, , 1–21 ()

Feature Selection *vs* Theory Reformulation: a Study of Genetic Refinement of Knowledge-based Neural Networks

BRENDAN DAVIS BURNS bburns@gte.com
GTE Laboratories, Waltham, MA 02454

ANDREA POHORECKYJ DANYLUK andrea@cs.williams.edu
Williams College, Williamstown, MA 01267

Editor: Floriana Esposito, Ryszard Michalski and Lorenza Saitta

Abstract. Expert classification systems have proven themselves effective decision makers for many types of problems. However, the accuracy of such systems is often highly dependent upon the accuracy of a human expert's domain theory. When human experts learn or create a set of rules, they are subject to a number of hindrances. Most significantly experts are, to a greater or lesser extent, restricted by the tradition of scholarship which has preceded them and by an inability to examine large amounts of data in a rigorous fashion without the effects of boredom or frustration. As a result, human theories are often erroneous or incomplete. To escape this dependency, machine learning systems have been developed to automatically refine and correct an expert's domain theory. When theory revision systems are applied to expert theories, they often concentrate on the reformulation of the knowledge provided rather than on the reformulation or selection of input features. The general assumption seems to be that the expert has already selected the set of features that will be most useful for the given task. That set may, however, be suboptimal. This paper studies theory refinement and the relative benefits of applying feature selection versus more extensive theory reformulation.

Keywords: Theory refinement, theory reformulation, feature selection, knowledge-based neural networks, genetic algorithms.

1. Introduction

This paper studies theory refinement and the relative benefits of applying feature selection versus more extensive theory reformulation. We have developed two systems – INDiGENT and TNT-INDiGENT – that refine theories that are represented as knowledge-based neural networks. INDiGENT refines a theory by focusing exclusively on feature selection. TNT-INDiGENT refines the entire network. We apply INDiGENT and TNT-INDiGENT to a number of domains in order to investigate the relative merits of the two different forms of theory refinement. In these domains we find that feature selection and theory reformulation lead to equivalent improvement in the domain theory, as measured by classification accuracy. This result runs counter to the intuition that more extensive modification would lead to better performance. The feature selection approach has two additional benefits, which are: (1) simplicity of the refinement algorithm, and (2) minimal change to the original theory, which is of importance for understandability of the results.

We begin with a description of the INDiGENT algorithm. The next section discusses the results of applying INDiGENT to domains from molecular biology and credit authorization. We demonstrate that, when taken as a theory revision system, INDiGENT's focus on feature selection gives performance comparable to or greater than theory revision systems that manipulate the theories themselves. In addition to presenting this comparison, we introduce TNT-INDiGENT, our implementation of a theory revision system combining genetic algorithms and neural networks. We present results of applying TNT-INDiGENT alone and also in combination with INDiGENT. We conclude that the feature selection alone gives results as powerful as more extensive revision.

2. INDiGENT

This paper focuses on two algorithms, INDiGENT and TNT-INDiGENT. Both are designed to be any-time algorithms. As discussed in Opitz (1995), the concept of an any-time machine learning system is an algorithm which has no discrete ending point and whose results are expected to improve at each time step. The strength of an any-time approach is that a user may determine the appropriate ratio of accuracy to time expended for their particular task. The concept of any-time learning allows a user to decide when the algorithm's solution is "accurate enough." Both algorithms described here use a genetic search to discover optimally (or nearly optimally) accurate neural networks. INDiGENT searches the space of potential input features. TNT-INDiGENT performs a genetic search on the space of potential neural topologies.

INDiGENT (Improving kNown Domains with Genetically Engineered Neural feaTures) focuses on evolving the input features that a network chooses to use. It has been demonstrated that many errors can be corrected simply by modifying the bottom-most nodes in a network (Towell & Shavlik, 1992). Expanding on this idea, the INDiGENT algorithm seeks to search the space of subsets of the total feature set in order to discover the subset of these features which creates the most accurate classifier. Discovering this optimal subset is a challenge, however, since an exhaustive search of all of the possible subsets is intractable. A genetic algorithm is ideal for this search since the input features can be easily represented as a genotype and the feature subset error space potentially contains many local minima.

INDiGENT performs knowledge-based feature selection. That is, it is guided, in part, by expert knowledge about the classification domain. It performs a genetic search of the feature space in order to select those features that best classify data from the domain, when considered *in conjunction* with the given domain theory.

INDiGENT uses knowledge-based artificial neural networks (Towell et al, 1990) to represent expert theories that have been previously written in a simplified form of Horn-clause logic. There are a number of systems that use neural network encodings to refine a variety of representation types, including finite state automata (Maclin & Shavlik, 1993), certainty-factor rule bases (Mahoney, 1996), and first-order horn clause logic (Towell et al, 1990). There has also been work done on genetically refining neural network topologies. Opitz (1995) describes the REGENT system, an extension to KBANN, which refines domain theories through the addition of rules between the input and output features. REGENT does not explore the modification of the network's input set as INDiGENT does. DAID (Towell & Shavlik, 1992) was an earlier extension of KBANN. It does modify the set of input features. The system is significantly different from INDiGENT, however, since it chooses the input nodes to add through induction on erroneous hidden nodes and the feature set, rather than utilizing genetic search.

The RAPTURE system (Mahoney, 1996) was developed to genetically refine certainty factor rule bases. Like REGENT, RAPTURE genetically refines the topology of hidden nodes in the networks. RAPTURE does attempt to improve the set of input features by adding relevant (as judged by the information gain metric (Quinlan, 1986)) nodes if the network fails to reach a certain level of accuracy. However, RAPTURE's treatment of input set selection is significantly weaker than INDiGENT's, since it only adds individual input nodes as a secondary refinement method and does not explore the entire space of possible input feature sets.

2.1. Finding an Accurate Input Feature Subset

INDiGENT focuses upon evolving the input features that a network chooses to use. It begins with the generation of an initial genotype for later use in the genetic algorithm. INDiGENT's initial genotype includes all of the input features referenced in the expert's domain theory, as well as features not included in the initial domain theory but that have been deemed "important" by Quinlan's C4.5 decision tree generator[1]. The initial genotype is simply a binary string where each bit represents one input feature and the corresponding value represents its presence or absence in the subset. INDiGENT takes this initial genotype representing the input features and mutates it to create the initial population of genotypes. INDiGENT's default population size is twenty.

INDiGENT then creates a knowledge-based neural network for each of the twenty genotypes in the population. Each network represents the domain theory initially provided by an expert, with the exception of the input features. The input features reflect those sets represented by the twenty genotypes in the population. The input features in a genotype are incorporated into the domain theory by fully connecting the input features to each node in the first hidden layer of the network.

After the initial population of networks has been created, the population of networks is evaluated using N-fold cross validation. The accuracy of the network is reported as the average accuracy of all N folds. This accuracy is taken to be the fitness value for the network (and thus for the genotype representing a particular feature set).

After the networks have been evaluated, the networks with the worst and best fitness values respectively are noted. Two parent genotypes are selected. The parents are chosen in a manner proportional to their fitness values[2]. The parent genotypes are then crossed over to create a new (child) genotype.

Once a child genotype has been created, it is mutated. Its corresponding network is built, trained, and tested as before. If the resulting fitness value is higher than the fitness of the worst genotype in the population, the worst genotype is replaced by the child. A search is then performed to find the new best and worst fitness values. This process of creation, evaluation, and (potential) replacement is repeated for an arbitrary number of generations. At each stage the best network is saved to a file. Note that the network saved is the one initially created from the genotype, rather than a trained network. Essentially, "best" networks represent those that can be best trained to achieve high classification accuracy. When a specified number of generations has been reached, the best network found is presented as INDiGENT's result.

2.2. *Crossover in* INDiGENT

Crossover in INDiGENT is performed as follows: Input features appearing in *both* parent genotypes are added to the child's genotype. If a feature appears in only one parent's genotype it has a chance, proportional to the fitness value of the parent genotype, of being added to the child. Rather than merging the genotypes together as INDiGENT does, many genetic algorithms operate by splicing two fractions of the parent genotypes together to form the child's genotype. In reality both of these methods are fairly similar. Both produce in the child the logical *and* of the two parent genotypes in addition to a number of elements existing solely in a single parent. INDiGENT's method of crossover, however, doesn't restrict the placement of the features from the single parents to proximity with each other. The method of crossover, therefore, allows for a greater possible number of genotypes to be created from a single pair of parents.

2.3. *Mutation in* INDiGENT

INDiGENT mutates a genotype as follows: For each potential input feature, a random decision is made about whether to add or remove the feature. INDiGENT favors adding input features rather than deleting them. The reasons for this are twofold: It is assumed that the given domain theory is largely correct. Therefore, nodes mentioned by the domain theory are more likely to be useful to the network than nodes that are not mentioned by the domain theory. It is likely that adding unknown inputs will be more beneficial to the network than removing known inputs. It has also been shown that KBANN networks have the ability through back-propagation to eliminate detrimental antecedents by reducing the weight of their connecting synapses (Opitz & Shavlik, 1993). Therefore, having a genetic algorithm performing the same task is at best redundant and at worst detrimental to performance. After a new feature set is selected, the corresponding network is constructed. Because the networks are fully connected, each of the new inputs which have been added are linked to the first hidden layer. This process allows the features, which are not present in the domain theory, to impact the network's accuracy.

3. Experiments

This section describes the experiments to evaluate the performance of INDiGENT.

3.1. Introduction to the Domains

Three of the four data sets used in these experiments come from the biologic domain and are concerned with the identification of certain features given a string of nucleotides (DNA or RNA). The domain is an important one to examine because research such as the Human Genome Project is sequencing far more DNA than can be effectively examined by humans. An automated system for the extraction of features from DNA is therefore a useful tool for biologists. Additionally, the domains are effective areas to test INDiGENT and TNT-INDiGENT because they are taken from the "real" world and therefore contain the irregularity and noise which is the challenge of any classification system. Finally, these domains are useful for the evaluation of INDiGENT and TNT-INDiGENT since several other systems have been developed to classify these domains and have established benchmark accuracies against which to compare INDiGENT and TNT-INDiGENT's results. The final data set concerns the prediction of the granting of credit to borrowers in Japan. This domain differs in many ways from the biologic domains. Its features are not in physical proximity as nucleotides are, and therefore the patterns recognized by the neural network are more complex and ephemeral. Also, the number of features for the domain (10) is significantly smaller than in any of the biologic domains. This too is an important difference because the reduction from or addition to the feature set represents a significant corresponding increase/decrease of information.

3.1.1. The Promoter Domain. The first domain is the problem of deciding if a promoter exists at a given starting point in a sequence of E. Coli DNA[3]. The initial domain theory was obtained, along with the dataset, from the machine learning archive at the University of Wisconsin[4]. The domain contains 234 positive examples, each from a different source, and 4,921 negative examples generated by sliding a 57-nucleotide window across a promoter-free sequence of 4,977 nucleotides. From this data ten folds of data were generated using the same random fold numbers that had been used in KBANN and REGENT experiments at the University of Wisconsin, benchmark results against which INDiGENT and TNT-INDiGENT are compared. The dataset contains 234 positive examples and 702 negative examples (1:3 ratio) randomly distributed throughout all ten folds.

3.1.2. The Ribosome Binding Site Domain. The second domain that was examined was the identification of a ribosome binding site (RBS) at a point in a sequence of m-RNA[5]. As with the promoter domain, the dataset, folds, and domain theory were all obtained from the machine learning archive at the University of Wisconsin. In the RBS domain there are 366 positive examples and 1511 negative examples that were generated by sliding a 49-nucleotide window along a RBS-free sequence of 1559 nucleotides. As with the promoters, the random folds were the same as the folds used in the KBANN and REGENT experiments.

3.1.3. The Splice Junction Domain. The third domain that was used is the identification of splice junctions[6]. The task for this domain is to determine whether a given location marks the boundary between a section that is to be retained (exon) and a section to be removed (intron), whether the boundary falls between an intron and an exon, or if the location is not a boundary. The splice junction domain differs from the previous two domains because there are three classification categories instead of two. The three classes are: (1) EI, the boundary between a section that is to be retained (exon) and a section to be removed (intron), (2) IE, the boundary between an intron and an exon, (3) Neither, the location is not a boundary.

The splice junction domain consists of 1200 strands of 60 nucleotides. There are 400 positive examples and 800 negative examples. Five random folds were used which were created locally. A subset of the total dataset of 3190 was used in order to maintain a 1:3 ratio of positive (IE and EI) and negative (neither) classifications. This is the ratio of positive to negative used in the RBS and Promoter data sets as well. Five folds were used instead of ten because the smaller size of the data set created a nearly one-to-one correlation of data points and features. This data set differs slightly from those above since the figures reported for KBANN and REGENT experiments show performance on a different data set with the same distribution of positive and negative examples.

3.1.4. The credit prediction domain. The final domain concerns the prediction of credit granting for potential borrowers in Japan.[7] The features represent a variety of information, such as age of the applicant, where the applicant is employed, and the item to be purchased. The credit screening data consisted of 125 examples, 40 positive and 85 negative. As with the Splice Junction data set, five folds were used instead of ten. Unlike the biologic domains, there are no directly comparable uses of this data to use a benchmark.

3.2. Measuring Performance

The goal of INDiGENT is to create a neural network which, when given a string of input data, returns the appropriate classification for the data. The measure of accuracy is the percentage of examples for which the classifier judges correctly. In the biologic domains it is important to note that this classification is for a particular position in the DNA strand. This creates a slightly artificial classification scheme. Though the negative examples were generated from a single continuous strand of nucleotides, the positive examples were all distinct individual strands. As a result, the accuracy may not exactly correspond to the algorithm's ability to discover positive classifications in a single long strand of DNA. It is important also to note that while INDiGENT and TNT-INDiGENT were run for 250 generations[8], the numbers reported for REGENT represent runs of 500 generations.

3.3. *Experimental Setup*

For the biologic domains in the experiment, ten runs of INDiGENT for 250 generations were conducted in each domain. For the credit domain the runs were conducted over 50 generations. The training data were shuffled prior to each training iteration in order to encourage generalization in the networks. The best network at each generation was recorded. The best network after the final generation was used as the result of each run, and its accuracy was evaluated (by first training and then testing the network).

3.4. *Numerical Results*

Before discussing the results of the experiments described above, it is important to note work that we did prior to running the experiments. In preliminary runs, we graphed the average performance of the genetic search for all ten runs of INDiGENT and found the following. While the average performance of the genetic search over all ten runs of INDiGENT is similar to the expected performance of genetic search, the individual performance of each run of INDiGENT displays one or two steep jumps upward in performance and then no improvement. Additionally, the majority of the improvements come in the first 150 generations and the later generations tend not to improve the performance significantly. Graphs of the performance of REGENT show similar results so the choice of 500 for REGENT's evaluation seems arbitrary. The reason that INDiGENT was only run for 250 generations came from noticing that longer run time didn't necessarily improve performance in a manner which was proportional to the computational expense. In fact, in some cases the performance decreased after further generations because the population became increasingly homogeneous.

This observation is an interesting commentary on the concept of the any-time algorithm which is discussed in Opitz (1995). The stated goal of an any-time algorithm is that the algorithm produces steadily increasing performance which corresponds to the length of computation. The performance described here shows that the goal of performance increasing in proportion to computation is not a possible goal. Instead as generations pass, improvements in performance become increasingly expensive. Not only is the time needed for increasing accuracy infinite in the limit, but more practically, after 200 generations the improvement with further computation is generally not worth the expense. Of course the design of the any-time algorithm is such that computation may be halted at the user's discretion, and the above observation is not a condemnation of the any-time approach but rather guidance for the appropriate number of generations to run a genetic search algorithm.

3.4.1. Biologic domains. Table 1 shows the final average accuracy of INDiGENT after 250 generations compared to the accuracy of several other neural network classification systems on the same data. Table 1 shows that feature selection has the ability to improve the initial domain theory to the same degree as more extensive theory revision systems like REGENT. Table 2 gives additional information about INDiGENT's feature sets.

Table 1. INDiGENT's accuracy compared to several other systems (with standard deviations in parentheses).

System	Promoter	RBS	Splice Junction
INDiGENT	95.56% (0.23)	92.76% (0.42)	95.32% (0.33)
REGENT	95.83%	92.17%	95.92%
TopGen	94.76%	91.85%	95.08%
KBANN	93.70%	91.05%	94.75%

Table 2. Differences between initial feature sets and INDiGENT's final feature sets.

	Promoters	RBS	Splice Junctions	Credit Screening
% Change	+26.3%	+68.4%	+33.3%	-12.0 %
% of Total #	71.9%	65.3%	46.7%	72.0 %

3.4.2. Credit screening domain. Table 3 shows that feature selection improved performance over the initial domain theory, as well as the inductive learner C4.5. Additionally, as in the biologic domains, the use of INDiGENT's features as a filter improved C4.5's performance.

Table 3. INDiGENT's accuracy on the credit domain.

System	Acc: INDiGENT features	Acc: domain theory features
INDiGENT	97.6 %	91.2%
C4.5	72.5 %	69.5 %

3.5. Discussion of Results

INDiGENT has shown itself to be an effective and simple method for theory revision. INDiGENT attains a level of performance that is on par with REGENT, a more extensive theory revision system. This is an appealing result for a number of reasons. Foremost, feature set revision is an understandable and explainable form of theory revision. Traditionally, many theory revision systems and (especially) neural network systems have suffered from an inability to explain their accuracy. While numbers showing improvement may document the improvement of a theory it is hard for a real life manager of a telephone system, dam, or power plant to relinquish control to a system whose programmers promise high accuracy but can't really explain how it works. Instead of this black box, INDiGENT's feature set revision improves upon a domain theory without altering it beyond recognition. In doing so it aids not only the understanding of where improvements are created but also the ease of application to the solution of real-world problems. While maximal accuracy is a goal to be strived for, it is ultimately useless if a system is never used.

INDiGENT's results are interesting in the effect the learned feature sets have on inductive learning systems. As shown in Table 4, the voted INDiGENT feature set has no statistically significant effect on the performance of C4.5 and Ripper when compared with the application of those algorithms to the total possible feature set. However, as shown in Table 5, the classifiers learned are smaller (and, thus, in a sense "simpler") when given as input only the features selected by INDiGENT.

Table 4. INDiGENT feature set accuracy used with several other systems.

	Accuracy on RBS Data		Accuracy on Promoter Data	
System	all features	INDiGENT features	all features	INDiGENT features
C4.5	88.3%	88.6%	88.3%	89.0%
RL5.6	87.84%	87.76%	87.92%	88.09%
Ripper	91.13%	90.92%	88.01%	88.33%

Table 5. Comparison of sizes of learned classifiers.

	C4.5	RL5.6	Ripper
RBS			
INDiGENT features	126.5 (pruned)	27.1	9.4
all features	133 (pruned)	29.8	13.1
Promoters			
INDiGENT features	63 (pruned)	25.8	7.8
all features	63 (pruned)	26.4	9.8
Credit Screening			
INDiGENT features	13.1 (pruned)	N/A	N/A
all features	13.3 (pruned)	N/A	N/A

4. Limitations of INDiGENT

As a feature selection algorithm, INDiGENT is both helped by and limited by its dependence on the given domain theory. Because INDiGENT is aware of domain theory information it necessarily selects feature sets which perform well in conjunction with that theory. Therefore, when the selected feature sets are used with machine learning systems which do not utilize an initial domain theory, they may be too restrictive. If a machine learning system does not receive a domain theory *a priori* it may be necessary that it receive more information through the data in order to induce a theory of its own. INDiGENT's feature selection, therefore, does not simply select features relevant to the domain, but it actually selects features relevant to the domain theory. It is understandable, then, that in the absence of a domain theory, the selected feature sets may be less appropriate.

When taken as a theory refinement algorithm, INDiGENT clearly does not have the ability to correct anything more than erroneous initial antecedents. If a theory contains errors at a layer higher than the input layer, INDiGENT has no way to correct this. INDiGENT can neither add nor remove nodes from the hidden layer and thus cannot truly explore all of the possible network topologies. As discussed above, however, when evaluated on classification accuracy, INDiGENT's feature selection performs as well as more extensive revision systems.

It would be reasonable to hypothesize that combining INDiGENT's ability to select inputs with a more general theory refinement strategy would give further improvements. To test this hypothesis, we implemented TNT-INDiGENT.

5. TNT-INDiGENT : Total Neural Topology Search

TNT-INDiGENT (Total Neural Topology INDiGENT) is a system which is able to modify the entire topology of input and hidden nodes[9]. The general course of the TNT-INDiGENT algorithm is very similar to that of INDiGENT. The significant differences come in the mutation and crossover operators (described below). Like INDiGENT, TNT-INDiGENT begins with an initial neural network. While the genotype used by INDiGENT was a binary string corresponding to possible input features, TNT-INDiGENT maintains a population whose members are entire neural networks. TNT-INDiGENT's mutation and crossover operators are thus performed on entire networks. TNT-INDiGENT mutates the initial network that it is given and adds these new networks to the original one in order to create the starting population of networks (ten by default). Each of these networks is evaluated, using N-fold cross validation as above. Two parents are then chosen from the population by the same selection method as INDiGENT. However, TNT-INDiGENT also keeps track of the number of times a particular network has been generated as a child. If this number of repetitions exceeds a set limit, one or both of the parents is chosen at random in an effort to encourage the algorithm to produce a diverse range of children. Having chosen two parents, the networks are crossed over to create a child. The child is evaluated by the fitness function. If its fitness exceeds that of the worst member of the population and if the topology of the child is not represented elsewhere in the population, the worst member of the population is replaced by the new child. Preventing duplicate networks is necessary for the maintenance of a diverse population. Because TNT-INDiGENT is fairly conservative in performing crossover (focusing on retaining the fundamental structure of the initial domain theory), it is not uncommon to generate the same children repeatedly.

5.1. *Mutation in* TNT-INDiGENT

TNT-INDiGENT stores and manipulates entire networks as its population. As a result, mutation in TNT-INDiGENT is more complicated than in INDiGENT. Because of this increase in complexity, mutation is also more directed towards the learning process. Mutation occurs in two distinct steps.

First the hidden nodes of the network are mutated. For each node in the hidden layer there is a chance equivalent to the mutation rate[10] that it will be mutated. Each node is mutated according to whether or not it represents a logical *and* or a logical *or*[11] and whether it has generated a greater number of false positives or false negatives. A node is considered to have generated a false positive when its error value is positive. Conversely a node generates a false negative when its error is negative. In practice, false positives or negatives are only accumulated if the error is greater than a threshold value. The manner in which the node is mutated is designed to reduce the number of errors made by the node.

If a node is a logical *and* with more false positives than negatives, a new antecedent node is added to the node and its bias is increased to maintain it as an *and* node. If an *and* node has more false negatives than positives, then two nodes are added. The node which is being mutated is made the antecedent of a newly created *or* node which inherits the mutated node's outgoing synapses. Another node is created to be an additional antecedent of the first created node.

An *or* node with too many false positives is modified similarly to a false negative *and* node. However, the consequent node which is created is an *and* node rather than an *or*. An *or* node with too many false positives may also be mutated by removal of the antecedent with the weakest connecting synapse. If an *or* node has more false negatives, an antecedent node is added to the mutated node. For further explanation, see Figure 1. This manner of mutation is very similar to that used by REGENT. REGENT, however is unable to subtract nodes from a neural network in mutation, as TNT-INDiGENT is able to.

The input nodes are mutated separately in a manner which is identical to the method used by INDiGENT. A binary string representing the inputs is generated from the network's input features, and this string is mutated with a slight bias for adding features. The mutated genotype is then translated back into the network's input nodes. The final step in mutation is to fully connect the network so that the nodes which are added gain connections to all nodes in the layer above and from all nodes in the layer below.

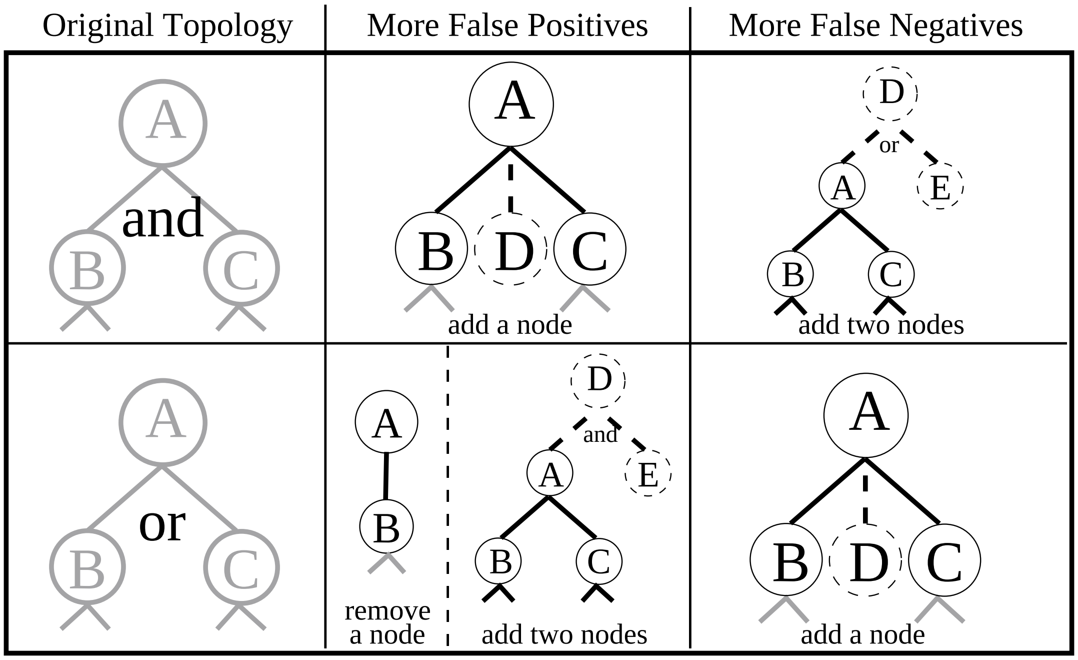

Figure 1. Mutation in TNT-INDiGENT.

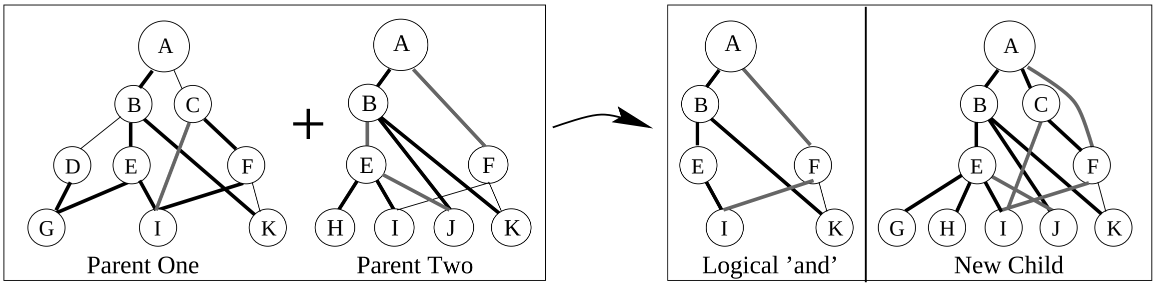

Figure 2. Crossover in TNT-INDiGENT

5.2. *Crossover in* TNT-INDiGENT

TNT-INDiGENT crosses two parent networks together to create a new child network. Because the networks encode logical information about the domain theory, it is important that the process of crossover not destroy this information. To achieve this, TNT-INDiGENT attempts to cross over entire logical rules. The first step in crossover is to copy the output nodes to the child.[12] Next TNT-INDiGENT searches for nodes which exist in both parent networks and copies them into the child network. (See Figure 2.) This process begins at the hidden layer directly below the outputs and proceeds downward. As a result, when a node is copied, any synapse which connects to a node already existent in the child network is copied as well. When a node is added into the network, the value of its synapses are the average of the values for the synapses in the two parent networks. The next step in crossover is to determine nodes which are present in only one parent network, but that are significant. A node is defined to be significant if it is strongly connected[13] to a node which has already been added to the child network. This process is repeated for the input nodes.

6. **Results of** TNT-INDiGENT

Two separate experiments were run with the TNT-INDiGENT algorithm. The first consisted of a ten-fold run of TNT-INDiGENT using the initial domain theory as the starting point. The second experiment was a ten-fold run of TNT-INDiGENT which used the voted feature sets from INDiGENT's runs as the starting network. The two separate experiments were run in order to better understand the contributions of both TNT-INDiGENT and INDiGENT. By running TNT-INDiGENT without INDiGENT features, the effect of TNT-INDiGENT on the initial domain theories can more accurately be gauged. Additionally, the results of TNT-INDiGENT can be compared more equivalently with REGENT . By running TNT-INDiGENT seeded with INDiGENT features and comparing the results with the unseeded runs, the effects of INDiGENT features upon a genetic theory revision system can be examined. Each of these experiments was run on the biologic domains described earlier. The accuracies that are reported are averaged over ten runs of 250 generations.

6.1. *Numerical Results*

Table 6 shows the accuracy of two separate runs of TNT-INDiGENT compared with several other knowledge-based neural network systems. The unseeded runs of TNT-INDiGENT started with the initial expert domain theory and features, while the seeded results represent the original expert domain theory in conjunction with the best voted features selected by INDiGENT.

Table 6. TNT-INDiGENT's accuracy compared to several other systems (†=unseeded, ⋆=seeded; standard deviations in parentheses).

System	Promoter	RBS	Splice Junctions
TNT-INDiGENT †	95.34 (0.22)	92.69% (0.14)	95.36% (0.44)
TNT-INDiGENT ⋆	95.75% (0.22)	93.43% (0.11)	96.09% (0.33)
REGENT	95.83%	92.17%	95.92%
KBANN	93.70%	91.05%	94.75%
REGAL (Anglano et al, 1998)	-	-	95.40%
G-NET (Anglano et al, 1998)	-	-	96.80%

The seeding of TNT-INDiGENT with INDiGENT features bears some discussion. While in the RBS domain the feature set used for seeding was the voted INDiGENT feature set, in both the promoters and splice junction domains the feature set used for seeding was the feature set from the ten separate INDiGENT runs which had produced the most accurate result. This difference occurred because, while the voted RBS features resulted in an accuracy that matched the average INDiGENT accuracy, the voted feature sets for both the promoters and splice junctions performed significantly worse than the average INDiGENT accuracy. There is no clear reason for this discrepancy in performance. The only conclusion that can be drawn is that the features that were unique to each INDiGENT feature set were crucial to the promoter and splice junction performances while those features that were important in the RBS domain were consistent throughout the feature sets. This may indicate that the RBS domain has a single "correct" feature set while the promoters and splice junction domains have several. Alternatively it may indicate that the RBS feature sets were closer to the single "correct" set while the others were at a greater distance on the error space and therefore necessarily disjoint.

7. Discussion of Results

Since INDiGENT alone had comparable performance to a number of existing theory revision algorithms, we had hypothesized that combining INDiGENT with TNT-INDiGENT would result in even greater accuracy. As can be seen in Table 6, that is not at all the case. The performance achieved by both TNT-INDiGENT and TNT-INDiGENT combined with INDiGENT is roughly equivalent to that of just INDiGENT.

As described earlier, INDiGENT has shown itself to be an effective and simple method for theory revision. INDiGENT attains a level of performance that is on par with other, more extensive theory revision systems. This is an appealing result for a number of reasons. Foremost, feature set revision is an understandable and explainable form of theory revision.

One criticism of feature set revision is that it simply adds enough features that there is a one-to-one correspondence created between the features and the data. In order to prevent this, the ratio of features to data points in all of the domains tested above is approximately 1:4. INDiGENT's resulting feature sets contain on average less than two thirds of the features. Additionally, INDiGENT removes some features from the initial domain theories, providing further evidence that its process of feature selection does not blindly add new inputs.

8. Related Work

Our thesis is that feature selection is a powerful mechanism for performing theory revision, on par with more exhaustive methods of revision. While the INDiGENT and TNT-INDiGENT algorithms are presented as a method for verifying that thesis, rather than as novel algorithms, in this section we describe other similar systems that integrate genetic algorithms and neural networks. We follow that with a discussion of systems that perform feature set selection and theory revision in general.

8.1. Revision of Expert Neural Networks

As discussed above, many have shown that knowledge bases represented as neural networks can be refined through the addition or removal of rules in the knowledge base. A number of different representations of human knowledge have been encoded in the form of neural networks, including deterministic finite automata (DFAs) (Omlin & Giles, 1996), (Maclin & Shavlik, 1993), certainty factor rule bases (Mahoney, 1996), pushdown automata (Das et al, 1992) and modified first order logic rule bases (Towell et al, 1990). Most of these systems translate the domain knowledge into a network topology and then refine the network's biases and weights through standard neural network techniques in order to "revise" the theory. The revision that occurs is essentially a weighting of the rules or concepts in the theory, such that certain rules are considered more accurate or valuable than others. Some systems refine the network's topology as well as the weights of its synapses.

The system that INDiGENT most closely resembles is REGENT. REGENT, an extension of KBANN, genetically refines KBANN topologies. REGENT operates by first generating the initial KBANN topology. It then randomly mutates the network's hidden nodes to create the initial population for its genetic algorithm. The population is evaluated and two parents are chosen for reproduction proportional to their fitness values. REGENT uses performance on a held-out data set as its fitness indicator. Similar to TNT-INDiGENT, REGENT's crossover operation attempts to cross over logical rules in the networks, rather than simply nodes. REGENT does not, however, refine the input nodes as INDiGENT has been designed to do.

The DAID (Towell & Shavlik, 1992) algorithm was an early attempt to refine KBANN networks. It is similar to INDiGENT in that it focuses upon refining and adding input antecedents to the knowledge base. The algorithm has significant differences, however. Rather than genetically searching for the most successful set of input nodes, DAID attempts to induce additional nodes into the domain theory from the given training examples. Rather than using standard back-propagation, the DAID algorithm operates under the (somewhat) substantiated assumption that most errors in a rule set occur in the bottom-most rules. The algorithm therefore recursively computes, through boolean arithmetic, the first-level hidden nodes whose boolean values are incorrect. At the same time, it induces correlations between unused data points and the given output. Using this information, it adds antecedents to the incorrect rule in an attempt to correct the error. This process is repeated for all of the training examples. The new rule set is then sent on through the standard KBANN network creation steps and evaluated. Unlike INDiGENT, DAID is unable to remove input features that exist in the initial expert domain theory. DAID created some small improvements in accuracy over KBANN.

FONN (Botta et al, 1997) extended KBANN to represent flat theories in first order logic. FONN translated theories in first order logic into theories expressed solely with AND and OR rules. These rules were then translated into neural networks using the same methods as KBANN.

Another method developed to modify KBANN topologies is the ADDEMUP algorithm (Opitz and Shavlik, 1996). Like REGENT, ADDEMUP starts by generating a population of topologies from a KBANN network. However, rather than evolving this population to create a single best network, ADDEMUP attempts to evolve an increasingly accurate population, or ensemble, of diverse networks. ADDEMUP's back-propagation rule was modified from the standard error measurement method in order to emphasize performance on examples previously classified erroneously by the ensemble. Likewise the fitness measure used in the genetic algorithm was modified to enhance diversity as well as accuracy. The output of ADDEMUP is the weighted average of the outputs of the individual networks in its ensemble. While ADDEMUP fails to develop a single best network, its ensembles show improvements in accuracy over REGENT and some improvement over INDiGENT.

RAPTURE (Mahoney, 1996) was designed to implement certainty or probability factor rule bases as neural networks. RAPTURE is also able to refine the topologies of these networks. However, the method that was devised for modifying the topology is significantly different from either TNT-INDiGENT or REGENT . RAPTURE modifies the topology of its hidden nodes through the UPSTART (Frean, 1990) algorithm, which adds hidden nodes to outputs in an attempt reduce the number of false positives and false negatives generated. RAPTURE also differs from REGENT and INDiGENT in that it is trained until it has correctly classified all of the training examples, rather than minimizing error on an unseen test set. Like INDiGENT, RAPTURE is able to add input nodes to its networks. However, the nodes are added using the information gain metric rather than using genetic selection as in INDiGENT. Like DAID, RAPTURE cannot remove input features mentioned in the initial domain theory. RAPTURE also does not fully connect between network layers, allowing less freedom for the network to modify the initial domain rules during training.

8.2. *Symbolic Theory Revision*

A number of other systems have been developed which take a much more directed approach to theory revision. EITHER (Ourston, 1991) was developed to revise domain theories in predicate logic form. The EITHER algorithm can modify theories by adding or deleting either rules or their antecedents. NEITHER (Baffes & Mooney, 1993) was developed as an extension to EITHER. It speeds up EITHER's algorithm and introduces the concept of M-of-N rules to the domain theory. NEITHER also improves upon EITHER by re-evaluating possible modifications to the domain theory after each modification it performs. NEITHER showed performance roughly equivalent to that of KBANN and an early version of RAPTURE. It also generated new domain theories that contained fewer rules than EITHER or KBANN.

Another approach worth noting is Numerical Term Refining or NTR. Developed as a corrolary to FONN, discussed above, NTR applied gradient descent refinement to the parameters of the numerical terms represented by the thresholding *Inside*(y, [a,b]) function.(Botta & Piolam, 1998) This expanded the types of first order theories that could be refined from the flat theories refined by FONN to general first order logic. It is also especially relevant to the work discussed here because one of its expressed advantages is that the refinement performed, while providing improvements in accuracy that matched or superseded FONN, maintains much more of the symbolic information present in the original theory allowing for easy interpetation of the refinements as well as the use of the refinements with other symbolic learning techniques. This focus upon maintaining symbolic information is also one of the advantages and focuses of the feature set revision performed by INDiGENT.

8.3. *Selection of Feature Subsets*

A number of researchers have shown that the selection of a feature set can dramatically improve a classification system's performance. Several earlier methods focus upon the selection of the set of relevant features *prior* to the application of a learning algorithm. Since the space of potential feature subsets is so large, most algorithms use some form of non-deterministic search.

DistAl (Yang & Honavar, 1998) is the system most directly related to INDiGENT. DistAl uses a genetic search to find optimum features for constructive neural networks. Constructive networks are built using a greedy search which adds a single hidden node at a time until the network has classified the entire training set. This method is fundamentally different from INDiGENT in two important ways. First, the construction of the neural network does not encode any expert information. Second, the greedy search approach does not create networks which have more than a single hidden layer. Both of these restrictions limit the ability of DistAl to capture accurate networks for large training sets. Additionally they severely curtail the interpretability of the constructed networks. DistAl showed the ability to improve the performance of its constructive neural network algorithm when compared to accuracies when all features were used.

Gabret (Vafaie & DeJong, 1998) also uses a genetic algorithm to search for optimal feature sets. Gabret was designed to select features to optimize the performance of the decision tree learning system C4.5. Gabret's binary representation of features and genetic algorithm are very similar to INDiGENT's. However at each generation, Gabret produces two offspring which together represent the entirety of the features in the two parents. Like DistAl, Gabret does not utilize any expert information. As a result, Gabret's initial feature sets are arbitrarily generated, which may ultimately hinder both the accuracy and speed of the algorithm. Significantly different from INDiGENT, Gabret has a module which focuses on feature construction, that is the combination of multiple existing features into a single new feature. Gabret is able to increase the accuracy of C4.5 at the task of locating eyes in images.

An exception to the general use of non-deterministic search in feature selection is FOCUS (Almuallim & Dietterich, 1991), which performs an exhaustive search of all feature subsets and finds the smallest subset of features which has the property that if any two data points in the feature subset agree, their classification agrees. FOCUS, however, is computationally expensive and unable to handle noisy data.

The RELIEF (Kira & Rendell, 1992) algorithm selects the set of "relevant" features and filters data for C4.5. RELIEF computes relevance by examining an arbitrary number of data points chosen at random and producing a weight which represents their summed square difference from the nearest positively and nearest negatively classified neighbors. These weights are summed for each feature. A feature whose weight is near zero is assumed to mean that the feature is equally close, on average, to both positive and negative classes. Such a feature is therefore *irrelevant*. A feature whose weight is positive indicates a connection to a class, and those features are judged to be *relevant*. A subset of these relevant features is chosen as those whose relevance eclipses a threshold value which can be set through inspection or automatically, proportional to the size of the feature set.

Asker and Maclin (1997) demonstrated that feature set selection was crucial to the success of a system which detected the presence of volcanos in exploratory images returned from Venus. Their method focuses upon the creation of a number of neural networks, each of which had a different subset of the total input features. These individual networks are then "bagged" (Breiman, 1996) to create the final classifier. Unlike INDiGENT or TNT-INDiGENT, however, the feature subsets are hand picked through a rigorous and time intensive process of human selection. Thus, though they demonstrate the importance of feature selection, they do not have a system that is truly autonomous. Devaney and Ram (1997) have also developed a system which autonomously searches the feature set space. Their algorithm differs from INDiGENT and TNT-INDiGENT in a number of ways. Most importantly, the search method that they use is backward and forward sequential selection. This method of search starts with either an empty (forward) or full (backward) feature set and adds the next best feature (forward) or removes the next worst feature (backward). This process continues until no more improvement is made. Sequential search suffers from the fact that it is a hill-climbing algorithm. As such it is unable to traverse down into a valley in the error space in order to ascend to a higher point elsewhere. Because genetic search is non-deterministic, INDiGENT and TNT-INDiGENT are often able to jump from one peak to another. Additionally, sequential search algorithms are unable to simultaneously consider radically different feature subsets and genetically combine them as INDiGENT and TNT-INDiGENT's genetic algorithm is able to do.

Recently there has been much work in the area of constructive induction. The concept of INDiGENT as a two-faceted system, both as a theory refinement system and as a feature selection system goes well with Michalski's definition of constructive induction as a two-phased search – one for the best representation space, and one for the best description in that space. (See, for example, Bloedorn (1998).)

9. Conclusions and Future Work

In this paper we have examined theory refinement as feature selection and as theory reformulation. We have shown that both feature selection and theory reformulation are effective in improving knowledge-based neural networks. We have also shown that, counter to intuition, feature selection alone performs as well as more extensive theory revision.

Future work includes applying both algorithms to additional domains. It also includes more careful inspection of the resulting domain theories, in order to better characterize the relative effects of the two forms of theory refinement.

Notes

1. The features that are chosen are those which have been used by the decision tree to correctly classify at least 5% of the data.
2. This choice is accomplished through a wheel of fortune selection process where each genotype receives a segment of the wheel whose size corresponds to the genotype's fitness.
3. Promoters are sequences which allow RNA polymerase to initiate and regulate the transcription of DNA into proteins.
4. ftp.cs.wisc.edu/machine-learning/shavlik-group/datasets
5. Ribosome binding sites are the locations at which the m-RNA binds to the ribosome to begin transcription into a protein.
6. Splice junctions are the points in a sequence of DNA which form the boundaries between areas of DNA which are retained or removed after splicing.
7. This data set comes from the U.C. Irvine dataset archive *ftp.ics.uci.edu*
8. Except for the credit domain which represent only 50 generations.
9. It is assumed that the designer of an expert theory has a clear idea of the correct categories for classification, thus modifying the output nodes of the network doesn't make sense.
10. The mutation rate is set manually by the user. In our experiments we used a mutation rate of one percent.

11. A node is considered to be an *and* if its bias is slightly less than the sum of all incoming positive weights. A node is an *or* if its bias is slightly greater than the sum of all incoming negative weights. If both of these are true the statement with the greater distance between the sum of incoming weights and the bias wins (See Opitz (1995) for more details).
12. As discussed earlier, the outputs are assumed to be constant.
13. A node is strongly connected if the weight of the connecting synapse is greater than a minimum weight set by the user.

References

Almuallim, H. and Dietterich, T. G. (1991). Learning with Many Irrelevant Features. *The Proceedings of the Ninth National Conference on Artificial Intelligence*, 547-552.

Anglano, C., Giordana, A., Lo Bello, G., and Saitta, L. (1998). An Experimental Evaluation of Coevolutive Concept Learning. *The Proceedings of the Fifteenth International Conference on Machine Learning.*

Asker, L. and Maclin, R. (1997). Feature engineering and classifier selection: A case study in venusian volcano detection. *The Proceedings of the Fourteenth International Conference on Machine Learning.*

Baffes, P. and Mooney, R. (1993). Extending theory revision to m-of-n rules. *Informatica*, 17.

Bloedorn, E. and Michalski, R. (1998). Data-driven constructive induction. *IEEE Trans. on Intelligent Systems*, 13(2):30–37.

Botta, M., Giordana, A., and Piolam R. (1997). FONN: Combining first order logic with connectionist learning. *The Proceedings of the Fourteenth International Conference on Machine Learning.*

Botta, M. and Piolam R. (1998). Refining Numerical Terms in Structured FOL Theories. *The Proceedings of the Fourth International Workshop on Multistrategy Learning.*

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Das, S, Giles, C.L., and Sun, G. (1992). Using prior knowledge in an nnpda to learn context-free languages. *Advances in Neural Information Processing Systems*, volume 5.

Devaney, M. and Ram, A. (1997). Efficient feature selection in conceptual clustering. *The Proceedings of the Fourteenth International Conference on Machine Learning.*

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2.

Kira, K. and Rendell, L. (1992). The feature selection problem: Traditional methods and a new algorithm. *The Proceedings of the Tenth National Conference on Artificial Intelligence.*

Maclin, R. and Shavlik, J. (1993). Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding. *Machine Learning*, 11:195–215.

Mahoney, J. (1996). *Combining Symbolic and Connectionist Learning Methods to Refine Certaintty-Factor Rule-Bases.* PhD thesis, University of Texas, Austin.

Omlin, C. and Giles, C.L. (1996). Rule revision with recurrent neural networks. *IEEE Trans. on Knowledge and Data Engineering*, 8(1):183.

Opitz, D. W. (1995). *An Anytime Approach to Connectionist Theory Refinement: Refining the Topologies of Knowledge-Based Neural Networks.* PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison.

Opitz, D. W. and Shavlik, J. W. (1993). Heuristically expanding knowledge-based neural networks. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1360–1365.

Opitz, D. W. and Shavlik, J. W. (1996). Generating accurate and diverse members of a neural-network ensemble. *Advances in Neural Information Processing Systems*, volume 8, pages 535–543.

Ourston, D. (1991). *Using Explanation Based and Empirical Methods on Theory Revision.* PhD thesis, University of Texas, Austin.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81.

Towell, G. and Shavlik, J. W. (1992). Using symbolic learning to improve knowledge-based neural networks. *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 177–182.

Towell, G., Shavlik, J., and Noordewier, M. (1990). Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866.

Vafaie, H. and DeJong, K. (1998). Feature space transformation using genetic algorithms. *IEEE Trans. on Intelligent Systems*, 13(2):57–65.

Yang, J. and Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Trans. on Intelligent Systems*, 13(2):44–49.