified. This is quite intentional: *any* knowledge representation/inference/learning technique can be used within each node. Heterogenous sets of techniques could be used within a single hierarchy. The specific technique or set of techniques that will perform best on a given problem depends on the specifics of the subproblems – choosing learning techniques that exploit known characteristics of each subproblem will, of course, lead to the best overall results. For instance, for some kinds of problems it may be that Bayesian learning of probabilities is the most effective technique at all nodes. In this case, the overall learner is somewhat similar to a particular type of Bayes net, augmented with a learning procedure that is sensitive to knowledge acquisition costs. See the section on related research for a more thorough comparison with Bayes nets. In other cases, it may make sense to use ANNs within some or all nodes, in order to introduce a different kind of inductive bias (based on smooth interpolation) for some subproblems. Generally, the point is that because the characteristics of the dependencies between members of $S \cup T$ are not fixed over the entire domain of interest, it does not make sense to fix a learning method for the subproblems in the absence of knowledge, nor is it necessary to do so in order to specify a solution exploiting domain characteristics that *are* given. Of course, when instantiating this technique for a specific domain, these choices must be made. These details are explicated for the FreeCiv city resource production prediction problem in the next section.

In addition to the content learning procedure described above, where the knowledge contained at nodes within the network is modified, we have also done some work on automatically tuning the equivalence classes at nodes within the hierarchy. This translates to automatically adjusting O, the set of output values, at nodes within the hierarchy, essentially adjusting the concepts represented by the nodes within the hierarchy. Changing the set of output values at a node also requires adjustments to the node's EVP, such that newly added output values will be learned to apply to some set of situations, or deleted output values will be learned to be obsolete. Finding the right level of information loss at nodes in the hierarchy is important because too much loss (i.e. too few output values) will provide insufficient information to the parent node and cause learning failure, while too little information loss decreases the generalization power of the network for learning. At the limit, if there is no information loss at any node in the network, the representation becomes equivalent to a flat representation. Because too much information loss results in learning failures, identifying these failures will help to identify locations at which information loss should be decreased by increasing the available output values. Given this capability, we can start with very high information loss at each node and allow loss to be decreased as breakdowns of the learning process are identified. Our experiments used such a procedure, which is tied to the specifics of the learners used within the network nodes. This work is described in more detail later in this document.

4 Formal Justification of SCA Technique

In this section, we sketch a proof that the structural credit assignment technique used for AN learning is optimal with respect to maximizing expected decrease in *diagnostic* search space entropy with each probe, under assumptions outlined below. Here, the search space consists of all possible error conditions that lead to an error observed at the root of the AN, under the following provisions:

- A value produced at a node is wrong only if it is both *objectively* wrong (wrong according to the associated EVP) and *subjectively* wrong (recursively, leads to the production of an erroneous value at the parent). In the language of diagnosis, this amounts to ignoring compensating faults. This view of error arises from our functional stance towards knowledge; since knowledge in an AN exists to serve a purpose, it is incorrect only if it fails to serve that purpose.

- Without loss of generality, we assume that each node produces values in a way that is actually dependent on the values of child nodes.

From these assumptions, we are left with a diagnostic search space for a given failure that consists of all possible contiguous sets of nodes that include the root. A given diagnostic solution is correct for a given failure situation if it is the maximal such set for which all nodes in the set produced incorrect values during the related inference. The set represents the nodes that produced erroneous values during the failure, and learning within nodes will occur at the fringe of the set, as per the credit assignment and learning procedure described earlier. Note that this diagnostic search space is distinct from the hypothesis space searched by the SCA/learning procedure applied to a sequence of examples; *this* search space is not concerned with any specific knowledge stored at the nodes, but only with explaining the fault location(s) that lead to a particular failure.

For any given node in the AN x, define X to be the entropy left in the diagnostic search space if x is probed (associated EVP executed) and found to have produced a good value, and $\neg X$ the entropy left if x is found to have produced a bad value. Then, the expected remaining entropy if x is probed is given by $X \cdot P(X) + \neg X \cdot P(\neg X)$, where $P(X)$ and $P(\neg X)$ represent the a priori probabilities that x will be found to be either good or bad, respectively, if probed.

Lemma 4.1 *For any pair of unprobed nodes a, b in an AN such that a is a (possibly indirect) ancestor of b:*

1. $B \geq A$
2. $\neg B \geq A$
3. $\neg A \leq B + \neg B - 2 \cdot A$

To see that (1) & (2) are correct, notice that if a is probed and found to have produced a correct value, all hypotheses consistent with this observation are consistent with *either* result of probing b; that is, the set of consistent hypotheses remaining if b is probed and found to have produced a correct value is a subset of the hypotheses consistent with a having been found to be correct, and likewise for b being found incorrect. The intuition behind this proof lies here. Since we do not know whether node b's status is important until we probe node a, it is more fruitful to probe at a first. (3) is a related inequality, and is based on the observation that if b is a direct descendant of a, the hypotheses consistent with b being shown correct are those hypotheses consistent with a being shown correct, plus some fraction of those consistent with a being shown incorrect. Likewise for b being shown incorrect. Since there is no solution consistent with neither b being shown correct nor b being shown incorrect, and there is no solution consistent with a being shown incorrect, b being shown correct and b being shown incorrect, we arrive at a variant of (3) with an equality rather than an inequality. Now notice that if b is an indirect descendant of a, B and $\neg B$ will not only be duplicating coverage of A, but will also be duplicating coverage of some portions of $\neg A$ that are consistent with correct values having been produced by some nodes on the path from b to a. This leaves us with (3) as presented above, since the RHS may exceed the LHS due to this duplication.

Theorem 4.1 *For any two unprobed nodes a and b within an AN, where a is a (potentially indirect) ancestor of b, the expected remaining entropy in the diagnostic search space is less at a, making a a more desirable probe point.*

To briefly sketch the proof, let us assume that b constitutes a better next probe point than a, on the basis of expected remaining entropy in the diagnostic search space. Then $B \cdot P(B) + \neg B \cdot P(\neg B) < A \cdot P(A) + \neg A \cdot P(\neg A)$. Substituting using our lemma and simplifying yields $P(A) > 1$, a contradiction.

Moving from this result to our SCA procedure is straightforward; it is preferable to probe nodes with ancestors that have been probed (and found incorrect, of course). Further, the order of probing among such nodes is arbitrary since the probing of any node with this characteristic cannot remove the characteristic from any other node, and no such nodes can remain when a diagnosis is uniquely selected.

5 Synthetic Experiments

In order to verify that the SCA technique described above does allow for correction of faulty knowledge engineered content in an AH, and show generality with respect to the learners used within AH nodes, we have performed a set of experiments in a synthetic domain. The environment in this domain consists of a fixed abstraction hierarchy, over which no learning will occur, that represents the correct, target content (and structure) for the problem. Given this fixed AH, we then create a separate ”learner” AH that will be initialized with incorrect knowledge content and expected to learn to functionally match[4] the content of the target AH. This is implemented by initializing the knowledge content of both the fixed and learner AH nodes separately with pseudo-random values. The randomly generated content of the fixed AH forms the target knowledge for the learner AH. Because the work described here is concerned only with repairing content and not structure, we do build the learner AH with a correct structure that matches that of the fixed AH. Training proceeds by (1) generating a pseudo-random sequence of floating point numbers to serve as the observations for the input nodes of the AHs, (2) performing inference with the fixed AH, saving the values produced by all intermediate nodes as well as the root node, (3) performing inference with the learner AH and (4) performing SCA and learning over the learner AH according to the procedures described above. EVPs within the inputs of both AHs are set up to quantize the floating point observations. EVPs are not needed at non-leaf nodes in the fixed AH, since no learning will occur. EVPs at non-leaf nodes in the learning AH are set up to examine the saved output value from the corresponding node in the fixed AH. In these experiments we use simple table update learners within each node.

Definition 5.1 *A* table update learner *is a tuple $< I, T, O, F, U >$, solving a classification problem that requires mapping a finite input space I onto a finite set of contiguous integers O. T is a finite set of contiguous integers that is symmetric about zero, and we call $(|T| - 1)/2$ the ”learning threshold” of the table update learner. F is a function from I to O, implemented as a composition of two functions, F_1 and F_2. F_1 maps from I to $O \times T$ and F_2 maps from $O \times T$ to O. F_2 is defined such that $\forall t \in T, o \in O, (o, t) \rightarrow o$.*

Given an input example (i, o'), U returns a new table update learner $< I, T, O, F', U' >$ where F' is a composition $F_2 \circ F'_1$. $\forall x \neq i, F'_1(x) = F_1(x)$. Let $F_1(i) = (o, t)$. Then, if $t + (o' - o) \in T$, $F'_1(i) = (o, t + (o' - o))$. Otherwise, if $t + (o' - o) < 0$, $F'_1(i) = (o - 1, 0)$ or if $t + (o' - o) > 0$, $F'_1(i) = (o + 1, 0)$. U' is an update to U to embed knowledge of the new function F' such that the next update can proceed by the same logic.

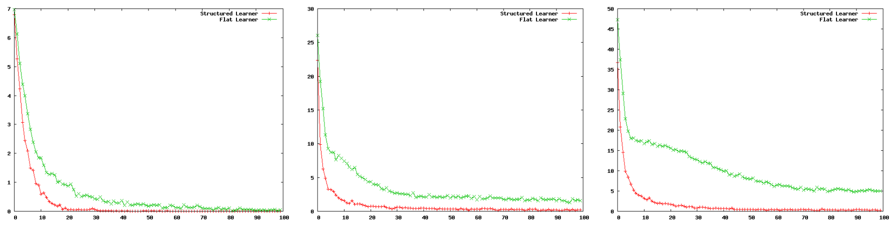
Informally, the table update learner requires indication of a significant error (an (i, o') where o' is quite different from $F(i)$) or demands some consistency in feedback before making a change to the classification of a given input. The intent of this scheme is to avoid making changes due to noise in input examples. In the synthetic domain,

[4]By ”functional matching”, we mean that we will only measure error based on whether the learner AH is able to produce output values that match those of the fixed AH – we will not actually inspect the contents of individual nodes.

noise is not an issue, but becomes significant in the additional domains reported on below. Table update learners as described here are only sensible if there is some natural ordering of class labels; otherwise, taking the difference of class labels is not meaningful. However, this limitation as well as the overall simplicity of table update learners is not a necessary aspect of AHs in general. We have used these very simple learners in our initial experimentation with AHs in order to demonstrate the power of the framework itself. In this work, all table update learners in each experimental setting used a threshold of 5, except for the comparison with Baysian networks discussed below. In the experiments in the synthetic domain, all of the structured AHs take the form of binary trees (each non-leaf node has a fanin of two). Every node, including the leaves and the root, chooses from among 3 possible output values in this set of experiments. Thus, each table update learner used in structured learners in the synthetic domain has $3^2 = 9$ entries, while the flat learner has 3^{inputs} entries. We experimented with three problem sizes. The largest had 16 inputs, with the binary structure yielding 8, 4 and 2 nodes at each subsequent layer. The other two domains used 8 and 4 inputs, respectively.

In addition to verifying that the SCA procedure described in this paper allows for correction of faulty AH content, we wished to determine the benefit of using a structured knowledge representation matching domain structure vs. using a "flat", unstructured representation. Thus, in addition to the learner AH described above, we also trained a flat learner in each problem setting for which we report results in the synthetic domain. These flat learners are implemented as AHs where the input layer is connected directly to the output node. Thus, in the flat learners used in these experiments, there is a single table update learner that must learn the full mapping from inputs to output values without the generalization enabled by a truly structured representation. Results in each tested configuration are reported for both a structured AH learner and a flat learner.

Since we train and evaluate the learners in an on-line, incremental fashion, we cannot apply the standard training set/test set approach to evaluation. Rather, we evaluate the learners' performance improvement during training by segmenting the sequence of examples into multi-example blocks, and comparing overall error rate between blocks. An error is counted whenever the learner's output on a given example does not match the output produced by the fixed AH. In this way, we are able to compare error rate around the beginning of a training sequence with the error rate around the end of that sequence.



(a) Layer sizes 4, 2, 1 (b) Layer sizes 8, 4, 2, 1 (c) Layer sizes 16, 8, 4, 2, 1

Figure 2: Per-block error rates of structured vs. flat learners for various problem sizes

The results of these experiments for the three synthetic domain sizes are depicted in figure 2 in terms of per-block error rate. The results shown are an average of 100 independent runs in each setting, with separate random table initialization at the beginning of each run. Each run in the large problem setting consists of 10,000 generated examples, which we segment into 100 blocks of 100 games each for the purposes of visualization. In the medium-sized problem setting, 100 blocks of 50 games were used in each run. Finally, in the small problem setting, each run consisted of 100 blocks of 10 games each. These results demonstrate the efficacy of the proposed method for SCA in repairing faulty knowledge engineered AH content, as well as the significant advantage of structured knowledge that reflects domain structure vs. flat representations in terms of learning speed. Of course, as problem size increases, the benefit of knowledge structure becomes more apparent, as can be seen in these results. This benefit is due to the restriction bias that knowledge structure imposes on the hypothesis space available to the learner. One remaining question is whether there are interesting realistic domains for which AHs are useful and can be designed in such a way that EVPs are definable. We have already mentioned two such domains, one from the game FreeCiv, and another from economics. In the following sections we describe results that we have obtained in these domains.

6 Application to FreeCiv

In this section, the use of Abstraction Networks for the city resource production prediction problem described above is given in detail. Results of this experimentation are also given.

6.1 AN Representation

For the FreeCiv city resource production prediction task, we use the structured matcher depicted in figure 1, producing predictive classifications of map locations in a sequence of games. Within each node we use a simple table update learner, defined as described in the previous section.

6.2 Procedure

We have experimentally compared an AN-based learner using the network depicted in figure 1 to a flat table-based learner. The goals of this experiment were to (1) determine the effectiveness of the AN learning method in increasing robustness in the face of faulty knowledge engineering and (2) to determine the effect of hierarchicalization on learning speed. The effect of hierarchicalization on inference complexity is already well understood and is known to make inference significantly more manageable (Bylander et al., 1989). The flat learner consists of a single table update learner (table update learners are described in detail in the previous section) with an input formed from the outputs at all leaf nodes in the AN from figure 1 and yielding the same output set as this AN. This output set contains three values, corresponding to poor, moderate and good resource production. These values indicate predictions about the resource production expected from a city built on a considered map location. Specifically, the values correspond to an expected degree and direction of deviation from a logarithmic baseline resource production function that was manually tuned to reflect roughly average city resource production. Each of the intermediate nodes in the AN has an output set consisting of 5 values in this experiment. The empirical verification procedures simply check values in the game, such as population growth of a city, and discretize the value into one of the 5 available output categories. The discretization functions were manually tuned in this experiment. The content of all involved table learners (those constituting the AN and the single one used for the flat learner) was initialized to zeros, which was known to be incorrect in some cases for each of the learners. All table learners used a learning threshold of 5. Because we expect resource production from cities built on various kinds of map locations to potentially differ qualitatively as games progress, we trained 3 AN-based learners and 3 flat table learners, with one of each learning to make predictions about resource production in the early, middle or late stages of the game. Results reported are cumulative across all three learners of the appropriate type.

Since we train and evaluate the learners in an on-line, incremental fashion, we cannot apply the standard training set/test set approach to learning and evaluation. Rather, we evaluate prediction improvement during training by segmenting the sequence of examples into multi-example blocks, and comparing overall error rate between blocks. In this way, we are able to compare error rate around the beginning of a training sequence with the error rate around the end of that sequence.

Each turn of each game played is treated as a separate example. This means that an error is potentially counted on each turn of each game by producing a prediction[5] based on the current state of knowledge, finishing the turn, perceiving the outcome of the turn, and then determining whether the value produced correctly reflects the resource production actually experienced on that turn. If the value is incorrect, an error is counted. Note that this error counting procedure contrasts with another possibility; producing a value only at the beginning of each game, and counting error on each turn of the game based on this value, while continuing to learn on each turn. While this alternative more closely matches the intended *use* of the learned knowledge, we chose to instead allow a value to be produced on each turn in order to reflect the evolving state of knowledge as closely as possible in the error count. A negative consequence of this choice is that some overfitting within games may be reflected in the error count. However, a decrease in error rate between the first and last block in a sequence can be seen as evidence of true learning (vs. overfitting), since any advantage due to overfitting will be as pronounced in the first block of games as in the last.

In each trial, a sequence of games is run, and learning and evaluation occurs on-line as described above. The AN-based learner is trained on sequences of 175 games, while the flat table learner is allowed to train on sequences of 525 games. We trained the flat table learner on sequences three times longer than those provided to the AN learner to determine whether the flat table learner's performance would approach that of the AN learner over a longer training sequence. As described above, we segment these se-

[5]Though as the game progresses, additional information becomes available, predictions are always made using only information available at the beginning of the game.

	AN learner	Flat Table Learner	
	7^{th} block	7^{th} block	21^{st} block
Without city improvements	24%	(4%)	1%
With city improvements	29%	7%	10%

Table 1: Average percent decrease (or increase, shown in parentheses) in error for decomposition-based learning implementation from block 1 to 7, and for the flat table learner from block 1 to blocks 7 and 21.

quences of games into multi-game blocks for the purpose of evaluation; the block size used is 7 games. Each game played used a (potentially) different randomly generated map, with no opponents. The agent always builds a city on the first occupied square, after making an estimate of the square's quality. Building in the first randomly generated occupied square ensures that the learners will have opportunities to acquire knowledge in a variety of states. In order to compensate for variation due to randomness in starting position and game evolution, results are averaged over multiple independent trial sequences. Each result for the AN learner is an average of 60 independent trials. Each result for the flat table learner is an average over 25 independent trials; each trial is time consuming, as each trial for the flat table learner is three times as long as for the AN-learner, and it did not seem likely that further trials with the flat table learner would offer significantly more information.

To compare the speed with which learning occurs in the two agents, we ran two separate sets of trials. The first set of trials was run in an environment where no city improvements were constructed in the area surrounding the city. The second set of trials did allow for the construction of city improvements, but had an identical environment in all other ways. For each set of environmental conditions, we measure the quality of learning by comparing the average number of errors counted in the first block of the sequences to the number of errors counted in the last block. In the case of the flat table learner, we make two comparisons. The first compares error in the first block to the block containing the 175th game, illustrating decrease in error over the same sequence length provided to the AN learner. We also compare error in the first block to error in the last block of the flat table learner's sequences, to determine whether the flat table learner's improvement will approach that of the AN learner over sequences three times as long. We perform this evaluation separately for each of the two environmental setups.

6.3 Results

The results of the experiment are summarized in Table 1, and are shown in detail for each block of games in Figure 3. The AN-based learner is able to produce a greater improvement in error rate in each case, as compared to the flat table learner, both after the same number of games and after the flat table learner has played three times as

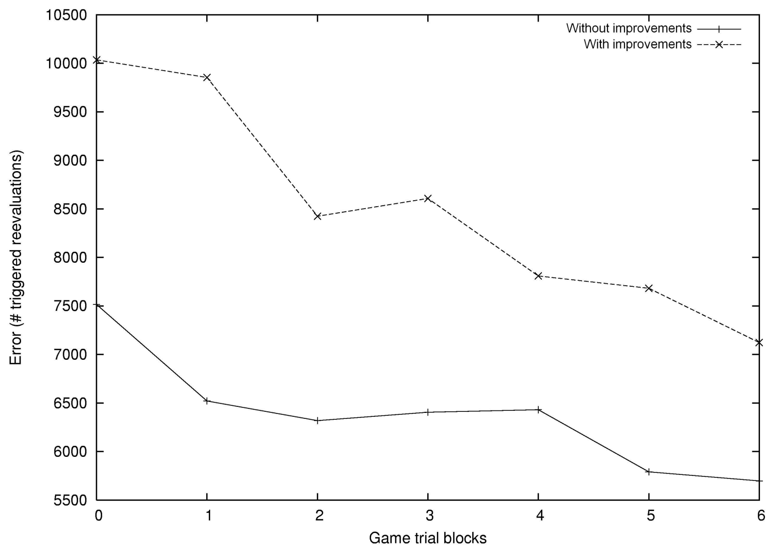


Figure 3: Average Error Rates By Block in Each Trial

many games. For the two scenarios, the average improvement in error rate is 26.5% for the AN-based learners, compared to only 1.5% after the same number of training examples for the flat learner. The decrease in error across a typical sequence was not strictly monotonic, but did exhibit progressive decrease rather than wild fluctuation. Even after three times as many games had been played by the flat table learner, the decrease in error rate is significantly less than that achieved using ANs after only seven blocks. In one case, it appears that learning has not yielded an advantage in error rate in the flat table learner even after 525 games. Examining the complete set of results for intervening blocks does mitigate this impression to some extent, as an overall downward trend is observed, with some fluctuations. However, given that, for the flat learner, the fluctuations can be of greater magnitude than the decrease in error, the learning that has been achieved after this number of games does not appear significant. Based on the significant difference in observed learning rate, these trials provide evidence that the composite structure of ANs offer an advantage in terms of allowing learning to occur more quickly in a large state space than with a flat knowledge representation. Because the AN-based learners are able to improve their performance over time, it also appears that the method for learning on hierarchical classification knowledge structures is effective in adding some robustness to certain kinds of deficiencies (content deficiencies) in engineered knowledge.

6.4 Equivalence Class Tuning in FreeCiv

The work on equivalence class tuning is implemented by monitoring F at each table update learner in the network. If oscillatory changes are detected in the function (i.e. the mapping for some $i \in I$ made by F is changed away from some $o \in O$ due to learning, and then later changed back to o by subsequent learning), we decrease the information loss at one of the child nodes. This procedure is based on the observation that learning failure manifesting as oscillatory behavior in F is due to insufficiently discriminatory information coming into the node computing F. This could be due to incorrect network structure, which we have not yet attempted to repair, or due to a child node not passing along enough information about the world state. Since we do not have a method to determine which child node should have its output space increased, our experiments used a random scheme. If the wrong child of a failing node is chosen, failure will occur again and we will have a chance to increase the output range at another node. This procedure may lead to some spurious increases in output value ranges, which will have a negative impact on generalization and learning speed. However, we performed experiments with this procedure and did achieve some moderately promising results (see below). Because all of the EVPs in the FreeCiv AN consist of quantizations of observed values, modification involves splitting one of the quantization ranges to create a new ”bin”. Specifically, we split the bin associated with the output value that indexed the learning failure location at the parent.

We tested the bin splitting procedure by initializing each node in the network to use an output set containing only one value, and allowing the information loss tuning procedure to increase the number of values in output sets when learning oscillation failures were detected. We compared the performance of this setup to a setup using knowledge engineered output ranges (as in the FreeCiv experiments reported above) with information loss tuning disabled. Each setup was run for 20 blocks of 25 games. These results are quite preliminary, but we did see learning that was both faster and resulted in fewer overall errors being committed when using automatic information tuning. The loss tuned network committed an average of 155 errors per block, while the fixed knowledge-engineered structure committed an average of 216.5 errors per block. The loss tuned network committed an average of 170 errors per block in the first 10 blocks, while the untuned engineered network committed an average of 262 errors per block over the first 10 blocks. Of course, these results may only indicate that the knowledge engineering was poor. However, at least this is some preliminary evidence that the information loss tuning procedure may be effective to some extent. Future work will expand the technique so that output sets can be both expanded and contracted, so that the tuning procedure can be used with networks that begin with engineered output sets, and will focus on more thorough evaluation.

More generally speaking, notice that the changes imposed in bin splitting are fundamentally changes to the EVP associated with a node. We affect the knowledge represented at a node by affecting the way that node is trained – the way that target values are generated by the relevant EVP. In general, one can imagine a range of different types of EVPs, beyond the ”observe and quantize” type at work in the FreeCiv city location quality problem. One can also imagine a range of different kinds of EVP adaptations that may be applicable depending upon the learning situation and the EVP type. Future

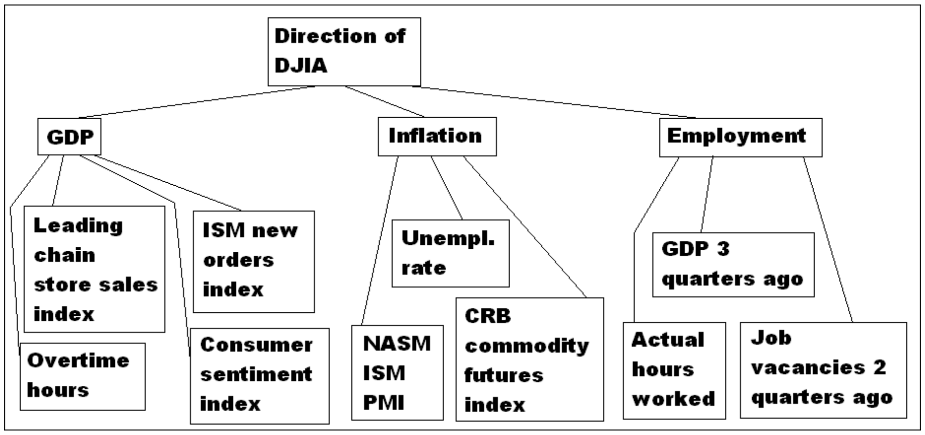


Figure 4: DJIA Abstraction Network

research is planned to investigate additional EVP types and associated adaptations.

7 Application to DJIA prediction

To demonstrate that neither the learning task nor the learning method is restricted to the FreeCiv game, we will also describe results in a different domain in the economic arena. In this domain, we are interested in classifying a current economic status as described by various economic indicators (see Figure 4) into one of two classes: the Dow Jones Industrial Average (DJIA) will rise next month or DJIA will fall next month (these class labels form T). We chose the indicators and set up the structure shown in Figure 4 based on some studies of economic indicators (eco, 2004)(Webb & Rowe, 1995)(eco, 2005). S contains the values of these selected economic indicators. Some of the values can be obtained before classification; these values come from the current or past months. However, some of these variables represent future values that cannot be observed at classification time, but must be inferred along with the class label. The same special conditions regarding experimentation cost that were described for FreeCiv also hold here. All leaves in Figure 4 can be observed before classification, while the remainder are future values at classification time, available only in retrospect.

We used data from Jan 1960 - Nov 2005, yielding a total of 497 training examples. Once again, table update learners were used within each node in the network. We manually tuned the number of output classes available to each node, based on observations of learning behavior. Again, each entry in each table update learner was initialized to zero. We observed a 23.4% decrease in error, comparing blocks consisting of the first 213 and the last 213 examples. This experiment helps to show that there is some more general applicability of AN-based learning beyond the FreeCiv problem in the context of which it was initially developed.

8 Integration with ANNs

In order to demonstrate the generality of AHs with respect to the classification learners used within nodes, we have integrated the AH framework with artificial neural network (ANN) code provided by Tom Mitchell and his students. This integration allows us to replace the table update learners used within AH nodes in most of the experiments described here with ANNs. So far, the results we have obtained with this new setup are quite preliminary. However, it does appear, as expected, that an AN-ANN system has significant advantages over an ANN-only classifier.

We used a randomly generated set of synthetic learning problems to compare the performance of AN-ANNs with unaugmented ANNs. The environment consists of a fixed abstraction network, over which no learning will occur, that represents the correct, target content (and structure) for the problem. Given this fixed AN, we then create a separate ”learner” AN that will be initialized with random knowledge content and expected to learn to functionally match the content of the target AN. We also create a randomly initialized unaugmented ANN that will be used to learn the same classification task. All ANNs, whether within the AN structure or operating in isolation, used the same backpropagation algorithm for learning[6]. Because the work described here is concerned only with repairing content and not structure, we do build the AN-ANN learner with correct structure that matches that of the fixed AN. In these experiments we first generate training and test sets. For every example that will be part of either the fixed training set or fixed test set, we generate a pseudo-random sequence of floating point numbers to serve as input values. Next, we repeat the following procedure, one repetition of which we call an ”epoch”:

1. For every example in the training set, perform inference with the fixed AN, saving the output values of all intermediate nodes and the root. Train both the AN-ANN and unaugmented ANN systems based on this inference over the fixed AN.

2. For every example in the test set, perform inference with the fixed AN, noting the value produced at the root. Perform inference with both the AN-ANN and unaugmented ANN systems, and determine whether the values produced match that produced by the fixed AN. If the value produced by a given learner does not match that of the fixed AN, count an error for that learner.

EVPs within the inputs of both ANs are set up to quantize the floating point observations, and these quantized values also form the inputs to the unaugmented ANN. EVPs are not needed at non-leaf nodes in the fixed AN, since no learning will occur. EVPs at non-leaf nodes in the learning AN are set up to examine the saved output value from the corresponding node in the fixed AN, while the output value from the root of the fixed AN is all that is needed to train the unaugmented ANN. In these experiments we use randomly initialized table update ”learners” within each node in the fixed AN, to simply provide a randomized mapping from inputs to outputs. Results obtained in a

[6]Learning rate was fixed at 0.3, momentum was fixed at 0.3, input layers contain one node per input, output layers contain one node per possible output value, and hidden layers contain a number of nodes equal to 3 times the number of nodes in the input layer.

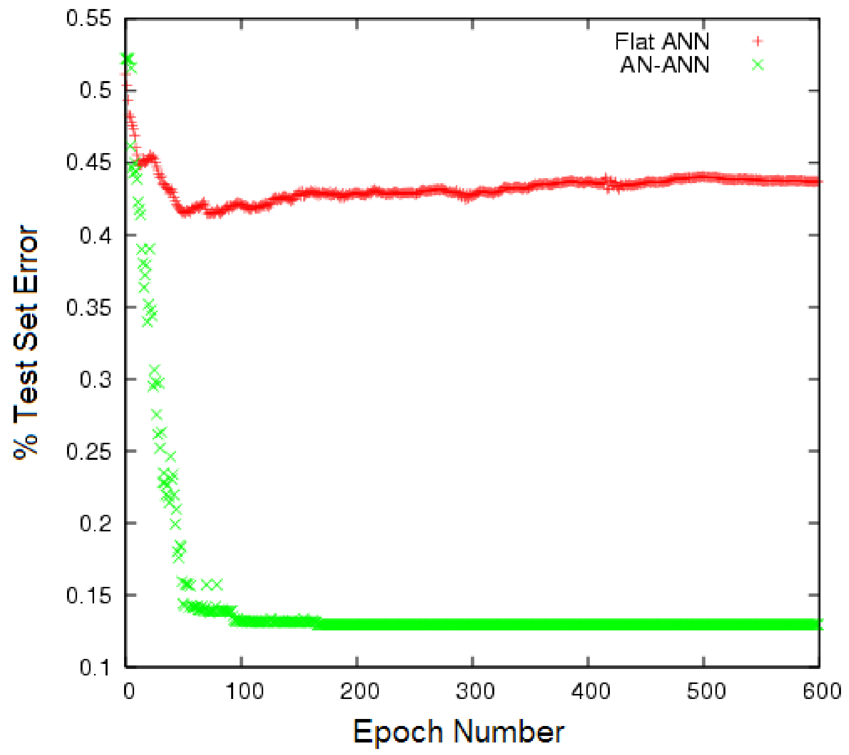


Figure 5: AN-ANN vs. Unaugmented ANN performance

representative experiment are depicted in Figure 5. In this experiment, we use ANs with a binary tree structure, with layer sizes 16-8-4-2-1. Each node is able to produce 3 values. The training set contains 1,000 examples, while the test set contains 10,000 examples. We ran the complete experiment 5 times (re-randomizing all learners and the fixed AN each time, etc), and Figure 5 depicts the average error values in each epoch across these runs.

Clearly, it appears that AN-ANNs have a distinct advantage in learning rate and in the final error achieved. A few other experiments with small parameter changes (e.g. hierarchy size) have been run, with similar results. It does appear, as expected, that the advantage of adding AN structure to an ANN-based solution to a classification problem grows as problem complexity increases.

9 Related Research

The credit assignment problem has been characterized as a core problem in learning (Minsky, 1961). Samuel (Samuel, 1957) first identified the problem in his work on checkers playing programs. Structural credit assignment is central to this work, motivating the justification of knowledge via falsifiable prediction and specifically encoding these justifications as empirical verification procedures.

Though we have begun with a knowledge-based approach to classification in this work, adopting an approach with well known benefits in terms of efficiency, there has been a significant amount of work on classification learning that does not take a knowledge-based approach. The remainder of this section is intended to position this work with respect to other techniques for classification learning.

One view of the contribution of this work is the development of a new method for scalable classification learning through use of domain knowledge. Work on hierarchical RL (Dietterich, 1998) has a similar goal, but in a different problem setting and with the use of a different kind of domain knowledge. In hierarchical RL, or partial programming (Marthi et al., 2005), procedural knowledge in the form of a hierarchy of temporally abstract actions is used, and action selection (which can of course be seen as a particular kind of classification) is the goal. This type of background knowledge is well-suited for an RL problem setting. In the scenario addressed in this work, knowledge about equivalence classes relevant to classification is more directly useful.

Work on tree-structured bias (TSB) (Russell, 1988)(Tadepalli & Russell, 1998) is the most closely related to work described here. In systems that make use of tree structured bias, a concept hierarchy like those represented by ANs is used to limit the hypothesis space that must be searched by a learner. One of the contributions of AN research is to apply the idea of tree-structured bias in new settings, including the use of ML techniques that have not been combined with tree-structured bias in the past, as well as application to non-synthetic problems. The proposed research will also move beyond past work on TSB in several other directions, studying, for example, the effects of noisy data and faulty knowledge structures on learning and expanding on theoretical results. More importantly, there are several *fundamental* differences between ANs and past work on tree-structured bias. First, TSB has dealt only with binary classifications at all nodes in the hierarchy, while ANs can deal with multivalue classifications. Next, TSB research does not have the concept of EVPs, instead relying on carefully constructed queries to the environment to learn the functions at internal nodes. This procedure can be seen as requiring a very specific kind of empirical verifiability for internal nodes – thus forcing a particular (and rather complex) form on the EVPs that a designer would write if applying TSB procedures within the AN framework. In the work described here, we take the stance that, in general, a broader set of queries to the environment may be possible. If this is the case, it will be more efficient to make use of the observations that most directly allow us to determine the value of an internal node when learning. In fact, the motivating example given by Tadepalli and Russell (Tadepalli & Russell, 1998), concerning a credit-card domain, appears clearly to have a strong kind of direct empirical verifiability at internal nodes that could be exploited by an AN using very simple EVPs. Thus, past work on TSB can be seen as a specialization of the techniques described in this paper, where only a particular kind of query is supported by the learning environment. The explicit representation of EVPs by ANs is also crucial to a major difference between AN research and past work on TSB. EVPs represent an abstraction from observable quantities to concepts used in an AN hierarchy. Because the relationship between concepts and observable quantities is explicitly represented, it becomes fair game to be operated upon during learning. It also means that we are able to adapt intermediate concepts themselves according to their functional roles – recognizing that intermediate concepts are not set in stone by the environment, but that they are constructs that exist in order to allow for correct overall classification.

Work on Probabilistic Concept Hierarchies (Fisher, 1987)(Iba & Langley, 2001) makes use of knowledge representations centered around the generation of progressively more abstract equivalence classes, as does our work. Probabilistic Concept Hierarchies however are used for unsupervised concept learning. In our work, there is a well-defined target concept (classification) that is to be learned. Thus, relevance of information and usefulness of specific equivalence classes can be defined directly in terms of the target concept. In unsupervised learning, the goal can be thought of as forming equivalence classes that explain the data. In this case, equivalence classes are valued based on how well they summarize available data. On the other hand, when learning a target concept, the usefulness of an equivalence class is based on its ability to discard information irrelevant to the target concept. Notice also that we rely on background knowledge about equivalence classes relevant to the target concept to speed up learning, while the goal of learning in Probabilistic Concept Hierarchies is specifically to learn *about* equivalence classes.

Layered learning (Whiteson et al., 2005) makes use of decomposition hierarchies to address large learning problems. In layered learning, each component's learner is trained in a tailored environment specific to the component. Our AN technique is more akin to what is called "coevolution" of components in work on layered learning, where multiple learners in the decomposition hierarchy are trained simultaneously in the actual target domain. However, in layered learning genetic algorithms are used for training. This means that the structural credit assignment problem is addressed through trial and error, which will not provide the type of scalability characteristics we expect to achieve with a systematic approach to credit assignment. An additional distinction is that ANs focus on progressive abstraction, limiting the number of inputs to each component and ensuring a learning problem of manageable dimensionality at each component. In contrast, layered learning focuses on temporal abstraction, where components responsible for selection of abstract actions are not necessarily shielded from the need to consider many raw state features. And we also allow the use of arbitrary (possibly heterogeneous) learners within each component.

Knowledge-based ANNs (Towell & Shavlik, 1994) and Explanation-based NNs (Mitchell & Thrun, 1993) both apply background knowledge in order to speed up learning in supervised classification problems. In KBANN learning, neural network structure and initialization are informed by background knowledge in the form of Horn clauses. Then, the network is trained using a standard method such as backpropagation. That is, credit assignment during learning is based on structural and numerical properties of the knowledge representation. In contrast, credit assignment over ANs is based on fixed semantic properties of the structural elements. These semantic properties are explicitly encoded as empirical verification procedures that ground the knowledge contained within a structural element in terms of falsifiable predictions about the environment. Also notice that the result of learning is different. With ANs, the structural elements of knowledge retain known, explicitly specified meanings. With KBANN, there is no guarantee that structural elements of the neural network that results from training will have any particular or identifiable meaning. So both the considered hypothesis spaces and the nature of the search through those spaces is different in KBANN vs. AN learning. Beyond the restriction bias of requiring fixed semantics for intermediate nodes, there are also other advantages such as the potential for transfer of partial networks to new problems and inspectability of knowledge.

In EBNN, a neural network is trained via the TangentProp algorithm. TangentProp works as backpropagation, however it is augmented with knowledge about the desired derivatives of the output function with respect to changes in the input values. EBNN finds the derivatives used as input to TangentProp on a per-example basis using provided background knowledge. This background knowledge is in the form of an approximate representation of the target function by a set of neural networks. The representation used for the domain theory is similar to an AN with ANNs at each node, but EBNN does not deal with learning over this representation, but rather learns while treating this information as fixed background knowledge. As in the discussion about KBANN above, notice that AN learning differs from EBNN in both representation and in the procedure for credit assignment. EBNN learning results in a trained neural network, where intermediate nodes are not guaranteed to have any identifiable interpretation. In contrast, the AN representation always maintains known, explicitly represented interpretations for all intermediate nodes. In a related point, TangentProp credit assignment distributes blame across network weights based on structural characteristics of the network, rather than based on analysis of fixed node interpretations as is the case in AN learning.

Work on Predictive State Representations (PSRs) (Littman et al., 2001) outlines rationale for using state representations that directly encode predictions about future events. In a general way, the notion that knowledge should have a predictive interpretation is central to this work as well. The requirement for empirical determinability is a requirement that knowledge make a verifiable prediction. However, the problem settings are once again different, and work on PSRs has not focused on the problems of decomposition, structural credit assignment, and bias in inductive learning that are central to this research.

Explanation-based learning (EBL) (Minton et al., 1989) is a knowledge-based approach to reasoning and learning in which classification (but not necessarily prediction) is also important. However, as noted by Russell and Norvig (Russell & Norvig, 2003), given a training example EBL "does not actually learn anything factually new from the instance. The agent could have derived the example from what it already knew, though that might have required an unreasonable amount of computation." The point here is that after learning, a system employing EBL is not capable of correctly solving any more problems than it was able to solve directly using provided background knowledge, although the computational expense may have been considerable. EBL instead makes the system more efficient at solving the kinds of problems that are input as examples by reencoding background knowledge appropriately. EBL produces processing templates that can be used to solve future problems similar to input examples more efficiently. That is, the task here is speedup learning. This is not the same task solved by AN learning. The state of knowledge after processing an example is different from the state of knowledge before processing an example. It is not simply a transformation in the form of knowledge. To see this notice that the set of examples consistent with a given AN are not the same as the set of examples consistent with an AN after modification through learning. Thus, the task here is expanding the set of solvable problems.

Stacked Generalization (Wolpert, 1990) is a method that allows classification learners (called 'generalizers' in this work) to be combined via a meta-level learner that learns to guess an answer based on the guesses retuned from all the base-level learners. This work differs from AN learning in that each of the base level generalizers is trained over the same learning problem – each attempting to learn the same target function. In contrast, ANs break a learning problem into distinct subproblems, each of which is handled by a node in the network.

Hierarchical Mixture of Experts (HME) learning (Jordan & Jacobs, 1993) trains multiple experts (learners) to solve the same problem and then combines their outputs via a series of gates in order to produce a result. By training both the experts and gates, the HME is able to learn complex decision boundaries. However, an identifiable interpretation of the purposes of the experts and gates is not guaranteed by the training algorithm. This differs from the AN representation in that a single learner addresses each learning task within a network, and in which an analytical credit assignment algorithm that respects assigned node semantics is used.

Bayesian Networks (Pearl, 1988) represent joint probability distributions efficiently by making use of conditional independence relationships among features. On the other hand, Abstraction Networks capture progressive aggregation and abstraction into equivalence classes, culminating in abstraction into a desired classification. This distinction has practical implications for the methods that operate on Abstraction Networks. First, the credit assignment procedure for ANs differs from learning in Bayes nets. During AN learning, empirical verification procedures must be invoked to determine whether a particular abstraction (intermediate equivalence classification) was accurate. When learning over a Bayes net, this is never required as the represented variables are expected to be directly observable, or are estimated (e.g. using EM). The fact that there is a level of abstraction between concepts represented at nodes in an AN and features directly observable in the environment is also a source of power for ANs. Because this level of abstraction is via an explicitly represented mechanism (the empirical verification procedure), this abstraction can be directly operated upon by learning. This means that the number of distinctions made by a given AN node can be adjusted, increasing or decreasing the level of distinction made by a particular set of equivalence classes. Also, the specific division of actual world states into these equivalence classes can be directly operated upon, potentially changing the constituency of equivalence classes. Because Bayes nets do not deal with features in terms of such explicitly represented abstractions, this type of operation is not possible when learning over Bayes nets.

9.1 Empirical Comparison with Bayesian Networks

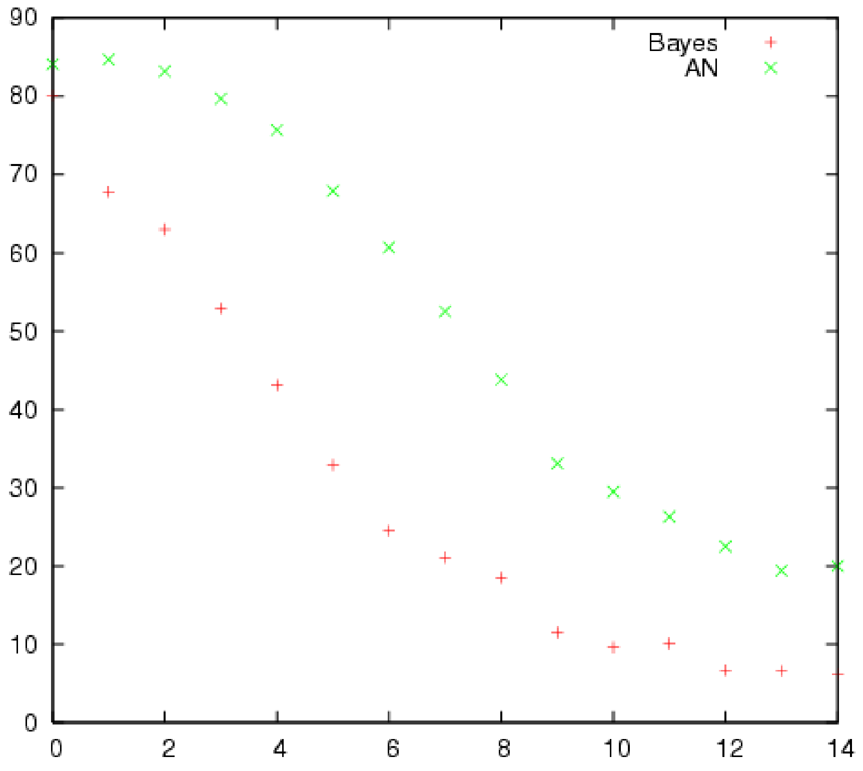


Figure 6: Error per block (Y-axis) vs. block number (X-axis) for network layer sizes 10-5-2-1, 6 values per node, 100 examples per block, averaged over 15 random repeats.

We used a randomly generated set of synthetic learning problems to compare the performance of ANs with Bayesian Networks (BNs). The environment consists of a fixed abstraction network, over which no learning will occur, that represents the correct, target content (and structure) for the problem. Given this fixed AN, we then create a separate ”learner” AN that will be initialized with random knowledge content and expected to learn to functionally match the content of the target AN. We also create a BN with identical structure and initialize the CPTs randomly. We used the Bayes Net Toolbox (BNT) implementation of BNs, learning with sequential parameter updating from complete data. Note that this means that the BN examines the true value of every feature when learning from each example, while the AN learner in general does not do so. Because the work described here is concerned only with repairing content and not structure, we do build the learners with correct structure that matches that of the fixed AN. Training proceeds by (1) generating a pseudo-random sequence of floating point numbers to serve as the observations for the input nodes of the ANs, (2) performing inference with the fixed AN, saving the values produced by all intermediate nodes as

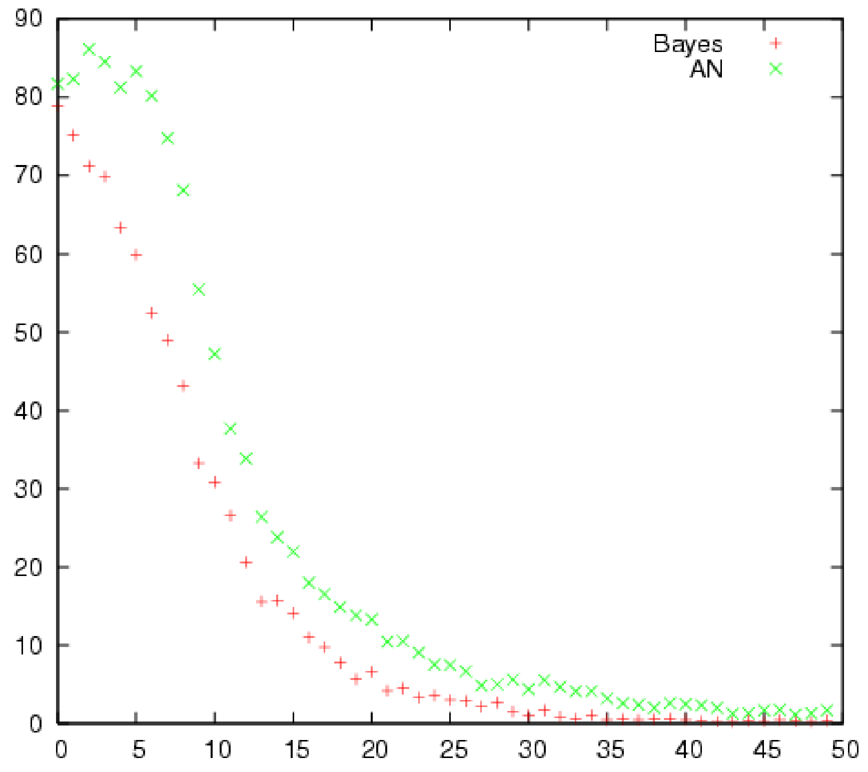


Figure 7: Error per block (Y-axis) vs. block number (X-axis) for network layer sizes 20-10-5-2-1, 6 values per node, 100 examples per block, averaged over 15 random repeats.

well as the root node, (3) performing inference with the AN and BN learners and (4) performing SCA and learning over the AN and BN learners according to the procedures described above. EVPs within the inputs of both ANs are set up to quantize the floating point observations. EVPs are not needed at non-leaf nodes in the fixed AN, since no learning will occur. EVPs at non-leaf nodes in the learning AN are set up to examine the saved output value from the corresponding node in the fixed AN, while the output values from all nodes in the fixed AN are composed into a complete feature vector for use by the BN learner. In these experiments we once again use simple table update learners within each node in the learner AN. Results obtained when inputs were drawn from a non-uniform distribution are depicted in Figures 6-8.

These results lend support to the claim that ANs offer competitive learning per example as problem size increases in at least some learning scenarios, even though fewer feature values are examined per example. For instance, in the scenario depicted in Figure 7, the AN learner examined an average of 6.98 feature values per example, while the BN learner examined all 8 non-leaf values for each example. If the resource

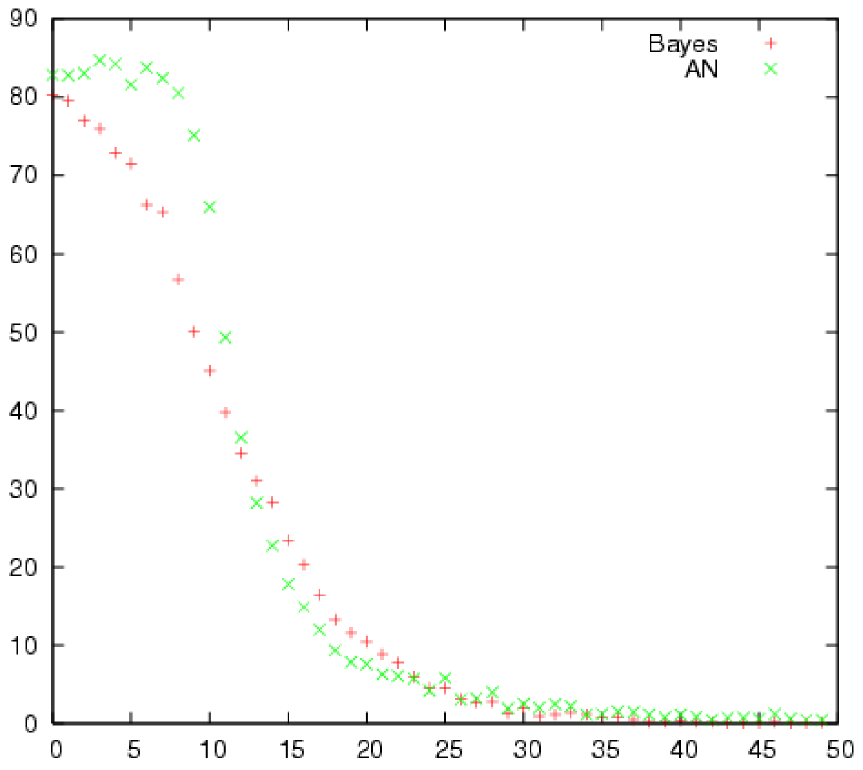


Figure 8: Error per block (Y-axis) vs. block number (X-axis) for network layer sizes 32-16-8-4-2-1, 6 values per node, 100 examples per block, averaged over 15 random repeats.

cost of obtaining feature values from the environment is significant, this property of ANs translates to resource (e.g. time) savings. However, the AN still gave comparable to better performance in terms of error rate decrease per example. In particular, it is interesting to note that the performance of the AN learner approaches and finally surpasses that of the BN learner as problem size increases from Figure 6 through Figure 8. The initial flat stage for the AN learner, most noticeable in the second two scenarios, is likely due to the need to progressively learn at each of the layers in the AN before an overall decrease in errors is realized.

The time cost of online learning and inference in ANs vs BNs as a function of network size is show in Figure 9. Here, time is shown on the Y-axis, and network size is shown on the X-axis. The X-axis values are the number of layers in the network trained, and each network is a binary tree (thus, for example, a 3-layer tree in this experiment will have 7 nodes).

Empirically, it appears that the cost of online learning and inference for BNs is increasing more rapidly than that of ANs as a function of network size. An additional

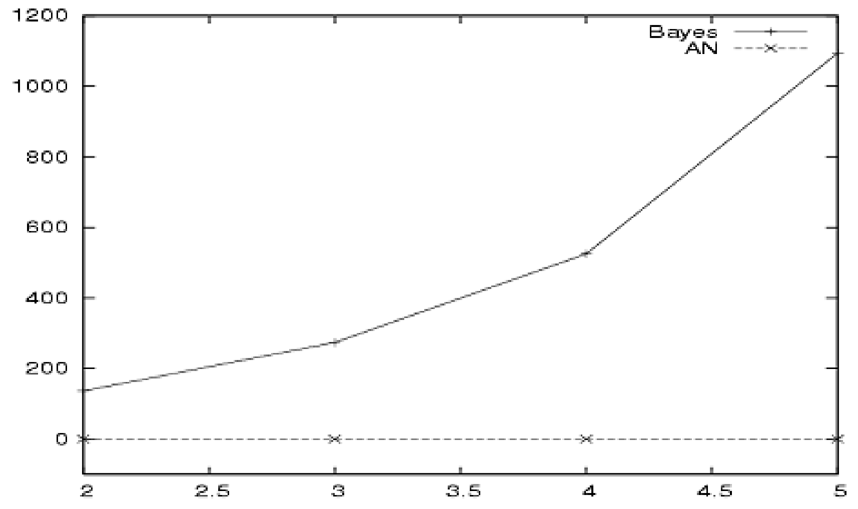


Figure 9: # of layers in binary tree vs time with 5 values possible at each node, 10 blocks of 10 examples.

note here is that probing of actual variable values was done once, and the results provided to both the AN and BN learners (though only results requested by the AN SCA procedure would actually be passed to the AN). This means that the time cost of the additional probes required for the BN learner is not reflected in the data presented in this section. If the procedures for obtaining variable values have non-negligible cost, the time used by BNs will in practice be even more significantly above that used by an AN. Of course, BNs are more general than ANs, but it does appear that there is some advantage in problems where ANs are applicable.

10 Issues and Hypotheses

Implicit in the discussion so far are a number of related issues and associated hypotheses. These are explicitly enumerated below.

- What is an effective cohesive strategy for compositional classification in terms of knowledge representation, reasoning and learning?

  > Hypothesis: Hierarchical classification knowledge representation is an effective way to store knowledge in order to both use the knowledge for classification and to allow rapid learning. Existing work provides evidence that both inference and learning are made more efficient through hierarchicalization. This work attempts to further support this hypothesis both theoretically and empirically.

- How can hierarchically represented classification knowledge be repaired automatically when incorrect results are generated?

> Hypothesis: As feedback about errors becomes available over time, this feedback can be propagated to specific knowledge elements in the classification system that can then be modified. This process of error propagation over knowledge structures is known as structural credit assignment. The claims of this work are closely tied to this issue.

- How can structural credit assignment be performed accurately over a classification knowledge structure?

> Hypothesis: In some cases, the classification is *predictive*; that is, the class label generated is interpretable as a prediction about hidden (perhaps temporally hidden, i.e. future) events or world state that can be revealed in some manner when learning is to be done. When this is true, and when the sets of equivalence classes identified by intermediate nodes in the classification hierarchy also have a predictive interpretation, this property can be exploited to allow accurate structural credit assignment. Thus, we require that knowledge elements be justified by entailing *verifiable predictions* about the world in order to be amenable to the mechanism for structural credit assignment described here. This hypothesis can also be viewed as prescriptive; when defining a classification hierarchy, one should choose intermediate nodes such that their outputs entail falsifiable predictions in order to facilitate learning.

- How can the predictive nature of the classification problem and the intermediate nodes in the classification hierarchy be exploited to perform structural credit assignment?

> Hypothesis: By explicitly encoding the predictive interpretations of the values produced by nodes in the hierarchy as *empirical verification procedures*, as defined in the section above on Abstraction Network representation, we are able to execute these procedures at learning time to determine the nodes at which faulty values were produced during an inference episode. This explicit representation of the connection between observable characteristics of the world and concepts, in conjunction with an explicit representation of the role of a concept within a classification knowledge structure, also allows us to operate directly on concepts themselves.

- What are the mechanisms of learning and inference over these predictive classification knowledge structures?

> Hypothesis: The procedures detailed above regarding Abstraction Networks are an effective means of learning and inference over predictive classification knowledge structures. Extensions will be required in order to allow for structural learning and concept modification. These procedures, in conjunction with the specified representation scheme, form a theory that addresses issues of learning, reasoning and representation in a consistent manner.

The ongoing research program on these topics is comprised of experimentation to support these hypotheses, in conjunction with extension and modification of the existing theory as required. Specific evaluation methods are discussed in a section below.

11 Open Issues

This section deals with some open issues that we intend to address with future research, including parameters that must be set in deploying ANs, a process for which we intend to develop a more principled approach. Finally, the need for extensions to cover concept modification is discussed.

11.1 Effect of ANs on Scalability

To take a particular perspective on the power of Abstraction Networks, note that scalability is a significant issue for learning techniques. For learning methods to scale to large problems, sufficient inductive bias must be introduced so that generalization from examples allows useful inferences to be drawn while respecting resource constraints. At the same time, a system designer must be careful not to eliminate or guide strongly away from the correct hypothesis when introducing bias. This can be challenging, since presumably the target hypothesis is not perfectly understood when the system is designed; otherwise there is no need for learning. One promising means of introducing appropriate bias is the use of background knowledge that, while incomplete, constrains or offers guidance about the target hypothesis. The idea of blending inductive and analytical learning by using background knowledge to add bias has been expressed in past work such as KBANN (Towell & Shavlik, 1994), EBNN (Mitchell & Thrun, 1993), MAXQ (Dietterich, 1998), ALISP (Marthi et al., 2005), and tree-structured bias (Tadepalli & Russell, 1998) and is also a useful way to view the power of Abstraction Networks. To understand how one could arrive at the idea of ANs starting from the need to scale classification learning techniques to large problems, consider the following reasoning.

Given that we have chosen the use of background knowledge as a means to introduce appropriate inductive bias, two questions arise. What is a useful form of background knowledge for this purpose? And, how should this background knowledge be used to aid the search through hypothesis space? To arrive at a possible answer to these questions, it is useful to reason backwards. First note that the goal, adding inductive bias, is equivalent to increasing generalization from examples, and generalization is the recognition and use of equivalence classes. This observation suggests the use of knowledge about equivalence classes relevant to the problem at hand to speed learning. Indeed, hierarchical knowledge-based classification techniques *do* encode such knowledge.

Increasing the a priori structuring of knowledge adds inductive bias because structure, in conjunction with progressive information loss, imposes restriction on content. However, using such a structure introduces a new level at which structural credit assignment must occur during learning. It is for this reason that we depend upon empirical determinability of the sets of equivalence classes associated with the AN nodes. That is, we require that the problem be framed such that the relevant features and equivalence classes are empirically determinable in order to facilitate credit assignment over the resulting AN. Here a connection between learning and representation is clear, even beyond the basic need for representation to capture the results of learning. Of course, designing structured representations over which learning will occur depends upon the availability of the right sort of background knowledge. Existing techniques such as MAXQ (Dietterich, 1998), ALISP (Marthi et al., 2005), tree-structured bias (Tadepalli & Russell, 1998) and layered learning (Whiteson et al., 2005) use engineered structured representations based on this type of background knowledge[7] to introduce bias.

Note that the extent to which the hypothesis space is limited by this structuring is sensitive to the amount of information loss in the function computed at each node. As long as there is information loss at each node, this hierarchy is progressively abstracting from features available at classification time to the class label, generating more general equivalence classes at each level. Future research is planned, including formal analysis and empirical study, to determine the impact of progressive information loss within ANs on scalability of learning to high-dimensional problems. This planned work is described in the section on evaluation below.

11.2 Parameters

There are a few important parameters that must be set in instantiating the AN framework. One of the open questions is how these parameters can either be automatically tuned, or how they can be manually configured in a principled manner.

1. *Nodes included in the network.* Including more nodes in the network gives parent nodes more information upon which to base their decisions. This can be good if the information is truly needed in order to accurately solve a subproblem. However, introducing unneeded information has a detrimental effect, in that it decreases generalization from examples. We base the selection of pertinent equivalence classes on background knowledge. However, in cases where this background knowledge is uncertain as to the importance of some indicated classes, adjustment may be required to achieve the best results.

2. *Network structure.* As in the previous point, network structure is based on background knowledge regarding the dependencies among the equivalence classes. Also as above, when this knowledge is less than certain, the network structure may require some tuning. One side effect of this procedure is that useful information about the actual relationships at work in the domain may be inferred, based on results with various configurations.

3. *Information loss.* As noted above, restriction of the hypothesis space occurs because information is lost at each node in the network as problem instances are

[7]In some of these cases the abstractions are temporal, i.e. relevant to action rather than state abstraction. However, these abstractions are still kinds of equivalence classes, and they are also useful in part because they enable state abstraction.

progressively grouped into equivalence classes, culminating in the desired top-level classification. The degree of information loss can be tuned at each node by choosing the number of output values allowed to be produced by the node. This parameter defines the granularity, or degree of discrimination made by each node, and controls the amount of information about the example passed through each node. Forcing too much information loss at a node will lead to failures at the parent node. Passing too much information will decrease generalization, and at the limit will become equivalent to learning over a flat representation.

4. *Definition of equivalence classes.* Beyond defining the number of equivalence classes to be produced at each node, the classification of actual situations into these equivalence classes must be defined. This amounts to defining the desired interpretation of experimental outcomes (or direct observations) as belonging to one of the available classes. There is significant interplay between this parameter and the previous one, as limited success in defining equivalence classes can be compensated for by increasing the number of classes, at the cost of generalization power.

In addition, the learners chosen to operate within nodes will also carry their own parameters that must be tuned appropriately for their assigned subproblems. Empirical studies are planned to determine the impact of each of these parameters on learning performance, discussed in the section on evaluation.

11.3 Structure Learning

We plan to extend the set of learning techniques encoded in the AN framework to allow for some learning of structural characteristics through modification of the concepts represented within an AN hierarchy. Specifically, we will design and implement algorithms for:

- Increasing or decreasing the number of output equivalence classes (i.e. tuning the information loss) at each node automatically based on the needs of the parent node and the desire for maximum generalization. Preliminary work has already been completed in this specific area, though additional extensions are required in order to provide a more complete solution.

- Changing the composition of equivalence classes identified at a node – that is, changing the definition of the empirical verification at a node such that different actual situations are identified with a given equivalence class.

The second item, changing equivalence class composition, will be dependent upon the nature of the empirical verification procedures used within the AN. This is because different sorts of empirical verification procedures have different kinds of explicit representations that will have to be operated over in different ways. For instance, all of the work performed so far has made use of empirical verification procedures that make an observation and then discretize that observation into one of the available equivalence classes. In this case, the quantization process can be modified to accommodate an increase or decrease in the number of available classes, or to adjust the membership of the classes. This specific procedure for quantization adjustment may not be applicable to any sequence of actions and observations that is used as an EVP in other settings. I will not claim to produce a *complete* taxonomy of strategies for empirical verification of knowledge, but will identify and produce automatic modification algorithms (where possible) for a subset of these strategies. Note that there are other possibilities for structural learning that will not be addressed by this research, including the automatic generation of relevant equivalence classes and the corresponding EVPs. I will determine the effectiveness of these extensions empirically, by systematically degrading the quality of knowledge engineering for a fixed problem and then running ablation studies on the extension targeted to repairing the introduced errors in background knowledge.

12 Evaluation

Additional evaluation will be required to demonstrate that structural credit assignment based on empirical verification will enable effective repair of knowledge engineered classification structures. Existing work such as layered learning, MAXQ, ALISP and TSB helps to support the general hypothesis that learning at multiple levels of abstraction is effective in improving learning scalability. An empirical demonstration that knowledge representations like those described in past work on ANs allow various inductive learning techniques to scale to large problems will add further support to this hypothesis, as well as supporting the effectiveness of the specific techniques for designing and learning over structured knowledge representations encoded in the framework. I plan to evaluate ANs in at least one additional problem setting, likely a sports prediction problem.

Experimentation in the clean, synthetic domain described previously will be useful in clearly understanding the effect of varying framework and environmental parameters, while experimentation in the realistic domains demonstrates the effectiveness of the framework for tackling real-world problems. In order to support the claim that the use of structured representations leads to improved scalability, I will perform further experiments with supervised classification learners, including ANNs and Bayesian networks. These experiments, like the preliminary ones described above, will apply supervised classification learners to problems both with and without the use of ANs, and observe and report the differences in learning rate between these cases as problem dimensionality is systematically increased.

I will determine the impact of varying each of the following aspects of the learner or learning problem while operating in the clean domain:

- Quality of knowledge engineering with respect to network structure; identification of appropriate and relevant equivalence classes; and network content.
- Type of learners used within nodes.
- Missing/incomplete vs. complete data.
- Noisy vs. perfect data.
- Degrees of mutual information among features/class label.

Rather than varying these parameters in isolation, I will systematically vary them together and use analysis of variance (ANOVA) based F-tests (Scheffé, 1959) to determine both the main and interaction effects of these parameters.

I will also formally analyze the computational complexity of learning over ANs to more completely understand their effect on learning scalability. Again the kind of scalability that is of interest is with respect to problem dimensionality. The benefits of hierarchical classification in terms of inference efficiency are already well understood. This analysis will similarly characterize the impact of hierarchicalization on classification learning in terms of the degree of progressive information loss within the hierarchy. It is this progressive loss of information that limits expressible hypotheses, and therefore the degree of information loss at each node determines the inductive bias imposed by the representation. The formal analysis to be undertaken will result in the identification of a precise relationship between information loss at the nodes of a classification hierarchy and scalability of learning.

12.1 Schedule of Research

This section outlines a proposed schedule for specific research activities leading to the dissertation.

- (Partial) taxonomy of EVP types and adaptations, plus empirical work on equivalence class adjustment (bin splitting). - Present - April 2008
- Experimentation in an additional domain. - April - June 2008
- Empirical Analysis. - June - August 2008
- Formal Analysis. - August - November 2008
- Writing. - November - May 2009

13 Expected Contributions

It is expected that the dissertation arising from the proposed research will constitute a significant contribution to the area of tree-structured bias in classification learning. More concretely, the following products are expected from this research:

1. A software library encoding the framework, including the domain-inspecific representation and methods for learning and reasoning over the representation.
2. Two or more instantiations of classification systems based on the framework, for specific domains.
3. A formal analysis of the effect of the structured knowledge representation on scalability.
4. A set of empirical studies (as described above) illuminating the effect of the representation and method on robustness to incorrect knowledge engineering and scalability to large learning problems.

5. An enumeration of more general principles of knowledge organization for the facilitation of both reasoning and learning that are expected to be more broadly applicable beyond the studied problem setting.

13.1 Applicability

Beyond the problems in FreeCiv and economic prediction given in this paper, the class of predictive classification problems to which Abstraction Networks are intended to be applicable is well-populated. Such areas as medical diagnosis, activity recognition and weather prediction are obvious additional examples where Abstraction Networks could be useful.

The existing and proposed work described here has focused specifically on predictive classification problems that deal with future events. For this reason, there is an implication that the reason that the class label and intermediate nodes in the hierarchy cannot be directly determined at classification time is because of a temporal constraint. That is, the implication is that these values pertain to future events that we must attempt to guess at before they occur. However, the class of problems to which Abstraction Networks are potentially applicable is not limited to predictions of this sort. There may be other resource limitations (economic, logistic, computational, etc.) that prevent the acquisition of some values within a classification structure when inference must be performed. It is expected that AN techniques will also be applicable to such problems, though the incorporation of some information value theory (Howard, 1966) will likely be required in order to properly handle the more general case.

Beyond increasing robustness to faulty knowledge engineering, or alternatively improving scalability of classification learning, there are several other potential benefits to learning over structured knowledge representations, especially when components of the structure have inspectable semantics, as is the case in this work. Although I do not intend to examine these points in detail as part of the work proposed in this document, I will briefly highlight them here.

- Knowledge Transfer
  Knowledge transfer may be enabled in two ways through the use of structure with semantically understood components:
  - Component transfer. When designing an AN for a new problem that shares some relevant equivalence class sets with another problem that was previously trained upon, it should be possible to reuse the relevant nodes in the new AN. Although some retraining may be needed for the knowledge to play its role correctly in the new context, it is likely that the transferred state of knowledge will be much more accurate than if randomly initialized.

  - Network transfer. When solving a problem that is substantially similar to a previous training scenario, the whole network may be transferred to the new problem. Although some modifications to the structure and/or content will certainly be required, training in the new setting should be much more rapid than if done from scratch – decreasing both designer and machine effort. The highly targeted structural credit assignment enabled by the explicitly defined component semantics should aid in quickly identifying and modifying only those portions of the network that require change, while leaving other parts untouched.

- Inspectability
  Again because knowledge is structured as a collection of components with known semantics, inspectability of the learned knowledge is enhanced. Inspectability is a useful characteristic for knowledge representations, because it has the benefit of enhancing user trust in the result of learning, and also has the potential to make ANs useful vehicles for model verification. If a researcher has some causal domain theory in terms of equivalence classes (e.g. economic indicators and market movements), ANs may be useful ways to represent and test the theories by training and observing either the structural modifications that are required, or the locations of training failures, if any.

13.2 Thesis

The central claim that will be explored in this research is that *in compositional classification problems, classification knowledge structures can be defined such that values produced by intermediate nodes entail verifiable predictions, and this characteristic can be exploited to perform effective structural credit assignment in order to correct faulty knowledge*. More specifically, I will claim that the specific representation and method developed is a way to achieve these effects. A parallel claim is that Abstraction Networks are a useful technique for combining background knowledge with inductive learning techniques in order to improve learning scalability to problems of high input dimension. For these claims to be supported, the representation based on the principle of empirical determinability must meet the needs of both reasoning and learning, synthesizing these processes at the symbolic and numerical levels. The AN framework will codify a specific theory of representing, reasoning and learning at multiple levels of abstraction, including credit assignment based on empirical determinability.

References

(2004). Leading indicators of employment. *Australian Economic Indicators*, *1350.0*.

(2005). RBC US leading economic indicator. *Economics Department, RBC Financial Group*.

Bylander, T., Johnson, T., & Goel, A. (1989). Structured matching: A task-specific technique for making decisions. *Proceedings of the IEEE International Workshop on Tools for Artificial Intelligence, Fairfax (VA), USA* (pp. 138–145).

Bylander, T., Johnson, T., & Goel, A. (1991). Structured matching: a task-specific technique for making decisions. *Knowledge Acquisition*, *3*, 1–20.

Chandrasekaran, B., & Goel, A. (1988). From numbers to symbols to knowledge structures: Artificial Intelligence perspectives on the classification task. *IEEE Trans. Systems, Man & Cybernetics*, *18*, 415–424.

Clancey, W. J. (1985). Heuristic classification. *Artif. Intell.*, *27*, 289–350.

Dietterich, T. (1998). The MAXQ method for hierarchical reinforcement learning. *Proc. 15th International Conf. on Machine Learning* (pp. 118–126). Morgan Kaufmann, San Francisco, CA.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification, second edition*. New York: Wiley.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*.

Howard, R. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, *ssc-2*, 22–26.

Iba, W., & Langley, P. (2001). Unsupervised learning of probabilistic concept hierarchies. *Lecture Notes in Computer Science*, *2049*, 39–??

Jensen, F., & Liang, J. (1994). drHugin: A system for value of information in bayesian networks.

Jones, J., & Goel, A. (2004). Hierarchical Judgement Composition: Revisiting the structural credit assignment problem. *Proceedings of the AAAI Workshop on Challenges in Game AI, San Jose, CA, USA* (pp. 67–71).

Jones, J., & Goel, A. (2005). Knowledge organization and structural credit assignment. *Proceedings of the IJCAI Workshop on Reasoning, Representation and Learning in Computer Games, Edinburgh, UK*.

Jones, J., & Goel, A. (2007). Structural credit assignment in hierarchical classification. *Proceedings of the International Conference on Artificial Intelligence (ICAI-07)*. Las Vegas, NV.

Jordan, M. I., & Jacobs, R. A. (1993). *Hierarchical mixtures of experts and the EM algorithm* (Technical Report AIM-1440).

Littman, M., Sutton, R., & Singh, S. (2001). Predictive representations of state.

Madani, O., Lizotte, D. J., & Greiner, R. (2004). The budgeted multi-armed bandit problem. *COLT* (pp. 643–645).

Marthi, B., Russell, S., & Latham, D. (2005). Writing Stratagus-playing agents in concurrent ALisp. *Proceedings of the IJCAI Workshop on Reasoning, Representation and Learning in Computer Games, Edinburgh, UK*.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings Institute of Radio Engineers*, *49*, 8–30.

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1989). Explanation-based learning: A problem solving perspective. In J. Carbonell (Ed.), *Machine learning: Paradigms and methods*, 63–118. London: MIT Press.

Mitchell, T. (1997). *Machine learning*. Boston: McGraw-Hill.

Mitchell, T. M., & Thrun, S. B. (1993). Explanation-based neural network learning for robot control. *Advances in Neural Information Processing Systems 5, Proceedings of the IEEE Conference in Denver*. San Mateo, CA: Morgan Kaufmann.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Mateo, CA: Morgan Kaufmann.

Porter, B. W., Bareiss, R., & Holte, R. C. (1990). Concept learning and heuristic classification in weakttheory domains. *Artificial Intelligence*, *45*, 229–263.

Russell, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach*. Prentice-Hall, Englewood Cliffs, NJ. 2nd edition edition.

Russell, S. J. (1988). Tree-structured bias. *AAAI* (pp. 641–645).

Samuel, A. (1957). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, *3*, 210–229.

Scheffé, H. (1959). *The analysis of variance*. New York: John Wiley & Sons.

Tadepalli, P., & Russell, S. J. (1998). Learning from examples and membership queries with structured determinations. *Machine Learning* (pp. 245–295).

Tong, S., & Koller, D. (2002). Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, *2*, 45–66.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*, 119–165.

Wasserman, L. (2004). *All of statistics : A concise course in statistical inference (springer texts in statistics)*. Springer.

Webb, R. H., & Rowe, T. S. (1995). An index of leading indicators for inflation. *Economic Quarterly*, 75–96.

Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Whiteson, S., Kohl, N., Miikkulainen, R., & Stone, P. (2005). Evolving keepaway soccer players through task decomposition. *Machine Learning*, *59(1)*, 5–30.

Winston, P. H. (1992). *Artificial intelligence (3rd ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Wolpert, D. H. (1990). *Stacked generalization* (Technical Report LA-UR-90-3460). Los Alamos, NM.