# Theory refinement through knowledge-based feature set selection

**Brendan D. Burns and Andrea Pohoreckyj Danyluk**
Department of Computer Science
Bronfman Science Center
Williams College
Williamstown, MA 01267
andrea@cs.williams.edu

**Abstract**

Classification systems depend upon having the best (i.e., most relevant) set of input features from which a classification decision can then be made. This is true both for the classifiers themselves, and for the inductive learners that might be used to build them. This paper describes INDiGENT (Improving kNown Domains with Genetically Engineered feaTures). INDiGENT utilizes genetic search to perform knowledge-based feature selection. It assumes that it will be provided with a knowlege-based neural network. It then performs a genetic search for better network topologies, focusing entirely on the incorporation (or deletion) of input features. INDiGENT can be viewed in one of two ways: (1) as a theory refinement system, or (2) as a feature selection system, guided by expert-provided domain knowledge. It can be used either alone or in conjunction with more extensive theory refinement systems.

## Introduction

Classification systems depend upon having the best (i.e., most relevant) set of input features from which a classification decision can then be made. This is true both for the classifiers themselves, and for the inductive learners that might be used to build them. This paper describes INDiGENT (Improving kNown Domains with Genetically Engineered feaTures). INDiGENT utilizes genetic search to perform knowledge-based feature selection.

Expert classification systems have proven themselves effective decision makers for many types of problems. However, the accuracy of such systems is highly dependent upon the accuracy of the 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. (Russell & Norvig 1995) To escape this dependency, many machine learning systems have been developed to automatically refine and correct an expert's domain theory or to learn a classification theory from data alone, completely bypassing any theory provided by an expert. While theory revision systems can be applied to the expert theories, they often concentrate on the reformulation of the knowledge provided rather than on the reformulation of input features.

Even in the case of inductive learning of a classifier from data, the system is often provided with a fairly small set of features that some expert has determined to be "good" for a particular task. That set may, however, be suboptimal. While this is often the case, the idea of considering all potential features may simply be too unwieldy.

INDiGENT is a system that has been developed to perform feature selection for classification tasks. It assumes that it will be provided with a knowlege-based neural network. It then performs a genetic search for better network topologies, focusing entirely on the incorporation (or deletion) of input features. INDiGENT can be viewed in one of two ways: (1) as a theory refinement system, or (2) as a feature selection system, guided by expert-provided domain knowledge. The selected features can then be input to a standard inductive learning system or to a more extensive theory refinement system.

In the next section we will discuss background and related work. We follow this with a description of the INDiGENT algorithm. The next section discusses the results of applying INDiGENT to domains from molecular biology. 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. We also discuss the results of applying standard inductive learning algorithms to the features selected by INDiGENT. We follow this with a description of TNT-INDiGENT and show how feature set selection can impact a more extensive theory revision system.

## Background and Related Work

A number of researchers have shown that the selection of a feature set can dramatically improve a classification system's performance[1]. 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.

The system which is most directly related to INDiGENT is DistAl (Yang & Honavar 1998). 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. Secondly, 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 demonstrated (Asker & Maclin 1997) 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 (Devaney & 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 & Michalski 1998).)

## Knowledge-based feature selection

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

[1]The selection of a feature set as performed by INDiGENT and the following algorithms differs from the implicit feature selection performed by inductive learning systems since the goal of INDiGENT's feature selection is to eliminate noisy data in order to improve a later machine learning algorithm's performance.

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, Shavlik, & Noordewier 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, Shavlik, & Noordewier 1990). INDiGENT uses the KBANN (Towell, Shavlik, & Noordewier 1990) method for encoding domain theories. There has also been work done on genetically refining neural network topologies. The REGENT (Opitz 1995) system is 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 Quinlan's (Quinlan 1986) information gain metric) 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 doesn't explore the entire space of possible input feature sets.

## INDiGENT: Finding an Accurate Input Feature Subset

INDiGENT focuses upon 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 all of the possible subsets is currently 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 is full of local maxima.

The first step in the INDiGENT algorithm is the generation of the 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[2]. 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 ten.

After the initial population of networks has been created from the domain theory's genotype, the population of networks is evaluated using N-fold cross validation.[3]

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[4]. The parent genotypes are then crossed over to create a new (child) genotype.

Once a child genotype has been created, it is mutated by the same mutation operator that was applied to create the initial population. The fitness function is then applied to the child genotype. 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. When the specified number of generations has been reached, the best network found is presented as INDiGENT's result. The operation of INDiGENT is summarized in Figure 1.

### 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, most genetic algorithms operate by splicing two fractions of the parent genotypes together to form the child's genotype. In reality both of these

[2]The features that are chosen are those which have been used by the decision tree to correctly classify at least 5% of the data.

[3]N-fold cross validation is a method of testing a program's ability to learn rather than memorize a data set. First the data is randomly divided into N sets. The following algorithm is performed N times: A single set is held out and the network is trained on the remaining N-1. The network is tested on the held out (and therefore unseen) data set. The accuracy of the network is reported as the average accuracy of all N folds.

[4]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.

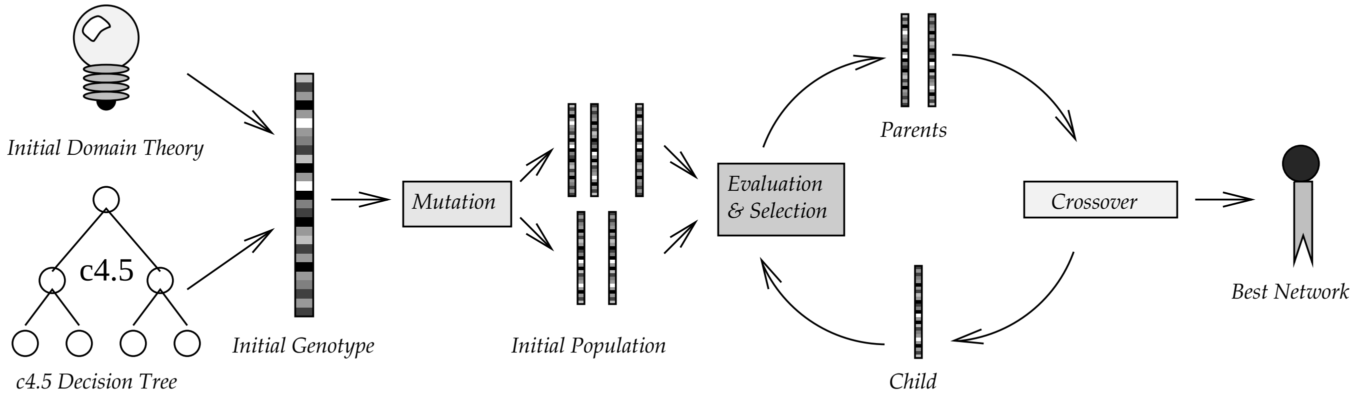


Figure 1: The INDiGENT algorithm.

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.

### 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 reasoning for this is 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 backpropagation 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.

## Experiments

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

### Introduction to the Domains

The three 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, the 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 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[5]. The initial domain theory was obtained, along with the dataset, from the machine learning archive at the University of Wisconsin[6]. The domain contains 234 positive classification examples, each from a different source, and 4,921 negative classification 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 the KBANN and REGENT experiments at the University of Wisconsin, the benchmark results against which INDiGENT and TNT-INDiGENT are compared. The dataset contains 234 positive examples and 702 negative examples (1:3 ratio) randomly dis-

[5] Promoters are sequences which allow RNA polymerase to initiate and regulate the transcription of DNA into proteins.

[6] ftp.cs.wisc.edu/machine-learning/shavlik-group/datasets

tributed throughout all ten folds.

**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[7]. 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 used were the same as the folds used in the KBANN and REGENT experiments.

**The Splice Junction Domain** The third domain that was used is the identification of splice junctions[8]. 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 twelve hundred strands of sixty nucleotides. There are four hundred positive examples and eight hundred negative examples. Five random folds were used which were created locally. A subset of the total dataset of 3,190 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.

## Measuring Performance

The goal of INDiGENT is to create a neural network which, when given a string of input data (in this case DNA), returns the appropriate classification for the DNA. The measure of accuracy is the percentage of examples for which the classifier judges correctly. 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, the numbers reported for REGENT represent runs of 500 generations.

## Experimental Setup

For the experiment, ten runs of INDiGENT for 250 generations were conducted in each data domain. During training, the training data was shuffled prior to each training iteration in order to encourage generalization in the networks. The best network at each generation was reported. The best network after the final generation was used as the result of each run, and its accuracy was evaluated.

Because the accuracy of knowledge-based neural networks are slightly variable, the accuracy reported below for each of the ten best networks actually represent the average of a number of evaluations. The accuracies for the ten networks were averaged together to form the final accuracy of INDiGENT for the domain.

## Numerical Results

Before discussing the results of the experiments described above, it it important to note some 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 hundred and fifty 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 five hundred for REGENT's evaluation seems arbitrary. The reason that INDiGENT was only run for two hundred and fifty 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 graphs show primarily that the goal of performance increasing in proportion to computation is not

[7]Ribosome binding sites are the locations at which the m-RNA binds to the ribosome to begin transcription into a protein.

[8]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.

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 two hundred 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.

Figure 2 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. Figure 2 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 . Figure 3 shows some statistical information about INDiGENT's feature sets.

| | Promoters | RBS | Splice Junctions |
|---|---|---|---|
| % Increase | 26.3% | 68.4% | 33.3% |
| % of Total # | 71.9% | 65.3% | 46.7% |

Figure 3: Differences between initial and INDiGENT's feature sets.

## 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 undertstanding 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.

Additionally, feature set revision is important because it can guide the reformulation of an expert theory. If a feature set is selected which improves the performance of the domain theory, it is likely that a more extensive theory revision influenced by this selection of features will lead to greater improvements. Such theory revision is addressed by the next generation INDiGENT system, TNT-INDiGENT, which is discussed below.

INDiGENT's results are interesting in the effect the learned feature sets have on inductive learning systems. As shown in Figure 4, the voted INDiGENT feature set improved the performance of c4.5 and showed slight improvements over both Ripper and RL in the Promoter domain. Perhaps more importantly, as shown in Figure 5, the classifiers learned are smaller (and, thus, in a sense "simpler") when given as input only the features selected by INDiGENT.

## 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 is limited in its 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. It is reasonable to imagine that combining INDiGENT's ability to select inputs with a more general theory refinement strategy would give further improvements.

We have addressed this in the implementation of TNT-INDiGENT. We find that INDiGENT can be used in conjunction with TNT-INDiGENT and that indeed feature selection adds to the performance of the more extensive theory revision system.

## 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

[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.

| 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% |

Figure 2: INDiGENT's accuracy compared to several other systems (deviation in parentheses)

| | 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% |

Figure 4: INDiGENT feature set accuracy used with several other systems

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, again using N-fold cross verification in order to measure generalization rather than memorization. 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.[10] As with INDiGENT, this process continues for a set number of generations.

## 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.

[10]Preventing duplicate networks is necessary for the maintenance of a diverse population.

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[11] that it will be mutated. Each node is mutated according to whether or not it represents a logical *and* or a logical *or*[12] 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 please see Figure 6.

[11]The mutation rate is set manually by the user. TNT-INDiGENT used a mutation rate of one percent.

[12]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).

| | Classifier size (RBS) | | Classifier size (Promoters) | |
|---|---|---|---|---|
| System | INDiGENT features | all features | INDiGENT features | all features |
| c4.5 | 126.5 (pruned) | 133 (pruned) | 63 (pruned) | 63 (pruned) |
| RL5.6 | 27.1 | 29.8 | 25.8 | 26.4 |
| Ripper | 9.4 | 13.1 | 7.8 | 9.8 |

Figure 5: Comparison of sizes of learned classifiers

This manner of mutation is very similar to that used by REGENT (Opitz 1995). 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.

### 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.[13] Next TNT-INDiGENT searches for nodes which exist in both parent networks and copies them into the child network. (See Figure 7.) 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[14] to a node which has been already been added to the child network. This process is repeated for the input nodes.

## Results of TNT-INDiGENT

Two separate experiments were run with the TNT-INDiGENT algorithm. The first consisted of a ten-fold cross validated run of TNT-INDiGENT using the initial domain theory as the starting point. The second experiment was a ten-fold validated 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 were run on the three domains described earlier. The accuracies that are reported are averaged in the same way as INDiGENT's final accuracies.

### Numerical Results

Figure 8 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.

## 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.

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 is approximately 1:4. Additionally, INDiGENT added an average of only 25% more features to the feature sets. INDiGENT's resulting feature sets contain on average less than two thirds of the features. Because of this it is clear that there can not possibly be a one to one correspondence between features and data points. 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.

[13] As discussed earlier, the outputs are assumed to be constant.

[14] A node is strongly connected if the weight of the connecting synapse is greater than a minimum weight set by the user.

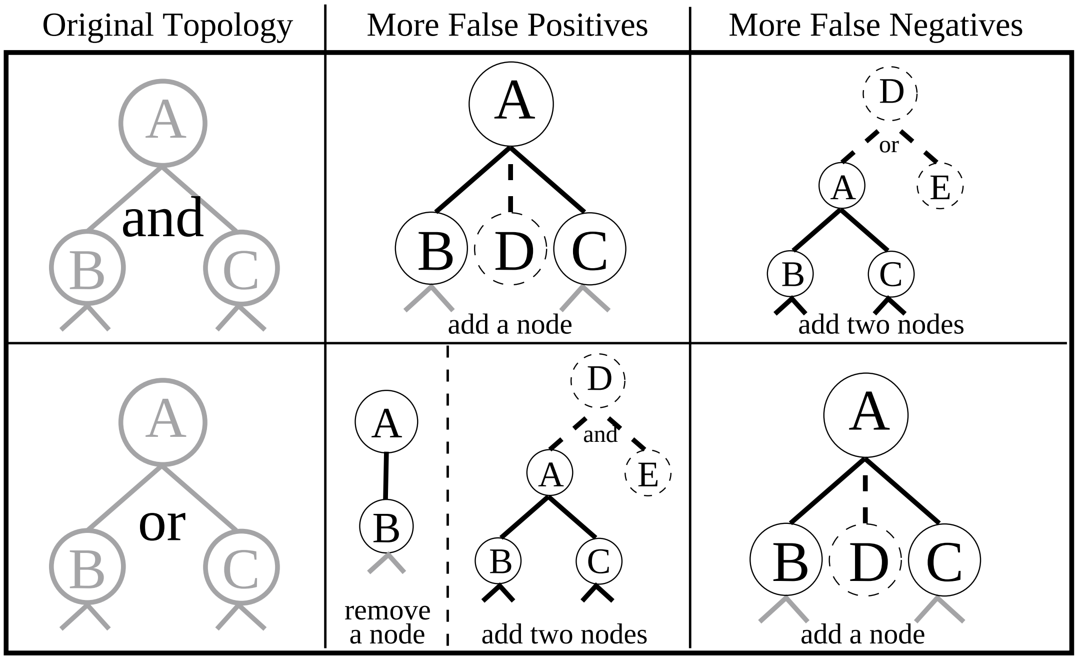


Figure 6: Mutation in TNT-INDiGENT

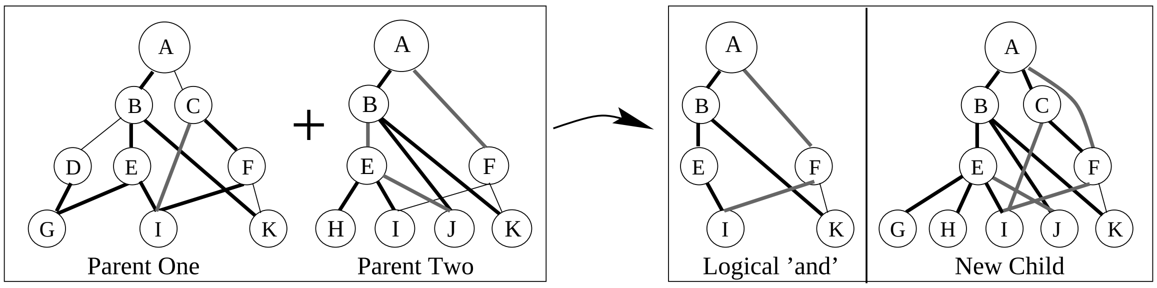


Figure 7: Crossover in TNT-INDiGENT

| 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% |

Figure 8: TNT-INDiGENT's accuracy compared to several other systems (†=unseeded, ⋆=seeded, deviations in parentheses).

Additionally, feature set revision is important because it can guide the reformulation of an expert theory. If a feature set is selected which improves the performance of the domain theory, it is likely that a more extensive theory revision influenced by this selection of features will lead to greater improvements as shown by the results of TNT-INDiGENT.

The results of TNT-INDiGENT are important for two reasons. First, the results of the runs seeded with INDiGENT's features show that INDiGENT is compatible with a theory revision system which modifies neural topologies. The resulting combination has been shown to produce domain theories which can be equally or more accurate than previously existing machine learning methods.

The seeding of TNT-INDiGENT with INDiGENT features however 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 which matched the accuracy of 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 which where unique to each INDiGENT feature set were crucial to the promoter and splice junction performances while those features which 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. Regardless of the reason, it seems that performance is clearly dependent on the initially seeded feature set.

TNT-INDiGENT's results are interesting at another level as well. While one important aspect of theory revision is ultimate accuracy, another, as discussed above, is the simplicity of the theory. TNT-INDiGENT's results for both the seeded and unseeded runs are significant because the modifications that it makes to the domain theories are much smaller than previous systems (see Figure 9).

| System | RBS | Promoters | Splice |
|---|---|---|---|
| TNT-INDiGENT † | 3.9 | 4.0 | 3.2 |
| TNT-INDiGENT ⋆ | 3.9 | 6.0 | 1.9 |
| TopGen | 24.1 | 9.2 | 7.4 |
| REGENT | 52.1 | 43.9 | 11.4 |

Figure 9: TNT-INDiGENT's nodes added compared to several other systems (†=unseeded, ⋆=seeded).

TNT-INDiGENT shows performance equal to or better than REGENT , with a dramatic decrease in the number of added nodes. As discussed above, the simpler the changes to an expert system, the more these changes are understandable. If the changes are understandable, then the system is more likely to be trusted (and therefore used). Additionally, simple theories result in simple implementations which are quicker to train and use than more complicated models. Curiously, the simplification of the changes does not seem to result from the use of INDiGENT feature sets. This is interesting because, when used with other machine learning systems, INDiGENT features appear to simplify the resulting theories.

## Conclusions and Future Work

In this paper we have examined INDiGENT, a system for the genetic selection of optimum feature sets for knowledge-based neural networks. We have also explored the application of these feature sets to other machine learning systems. INDiGENT has shown the ability to improve the accuracy of an initial domain theory. Additionally, we have shown that the feature sets selected may contain information which can be generalized to other machine learning systems. Most importantly we have shown that feature set selection is an important way to increase a machine learning system's accuracy without obfuscating its clarity.

Future work includes applying INDiGENT to a number of other domains in order to further examine its capabilities. Finally, it would be useful to construct an inductive rule learning system which can take greater advantage of the information provided by INDiGENT's selection of feature sets than TNT-INDiGENT currently does.

## References


Almuallim, H., and Dietterich, T. G. 1991. Learning with many irrelevant features. In *The Proceedings of the Ninth National Conference on Artificial Intelligence*, 547–552.

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

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

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

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

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

Maclin, R., and Shavlik, J. W. 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.* Ph.D. Dissertation, University of Texas, Austin.

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

Opitz, D. W. 1995. *An Anytime Approach to Connectionist Theory Refinement: Refining the Topologies of Knowledge-Based Neural Networks.* Ph.D. Dissertation, Department of Computer Sciences, University of Wisconsin-Madison. (Also appears as UW Technical Report.

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

Russell, S., and Norvig, P. 1995. *Artifical Intelligence: a Modern Approach.* Prentice Hall.

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

Towell, G. G.; Shavlik, J. W.; and Noordewier, M. O. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 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.