Structural Credit Assignment in Hierarchical Classification

Joshua Jones and Ashok Goel
College of Computing
Georgia Institute of Technology
Atlanta, USA 30332
{jkj, goel}@cc.gatech.edu

***Abstract*— Structured matching captures the bottom-up pattern of the hierarchical classification techniques used in many knowledge systems: subsets of features describing the state of the world are progressively aggregated into equivalence classes in an abstraction network, until the required decision is made at the root node. In this paper, we describe a supervised learning technique for automatic repair of abstraction hierarchies in structured matching. Note that the task of repairing an abstraction network involves the structural credit assignment problem. We describe the property of empirical determinability, and show that the design of an abstraction network according to this property enables structural credit assignment.**

I. Introduction

AI research on knowledge systems has led to the development of several techniques for hierarchical classification [9]. Structured matching [3] captures the bottom-up pattern of some of these techniques. Briefly, in structured matching subsets of features describing the state of the world are aggregated into equivalence classes that form the leaf nodes in an abstraction network, the features describing outputs of the leaf nodes are aggregated into equivalence classes at the next higher level in the hierarchy, and so on, until the required decision is made at the root node. The use of structured matching in a variety of knowledge systems shows the efficacy of the method.

In this paper, we describe an automated supervised learning technique for automatic repair of these hierarchies (here called ANs) when the decision at the root node turns out to be incorrect. The results we describe in this paper are for repair of AN *content*, not its *structure*. Note that the task of repairing AN content involves structural credit assignment. Thus, a central research question here is what properties of the equivalence classes in an AN are desirable in terms of being useful for performing structural credit assignment over the hierarchical structure?

II. The Learning Problem

A. The FreeCiv Domain

Let us consider a situation from FreeCiv (www.freeciv.org), a turn-based strategy game. 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.

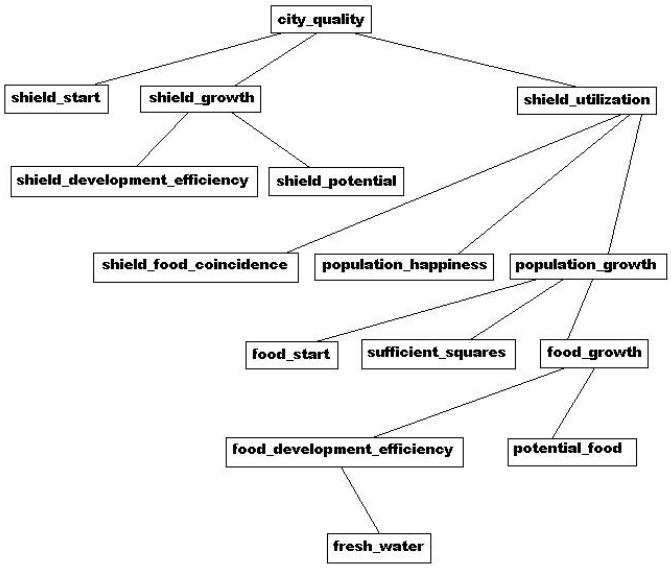

Fig. 1. City Estimate Abstraction Network

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. We have designed an agent that plays FreeCiv, and have evaluated the use of ANs (defined in more detail below) in learning to make predictions about the resource production rate of a potential city over time. Figure 1 illustrates the AN used by our agent.

B. The Learning Task

Let T be a discrete random variable representing the class label. Let $S = \{s : s \text{ is empirically determinable and } h[T] > h[T|s]\}$, where $h[x]$ denotes the entropy of x. S is a set of random variables that have nonzero mutual information with the class label and are "empirically determinable." 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

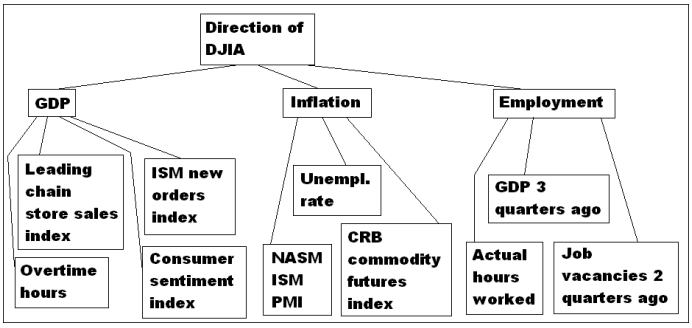

Fig. 2. DJIA Abstraction Network

govern the relationships among the variables in S. 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 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. 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.

Learning is required because the distributions $\mathcal{P}(s|K), s \in S \cup T, K \subseteq S$ are not given, but must be inferred from experience. After the class label prediction is produced, the correct class label is given, along with some additional quantity of resources R_a so that additional experiments may be performed before the next task instance is presented. The relative costs of performing experiments before predicting the class label may not be the same as the relative costs of performing experiments 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. Success at this task is measured in terms of the final classification accuracy achieved, the rate at which accuracy improves as examples are presented, and the resources saved.

C. DJIA Prediction

In this domain, we are interested in classifying a current economic status as described by various economic indicators (see Figure 2) 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 2 based on some studies of economic indicators [1][11][2]. 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 2 can be observed before classification, while the remainder are future values at classification time, available only in retrospect.

III. Learning Using Abstraction Networks

The learning technique is based on a few observations, assumptions, and hypotheses :

1) Evaluation takes place during learning, i.e. learning must occur on-line.
2) 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 restricted version of the FreeCiv domain considered in this paper, only the latter consideration is significant, as a fixed set of variables will be (freely) available for use in classification.
3) Although the specific dependencies among the relevant variables are not given, here we will assume that the *structure* of the dependencies among the variables is 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 background knowledge about these relationships is available.
4) 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 is not critical to the learning method, but it simplifies its presentation.
5) In terms of learning speed, a technique that exploits the available knowledge to maximize generalization from examples will outperform a technique that does not.

The benefit of structuring knowledge in a hierarchy according to the known dependencies is well understood in terms of inference speed, as studied in work on structured matching [3][9]. The benefit for learning is that structure introduces a restriction bias, because the hierarchical representation has less expressive potential than a flat representation. 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 members of S. 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.

A. Learning Method

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. The structure used in the FreeCiv problem is depicted in Figure 1, while the structure used for DJIA prediction is shown in Figure 2. 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 represent 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. Note that the extent to which the hypothesis space is limited by this structuring is sensitive to the amount of information loss[1] 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 a priori to the class label, generating more general equivalence classes at each level.

In a given task instance, the values of the leaf nodes are fixed by observation. 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.

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 the predictions are checked.
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.

Notice that this procedure has a base case when the leaves are reached, as their true values were obtained before prediction, 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 will then be modified.

Another point to notice here is that the specific procedure for the modification of local knowledge is not specified. 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.

There are a few other parameters that must be set.

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 use of this procedure is to infer useful information about the actual relationships at work in the domain 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 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 decreases generalization, and at the limit

[1]By "information loss" here, we mean that the inputs to a node cannot be recovered given the output value of the node and the knowledge stored within the node – that is, more than one set of inputs will be mapped to the same output.

becomes 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.

IV. Experiments

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. In this way, we can compare error rate around the beginning of a training sequence with the error rate around the end of that sequence.

In all problem examples described here, we use simple table-based learners within each node. These "table update" learners have dimensions equal to the associated node's fanin. Inference is performed by indexing the table based on inputs to the node, and learning is performed by modifying table entries, with a thresholding mechanism that refrains from updating a table entry until some consistent feedback confirms that the change is necessary and is unlikely to be due to noise.

A. Application to FreeCiv

In the case of FreeCiv, we have compared the performance of an unstructured, flat table-based learner to the learner that makes use of an AN-based knowledge representation. Each turn of each game in the FreeCiv setting is treated as a separate example. This means that errors are counted on each turn of each game by producing a classification 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 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 should 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 sequences of games into multi-game blocks for the purpose of evaluation; the block sized 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 yield 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 environments.

The results of the experiment described above are summarized in Table I, 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 many games. For the two scenarios, the average improvement in error rate is 26.5%, 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,

	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 I

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.

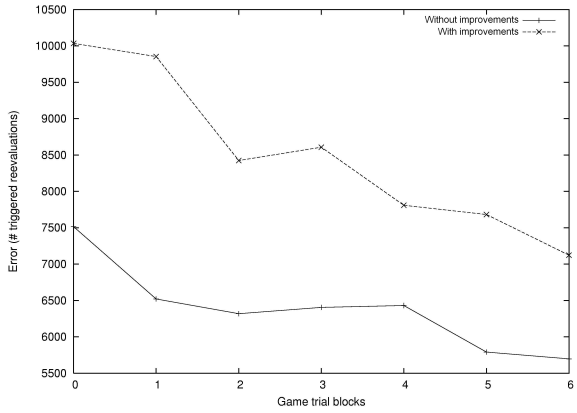

Fig. 3. Average Error Rates By Block in Each Trial

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 do offer an advantage in terms of allowing learning to occur more quickly in a large state space.

B. Application to DJIA prediction

We have also experimented with ANs in the DJIA economic problem domain. We used data from Jan 1960 - Nov 2005, yielding a total of 497 training examples. We observed a 23.4% decrease in error, comparing blocks consisting of the first 213 and the last 213 examples. These preliminary experiments help 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.

V. DISCUSSION

The use of decomposition to manage large problems that would otherwise be infeasible is a fundamental technique in AI. The approach taken in this work is similar to a common pattern of decomposition in AI systems characterized as structured matching in [3]. This particular type of decomposition is well suited to the type of classification problem we examine here. The contribution of this work is the development of a new method for scalable learning through use of domain knowledge. Work on hierarchical RL [4] has a similar goal, but in a different problem setting and with the use of a different kind of domain knowledge. In hierarchical RL procedural knowledge in the form of a hierarchy of temporally abstract actions is used. 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 Probabilistic Concept Hierarchies [5][6] makes use of knowledge representations centered around the generation of progressively more abstract equivalence classes, as does our work. While the representations share an intuition about concept encoding, 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.

Layered learning [12] 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 all of the learners in the decomposition hierarchy are trained as a complete system in the actual target domain. Some notion of empirical determinability is implicit in hierarchies built to be trained with coevolution, as the evaluation functions used for each component must be evaluable based on available percepts. Here, we are explicit about the need to design decompositions specifically around this property, and we allow the use of arbitrary (possibly heterogeneous) learners within each component. An additional distinction is that ANs focus on progressive state 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.

Knowledge-based ANNs [10] and Explanation-based NNs [7] both apply background knowledge in order to speed up learning in supervised classification problems. In KBANN learning, neural network structure and initialization are in-

formed by background knowledge in the form of Horn clauses. However, once training begins, any semantic interpretation of the nodes is dropped, and may (in general) be arbitrarily changed during learning. This difference with the work described here gives KBANN learners more flexibility, but also limits the bias introduced. There are also other advantages to using fixed semantics for intermediate nodes, such as the potential for transfer of partial networks to new problems and inspectability of knowledge. Also note that in KBANN learning, the background knowledge available is somewhat richer, including (limited) information about the nature of the dependencies between relevant variables as well as the structure of those dependencies. EBNN also requires that the background knowledge contain information about the specific relationships between pertinent variables, in the form of a collection of pre-trained ANNs. EBNN uses the background knowledge to guide the search through hypothesis space. As with KBANN, there is no constraint on the semantics of nodes within the target ANN that is trained.

A. Empirical Comparison with BNs

Bayesian Networks [8] (BNs) represent joint probability distributions efficiently by making use of conditional independence relationships among features. On the other hand, ANs 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 ANs. First, the credit assignment procedure for ANs differs from learning in BNs. During AN learning, empirical verification procedures must be invoked to determine whether a particular intermediate abstraction was accurate. When learning over a BN, 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), these abstractions can be directly operated upon by learning, though these procedures are not detailed here. 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 BNs do not deal with features in terms of such explicitly represented abstractions, this type of operation is not possible when learning over BNs.

We used a randomly generated set of synthetic learning problems to compare the performance of ANs with 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. By "functional matching", we mean that we will only measure error based on whether the learner AN is able to produce output values that match those of the fixed AN – we will not actually inspect the contents of individual nodes. We also create a BN with identical structure and initialize the CPTs randomly. We used the Bayes Net Toolbox (BNT; bnt.sourceforge.net) 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 well as the root node, (3) performing inference with the learner AN and (4) performing SCA and learning over the learner AN 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 Figure 4, along with the results of linear regression.

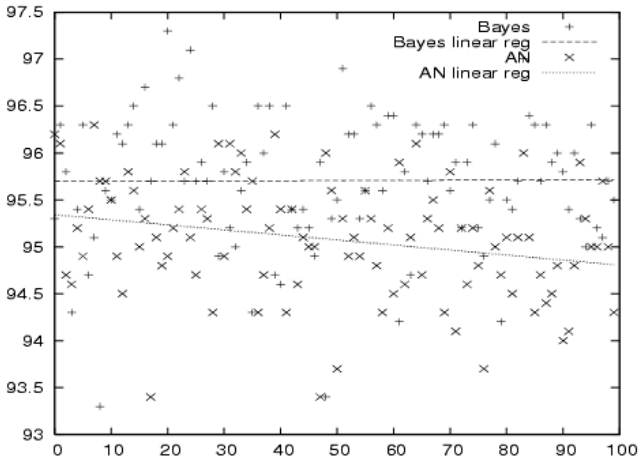

Fig. 4. Error per block (Y-axis) vs. block number (X-axis) for network layer sizes 10-5-2-1, 20 values per node, averaged over 10 random repeats, 100 blocks of 100 examples.

These results lend support to the claim that ANs offer competitive learning per example in at least some learning scenarios, even though fewer feature values are examined

per example. For instance, in the scenario of Figure 4, 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. However, the AN still gave comparable to better performance in terms of error rate decrease per example. If the resource cost of obtaining feature values from the environment is significant, this property of ANs translates to resource (e.g. time) savings.

The time cost of online learning and inference in ANs vs BNs as a function of network size is show in Figure 5. 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).

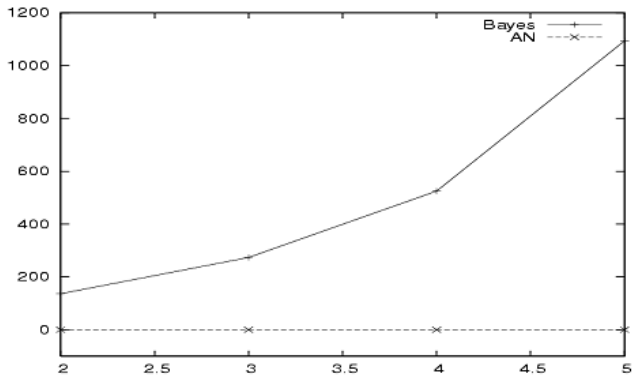

Fig. 5. # of layers in binary tree vs time with 5 values possible at each node, 10 blocks of 10 examples.

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 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.

VI. Conclusions

This paper presents a learning technique for automatically repairing the abstraction network in bottom-up hierarchical classification methods such as structured matching. Through the property of empirical determinability, the AN method prescribes representations that directly address the need to perform credit assignment over the resulting top-level structure. The experiments described here provide evidence that the AN representation and accompanying learning technique are sufficient for knowledge repair, and secondarily that the technique provides benefits in terms of efficiency. Further, applicability in multiple domains supports the generality of the method. This means that if a problem framing and set of components can be defined according to the property of empirical determinability for a given problem with the proper characteristics, the resulting AN can be repaired automatically.

References

[1] anon., "Leading indicators of employment," *Australian Economic Indicators*, vol. 1350.0, April 2004. [Online]. Available: http://www.abs.gov.au/ausstats/abs@.nsf/0/

[2] ——, "RBC US leading economic indicator," *Economics Department, RBC Financial Group*, 2005. [Online]. Available: http://www.rbc.com/economics/market/pdf/usindicator.pdf

[3] T. Bylander, T. Johnson, and A. Goel, "Structured matching: a task-specific technique for making decisions," *Knowledge Acquisition*, vol. 3, pp. 1–20, 1991.

[4] T. Dietterich, "The MAXQ method for hierarchical reinforcement learning," in *Proc. 15th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1998, pp. 118–126. [Online]. Available: citeseer.ist.psu.edu/dietterich98maxq.html

[5] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, 1987.

[6] W. Iba and P. Langley, "Unsupervised learning of probabilistic concept hierarchies," *Lecture Notes in Computer Science*, vol. 2049, pp. 39–??, 2001. [Online]. Available: citeseer.ist.psu.edu/296596.html

[7] T. M. Mitchell and S. B. Thrun, "Explanation-based neural network learning for robot control," in *Advances in Neural Information Processing Systems 5, Proceedings of the IEEE Conference in Denver*, C. L. Giles, S. J. Hanson, and J. D. Cowan, Eds. San Mateo, CA: Morgan Kaufmann, 1993. [Online]. Available: citeseer.ist.psu.edu/mitchell92explanationbased.html

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

[9] M. Stefik, *Introduction to knowledge systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995.

[10] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial Intelligence*, vol. 70, no. 1-2, pp. 119–165, 1994. [Online]. Available: citeseer.ist.psu.edu/towell94knowledgebased.html

[11] R. H. Webb and T. S. Rowe, "An index of leading indicators for inflation," *Economic Quarterly*, no. Spr, pp. 75–96, 1995.

[12] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone, "Evolving keepaway soccer players through task decomposition," *Machine Learning*, vol. 59(1), pp. 5–30, 2005.