MACHINE LEARNING TECHNIQUES FOR THE EVALUATION OF EXTERNAL SKELETAL FIXATION STRUCTURES

by

Ning Suo

(Under the direction of Khaled Rasheed)

Abstract

In this thesis we compare several machine learning techniques for evaluating external skeletal fixation proposals. We experimented in the context of dog bone fractures but the potential applications are numerous. Decision trees tend to give both binary and multiple-class predictions quickly and accurately. The classifier system method does worse due to the small size of the data set and missing values. The use of Artificial Neural Networks is promising, although it takes considerable time in training. A Genetic Algorithm is also employed to find the best parameters of the Neural Network. Experimental results for the different methods are presented and compared.

Index words: External Skeletal Fixation, Decision Tree, Classifier System, Artificial Neural Network, Genetic Algorithm.

MACHINE LEARNING TECHNIQUES FOR THE EVALUATION OF EXTERNAL SKELETAL FIXATION STRUCTURES

by

Ning Suo

B.S., NanJing Forestry University, 1996

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2003

MACHINE LEARNING TECHNIQUES FOR THE EVALUATION OF EXTERNAL SKELETAL FIXATION STRUCTURES

by

Ning Suo

Acknowledgments

I would like to thank Dr. Khaled Rasheed, who got me interested in this project and pointed out the right directions for me. His invaluable suggestions and friendly encouragements were the source of my inspirations. I would also like to thank Dr.Walter D. Potter, my advisor and friend who guided and cultured me through the past two years. I thank Dr. Ron McClendon for his invaluable advice in my thesis work and sitting on my thesis committee. Dr. Dennis N. Aron deserves special thanks for preparing all the data and instructing me while I was working on this project. Also, I thank Dr. Michael Covington for supporting and helping me in my thesis development.

I am grateful to my beloved parents, brother and sister for their continuous encouragement, without their support, I would not have gone thus far. I also thank all my friends for their selfless help.

Finally, special thanks goes to my wife and best friend–Yan Ding, for her encouragement and love during my graduate study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Accidents can happen to anybody including your pets. Bone fractures that occur in dogs need special attention and treatment. Bone fracture repair in dogs is very critical for the health of the pet when an unexpected accident occurs. There are various methods of repairing fractures from simple confinement and rest to internal mechanical fixation with bone grafts. However, animals are not as cooperative as humans in following instructions. Therefore, veterinarians usually use external skeletal fixation devices (also called fixators) that can be attached to the bone in order to secure it during the healing process. External Skeletal Fixation (ESF) is a technique that is frequently used to treat patient with orthopedic injuries. An simple ESF consists of three elements:

- Fasteners. These pins transfix bone segments and act as handlers to those bone segments.
- Connectors. Used to connect rods or acrylic columns.
- Linkage devices. Attach the fasteners to the connectors.

Many different configurations of ESF with different level of stability (type I, II and III) can be obtained by utilizing these three basic components. For more information, refer to [7]. A typical type Ib fixator is shown in Figure 1. It contains two frames with two planes, which secure the bone fracture(s), yet allow the animal to move around moderately.

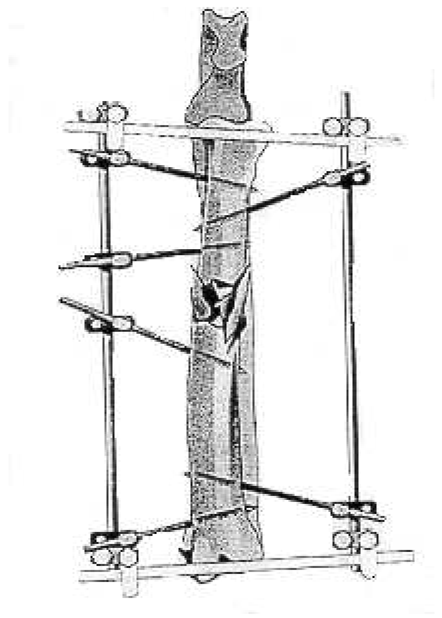


Figure 1.1: An external skeletal fixation device–Type Ib fixator

Researchers and veterinarians at the Veterinary School of The University of Georgia have been collecting and studying the data of their dog patients in order to learn which treatment is suitable for which dog. Not all dog patients can be treated in the same way. Veterinarians have to consider biological, mechanical and clinical information in order to provide the best treatment. Therefore, the researchers have to consider different treatments for each dog. An efficient evaluation (or better yet recommendation) system would be a useful tool for them in several aspects. It can be used in teaching by allowing students to test their proposed treatment plans through this system before going through expensive experiments. It can also be used to demonstrate the bone fracture(s) and different treatment proposals to the owner of the patient. Since the owners usually understand their pets better, having them involved would greatly help veterinarians make a final decision about treatment. The system can also be used to assist veterinarians in building databases, selecting fixators and recommending treatments. Most importantly, through testing this system on dog external skeletal fixation, we could optimize this system, and hopefully it could be used to help human patients and their doctors in selecting a treatment plan to some extent.

There are many factors that affect the selection of treatment. Many institutes have their own rules for selecting the effective parameters, but there are no existing standards in the public domain. Based on operational experience, professors in the veterinary school of the University of Georgia proposed a standard for recording the patient condition, which contains two parts with five categories. The first part records patient characteristics like age, history, size and owner's responsibility level. The second part refers to the proposed treatment itself. Table 1.1 shows an example of the score system. In the table, the biological parameters describe the general situation about a patient (a dog); the clinical parameters denote the responsibility of the owner; the mechanical parameters describe the mechanical situation of the fracture; and the mechanical treatments and the biological treatments describe the treatment plan proposed by the veterinarian.

Following the standard, a profile for each patient that describes its physical situation and treatment proposal is constructed. For each patient, 5 different treatment proposals are provided for training purposes; each treatment gets a different assessment. In order to make this problem suitable for computer program handling, a scoring standard is proposed as well. A score is assigned for each parameter that refers to the scoring standard, and a final balance score concludes the appropriateness of each treatment. The final balance score is the training target and ranges from 1 to 10. It should be noted that every training example has 33 features corresponding to the patient description and the fixator description, plus one target attribute, which is the final score that measures how suitable the proposed fixator is for this patient.

Table 1.1: An example of the Scoring System.

BIOLOGICAL PARAMETER		**MECHANICAL TREATMENT**	
age	3	KE vs. SK	4
history	2	number of pins	6
pretreatment	0	type of pins	8
limb appearance	2	insertion techniques	5
fracture pattern	4	load share	6
displacement	5	grade of pin	7
palpation stability	2	number of frames	4
metabolic abnormalities	3	articulation	3
open fracture	4	implement vs. bone	4
CLINICAL PARAMETER		supplemental fixation	4
patient	3	pin location	6
client	4	**BIOLOGICAL TREATMENT**	
MECHANICAL PARAMETER		open vs closed	6
size	3	duration of surgery	3
load of share	4	tissue observation	6
palption stability	5	amount of implement	4
abnomal limbs	4	graft	3
bone	5	amount of HBS	5

1.1 Related Works

Previously, Potter [1] and Dale [3] designed an expert system called "Bone-FixES" to analyze and evaluate treatment plans. An expert system is a computer program that utilizes human experts' experience to tackle real-world problems. It is suitable for domains with complex symbolic representation. Typically, there are three major components: knowledge base, inference engine and user interface. In the "Bone-FixES" program, the knowledge base contains the domain specific and problem-solving knowledge, i.e. facts and rules. The inference engine controls the flow of the program; it adopts a backward chaining structure, which considers all the possible solutions, then eliminates the weaker ones. Through a user interface, users can interact with the program by answering questions.

One feature of this approach is that a domain expert's knowledge is captured by the system. Since the knowledge base is constructed ahead of the prediction phase, there is no training process necessary. The system can be put to use very quickly. However, in this problem, like in many real life problems, it is easier for the domain experts (veterinarians) to describe different patient-treatment examples and their relative merit, rather than directly describe rules. They prefer inductive learning to expert system learning.

The expert system approach is the first attempt of applying Artificial Intelligence techniques in this domain. Thereafter, another technique was tested on this problem, which is decision tree learning.

Two students did a course project on this problem. In each of their experiments, four patients with five treatments each were used for training, and another patient for testing. They concluded that decision trees tend to predict better if the target is a binary decision, i.e. good fixator or bad fixator. They also mentioned that insufficient data (5 patients in total, with 5 treatments proposed for each patient) impacts the result of multiple-class prediction.

1.2 Objectives of this study

This research is the first step in a project, which includes several parts: developing an evaluation system to assess the future treatment proposal; developing a recommendation system that can be used to provide a treatment proposal for a new patient; implementing an image view tool that can be used to manipulate the bone fracture picture to help veterinarians suggest a cure plan. Among those, the evaluation system is the core of this research.

The goal of this study was to search for an optimal model building method that can serve as an evaluation system. More specifically, the objectives were:

1. Propose appropriate methods by analyzing the data set provided by the researchers from the veterinary school of UGA.

2. Implement proposed methods.

3. Evaluate these methods on the data set and evaluate each method's performance.

We continued this project by considering several different approaches. With the help of researchers in the veterinary school, more data sets along with a revised recording standard were constructed.

The remainder of this thesis is organized as follows: chapter 2 presents several learning methods that we used. In chapter 3, we explain the implementation details. In chapter 4, we describe the experiments and results. Following that, we conclude the thesis with a discussion of the results and future directions.

Chapter 2

The machine learning methods used

We tested four methods in our experiments. One is decision trees, another is classifier systems, the third one is artificial neural networks, the last one is to use a genetic algorithm for finding the design parameters of an artificial neural network. Before we go into details, we have to describe the data. There are 12 patients in total; each patient has 5 treatments. For each patient-treatment combo, there are 33 parameters that contribute to the final score as mentioned in chapter one.

In preparing data for model building, one obvious way is to take all 33 patient-treatment parameters as variables, with the final score being the target. This is the so-called *full parameters approach.* An alternative is to cluster the related parameters and treat them as a single variable. We name this the *reduced parameters approach.* The reason of introducing the reduced parameter concept is that we only have 60 observations in total, with each observation containing 33 parameters, which means the model we are building is going to search in a 33-dimensional space in order to find an optimal solution. It is a very expensive process and highly unlikely to find the optimal solution. Hence, we need to reduce the dimensionality to help the model converge better. In order to accomplish this, we chose to transform those variables using some mathematical formula. We take the average of the parameters in our experiment, which shrinks the parameters into four key factors. The validity of both approaches is checked in the experiments. We then describe each method in detail in the remainder of this section.

2.1 Decision Trees

Decision tree learning (Mitchell [11]) is a widely used and practical machine learning technique for inductive learning. A decision tree can be easily converted to a set of if-then rules, which are human readable. The decision tree classifier is constructed by studying sets of positive and negative examples. It then measures the information gain by choosing different attributes for constructing the tree. The formula for calculating the information gain is given below:

$Gain(S, A) \equiv \Sigma_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$

where:

S is the set of data samples

$Entropy(S) = \Sigma_{i=1}^{c} - p_i \log_2^{p_i}$

Values(A) is the set of all possible values for attribute A

S_v is the subset of S for which attribute A has value v

The best attribute at each level is considered as the pivot value, and based on that pivot value data are partitioned into sub-trees. It is a heuristic method since it always uses the attribute that leads to maximizing the information gain, plus it is very fast.

A tree structure is something like the one in figure 2.1. And the corresponding expression is:

$(Age \leq 3 \wedge (Surgical\ time) \leq 2)$

$\vee (Age \leq 1)$

$\vee (Age \leq 4 \wedge Wound = Closed)$

In plain English, this means, if the age of the patient (dog) is less than 3 and the surgical time is less than 2 or if the age of the patient is less than 1 or if the age is less than 4 and the wound is closed, then we consider the patient easy to cure.

Otherwise, it is not. Clearly, you can see how concise yet powerful the decision tree is.

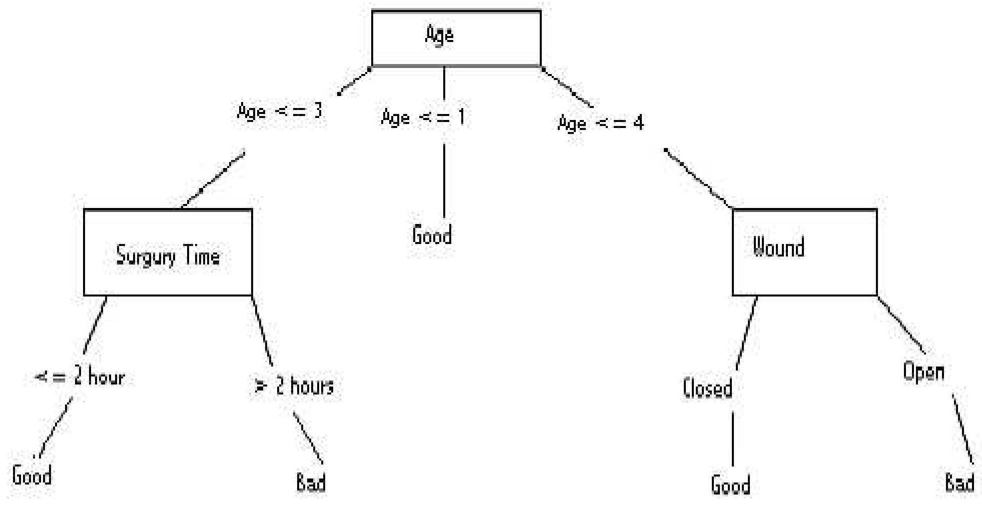


Figure 2.1: A typical decision tree structure

2.2 Classifier System

A classifier system is another machine learning technique that uses evolutionary algorithms such as a Genetic Algorithm as an exploration engine. Essentially, a classifier system contains 5 parts: detectors, a classifier list, a genetic algorithm, a credit assignment scheme and effectors. A classifier system is shown in figure 2.2.

Information flows from the environment through the detectors into the classifier system, once this finite length input matches one or more classifiers maintained in the classifier list, the corresponding action of these classifiers are posted to the message list. A classifier is made up of two parts, a condition part and message part. The condition is a simple pattern recognition device where a wild card character (#, means don't care) is added to the underlying alphabet. A message is simply a finite-length string over some finite alphabet, usually, a binary string. From the

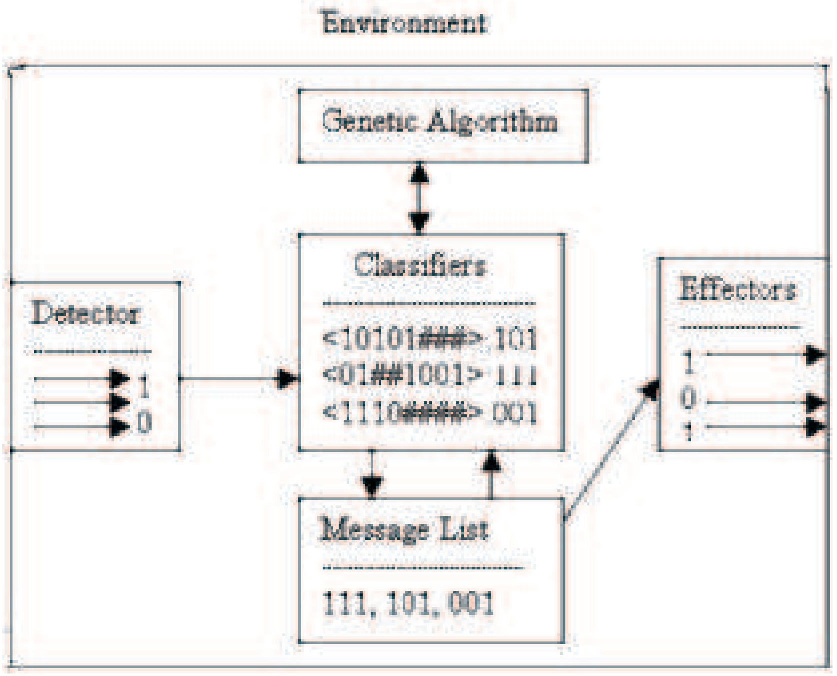


Figure 2.2: A simple classifier system

message list, the system can either select the best action or all of the actions act to the environment through effectors. Next, some credit value is returned from the environment, and the system will distribute the credit to each classifier that fires to the environment according to some criteria, called the credit assignment scheme. In this way, the classifier system learns and adjusts its action to the environment. The genetic algorithm acts as the “quality improvement” rule search. It can be triggered either at certain intervals or once the whole population fitness is below a certain threshold. Through genetic operators we hope that better rules will be generated, which corresponds to the rule generalization. We will explain more about Genetic Algorithms in section 2.4.

2.3 Backpropagation Neural Network

Frequently, we have to make decisions in our real life. If the relationship between each individual factor and the final decision is clear, our life would be much simpler, but most of the time it is not. Nevertheless, our brain can handle these situations surprisingly well. An artificial neural network ([9], [10]) is a computer program that can be thought of as a device that mimics the human neural system, which has considerable strength in handling complex real world problems. We applied Artificial Neural Networks to this problem as well. Instead of explicitly telling the output score corresponding to a certain combination of the input variables, we leave it to the program to figure out what the output could be by training the neural network using existing examples. An example of backpropgation neural network is shown below.

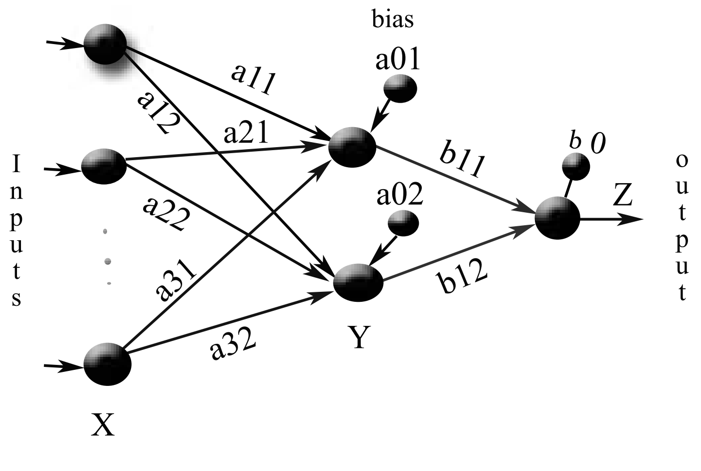


Figure 2.3: A simple artificial neural network

In the backpropgation neural network, The number of features determines the number of input nodes. The number of output nodes can be either one node with several partitions with each partition corresponding to one target or multiple nodes.

The number of hidden nodes varies, it can be set upon experiment. Corresponding to each interconnection, there is a weight value that is adjusted in the training process, which is used to denote the importance of the node in the previous layer. In the training process, the network sums up all the incoming values multiplied by the corresponding weights along with the bias weights for each node. The total is then mapped through a logistic function (usually a sigmoid function), then the output is sent to the nodes in the next layer(Fig. 2.3). The formula for updating the weights is as follows:

$\Delta a_{ij}^{m} = \epsilon * \frac{\partial E}{\partial a} + \mu * \Delta a_{ij}^{m-1}$

where:

m is the epoch number.

Δa_{ij}^{m} is the weight change in this epoch.

ϵ is the learning rate, which determines how large the change is going to be.

$\frac{\partial E}{\partial a}$ is the error derivative with respect to weight a_{ij}.

μ is the momentum, which determines how significantly the previous weight change affects the current weight change.

Δa_{ij}^{m-1} is the previous weight change.

The sigmoid function is:

$g(u) = \frac{1}{1+e^{-u}}$

where u is the summation of incoming inputs (including the bias) multiplied by the corresponding weights.

The structure of a neural network significantly impacts its performance and training ability. If a neural network does not have enough connections between nodes or is insufficiently trained, it will be difficult to converge or to approximate the desired function well. On the other hand, using a network with dense connections or over-training will cause over-fitting. In order to find the best structure of a network, we turn to the help of genetic algorithms.

2.4 Genetic Algorithms: Finding the Structure for the Neural Network

This is a particular search problem where the search space is all possible configurations of networks. Genetic Algorithms (GAs) (Goldberg [2], Michalewicz [14]) are search algorithms that mimic natural selection. They have been widely used in very complex domains where regular methods are not applicable.

A Genetic Algorithm contains 5 key components: a representation or encoding, a selection method, genetic operators that include crossover and mutation, a fitness function, and a style of the Genetic Algorithm, generational Genetic Algorithm or steady-state Genetic Algorithm, which in turn determines population maintenance strategy.

A Genetic Algorithm maintains a population of solutions for a particular problem. Each solution is referred to as an individual. Each individual will get a fitness value according to some "goodness" measurement (fitness evaluation). During the evolution process, individuals first pair up, followed by crossover and mutation, offspring are born. The offspring are evaluated with the same fitness evaluation process. A new population is formed by the offspring with/without candidates from previous generations (generational GA with/without generation gap) or replace an existing individual with the offspring (steady-state GA). After several generations, the population will hopefully converge to a better population where a global optimum exists, which refers to the best solution to the problem.

A design of a genetic algorithm example is shown in Figure 2.3. The population size at generation t and generation t+1 is the same. The population sizes for crossover and mutation depend on the crossover probability Pc and mutation probability Pm respectively. Notice that in the example above, individuals 4 and 7 are selected

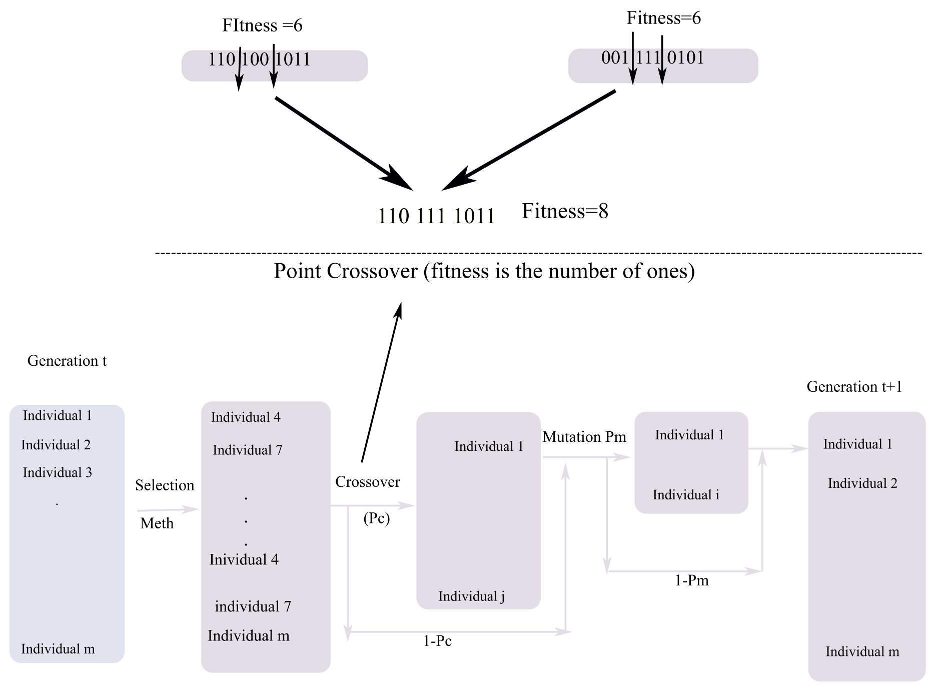


Figure 2.4: Elements of Genetic Algorithm

multiple times right after the selection, because they have better fitness value and the selection method favors better individuals.

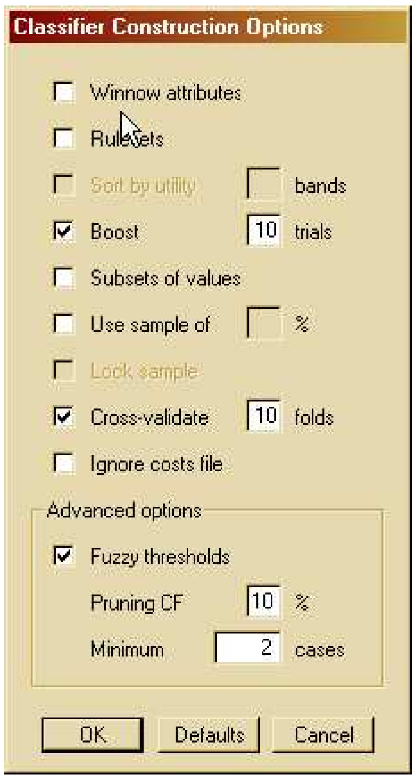
Chapter 3

Implementation details

3.1 Decision Tree Implementation

We continued the decision tree approach by using See5.0 [4] as our testing tool. See5.0 is the latest successor of ID3, an algorithm developed by Ross Quinlan (1993). It is very fast and easy to manipulate, and the performance of the decision tree can serve as the base line method that the other methods can be compared with.

Different methods were used to improve the prediction and reduce the error. Proposed methods include two kinds of predictions, one was binary prediction; the other one was multiple-class prediction. In the binary prediction case, we aggregate several treatments into one class, and the remaining parts into another class. In our case, we group treatments with better score values into a class that shares a common score 1, and the others into another class with score 0. In the multi-class prediction scenario, we allow 10 classes to exist. In each experiment, we constructed the decision tree with 11 patients (each patient with 5 treatment proposals, 55 cases in total), and left one patient for testing. Different options such as: cross-validation, boosting, pruning, and winnow attributes were combined together. Each option can be turned on by checking the corresponding check box located on the program menu. A snapshot of the program menu is shown in Fig 3.1. At the same time, we conducted training on both the full parameters and the reduced parameters (averages).



Classifier Construction Options

☐ Winnow attributes
☐ Rulesets
☐ Sort by utility [ ] bands
[✓ Boost] [10] trials
☐ Subsets of values
☐ Use sample of [ ] %
☐ Lock sample
[✓ Cross-validate] [10] folds
☐ Ignore costs file

Advanced options
[✓ Fuzzy thresholds]
Pruning CF [10] %
Minimum [2] cases

Figure 3.1: See5 program control menu

3.2 Classifier System Implementation

We adopted the idea of classification based on accuracy by Stewart W. Wilson [13]. Instead of using strength, it uses prediction accuracy as its fitness. A Java implementation of this algorithm (XCS) by Martin V. Butz (2000)[12] can be found on the Illinois Genetic Algorithms Laboratory web page[1]. We added a class to send our problem to the system, and we rewrote the Genetic Algorithm to meet our needs. Since the original system takes a binary representation and we want a real valued prediction, we implemented a sub-routine to convert the actual input and output into the right format.

[1]http://gal4.ge.uiuc.edu/sourcecd.html

In the training phase, the classifier system takes each treatment as an input; it generates a classifier, which is made of a condition part and an action part. An internal classifier list is maintained and each classifier covers some examples. The system tries to generalize each classifier, make it cover as many examples as it can, and the corresponding action would match the target action value. In the prediction phase, the system takes the query from environment–in our case a string describing the new patient–and tries to match it with existing classifiers. A classifier that matches this statement will post its message to the environment. If several classifiers fire, the classifier with the highest strength will be considered the winner. Its action is posted to the environment, and a reward based on the reward scheme will be obtained.

3.3 Artificial Neural Network Implementation

A simple backpropagation neural network (A.N.N.) that includes momentum and learning rate features was implemented for testing. Although there are many existing artificial neural network packages, we decided to write our own program simply because we planned to use genetic algorithms (GA) to set artificial neural network architectures(G.A.A.N.N.). Writing our own program adds more effort, but actually we gained more control. Also, we want to compare the performance of pure A.N.N approach and G.A.A.N.N. approach. Because we had 10 possible scores, we experimented with networks with 10 output nodes and networks with one output node with 10 partitions. We selected the choice of a network with one output node with 10 partitions for the faster training process, especially for the G.A.A.N.N. approach(refer to the next section). Users can specify the value for momentum, learning rate and tolerance. Tolerance is a value that controls the variance of the network, it is used here to decide how close an output is to the target to be considered a match, in another words, it defines the partial match rate. If there is a difference between an output and the actual target within the tolerance value, we consider the prediction to be correct.

A stopping detector was implemented as well. We considered the testing error as the key factor to determine when to cease training. We computed the mean error for 10 consecutive epochs, and we compared it with the global minimum (best error so far). If the current average error is smaller than the global minimum, which corresponds to the network searching towards the right direction, it should continue; if it is greater than or equal to the global minimum, it indicates that either the network is in a local minimum or the beginning of over-fitting. Instead of stopping the training right there, we allowed the program to continue searching for another certain number of epochs. If the situation changes within the specified epochs (i.e. the current average error got smaller than the global minimum) the program continues the training, otherwise, training is stopped. The main idea behind this was to avoid over-fitting without getting stuck in local optima. The best network was saved during the training process.

3.4 Genetic Algorithm Implementation

We selected a generational GA since it is easy to maintain population diversity. For the representation, we considered that the number of nodes in the hidden-layer, momentum value, learning rate value and the tolerance value that controls the variance of the network are the most important factors. A four-tuple real valued vector forms the representation. There are many selection methods we can choose from. In this project, we adopted a simple proportional selection method, which is easy to implement, yet provides enough selection pressure. A single point crossover and a random mutation have been tested in the program. In our case, no single function can be used to evaluate the goodness of an individual, instead of an actual mathematical formula, the fitness function is the artificial neural network simulator that we constructed before. To get a better view of this approach, refer to Figure 3.2.

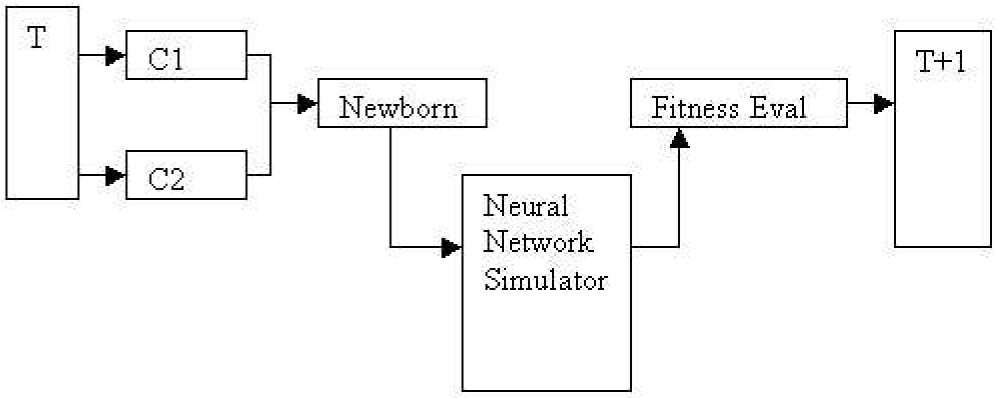


Figure 3.2: A Genetic Algorithm with an Artificial Neural Network simulator

Upon evaluating the fitness of a point, the simulator is invoked and a value is returned. This value acts as the fitness value. The idea is that the Genetic Algorithm generates a population of solutions, with each solution representing a possible training setup. Each solution is assigned its fitness value by constructing a network with the specified configuration and training the network followed by testing, then the mean squared error of the prediction set is returned as fitness. During the search process, a pair of candidates is randomly selected, then crossover and mutation are conducted, which generates an offspring with a fitness value. The best individual is preserved for the next generation.

Several features are included in this design. A dynamic data partition sub-routine that divides the whole data set into a training set, a testing set and a validation set. The benefit is obvious, we do not need to determine which instances are used for training, which for testing and so forth. At the same time, we hope that this idea could prevent the network from favoring a particular region of the search space, hence it will generalize better. The same stopping detector described above was used.

Chapter 4

Experiments and Results

To compare the performance of the different methods, we conducted our experiments on the same data set. For each method, we left one case (dog) out for testing, one for validation, and the other cases for training, so-called leave-one-out cross validation. We repeated this method for all the data and took the average. Then, we constructed a confusion matrix for each method to summarize the overall performance. In the table, the horizontal direction stands for the target value and the vertical direction for prediction value. The digits in the table denote the frequency of prediction for a particular target. The more the data are concentrated on the diagonal direction, the better the performance of the method. Our final conclusion and discussion follows.

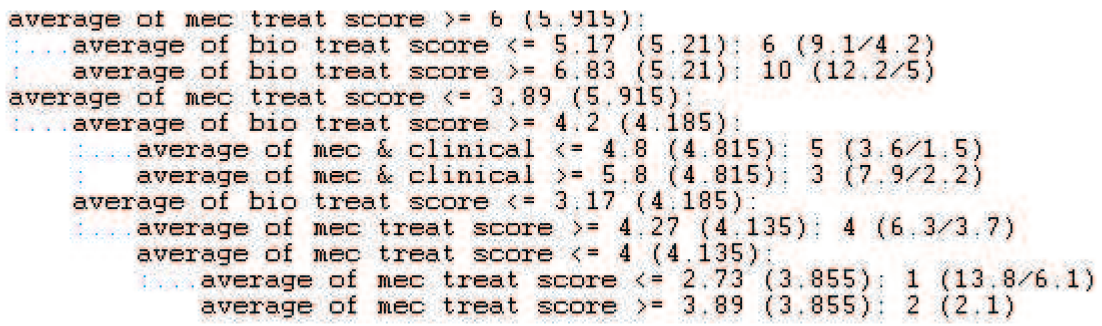
4.1 Decision Trees

The first method that we used was decision trees. As we mentioned before, we tested each method on both full parameters and reduced parameters, we repeatedly constructed a tree classifier for each method followed by testing. The average accuracy for binary prediction using average parameters is about 91.7%. The average accuracy for binary prediction using full parameters is 86.7%. The average accuracy for multi-class prediction on both reduced parameters and full parameters was worse than binary prediction with the same parameters. The average accuracy for the 10-class prediction was 73.3% with full data and 66.6% with averages only. We also

Table 4.1: Decision tree with full data and multiple-class prediction.

		Target									
		1	2	3	4	5	6	7	8	9	10
	1	9	1	1							
	2	1	3								
	3		1	8	2	1					
	4				1						
Prediction	5			1		4					
	6						7	1	2		
	7							2			
	8			1			1		3		2
	9										
	10									1	7

noticed that different aggregation methods would affect the accuracy of the prediction. A good partition we discovered is to make the treatments with a final score above 5 into one class and the rest are put into another class.



```
average of mec treat score >= 6 (5.915):
:...average of bio treat score <= 5.17 (5.21): 6 (9.1/4.2)
:   average of bio treat score >= 6.83 (5.21): 10 (12.2/5)
average of mec treat score <= 3.89 (5.915):
:...average of bio treat score >= 4.2 (4.185):
    :...average of mec & clinical <= 4.8 (4.815): 5 (3.6/1.5)
    :   average of mec & clinical >= 5.8 (4.815): 3 (7.9/2.2)
    average of bio treat score <= 3.17 (4.185):
    :...average of mec treat score >= 4.27 (4.135): 4 (6.3/3.7)
        average of mec treat score <= 4 (4.135):
        :...average of mec treat score <= 2.73 (3.855): 1 (13.8/6.1)
            average of mec treat score >= 3.89 (3.855): 2 (2.1)
```

Figure 4.1: Reduced parameters with 10-class prediction.

From Tables 4.1, 4.2 and 4.3, we concluded that the result obtained by using full data for training and testing is slightly better than the result from using the reduced data set for 10-class prediction. We observed that the tree rules are more oriented towards treatment rather than patient information. This information can be perceived by studying tree classifiers generated from the program. As a tree goes deeper, the importance of each pivot value at that level decrease. We noticed that

Table 4.2: Decision tree with average data and multiple-class prediction.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	10	3	3							
	2		2								
	3		1	6	1	2					
	4				1		1				
	5			1		3		1			
	6						7		1		1
	7										
	8				1			1	3		
	9										
	10							1	2		8

Table 4.3: Decision tree with full data and average data for binary prediction

	Target					
	Average Data			**Full Data**		
Prediction		0	1		0	1
	0	30	2	0	32	6
	1	3	25	1	2	20

almost all the trees start with treatment parameters at the root level, only at lower level patient information parameters act as pivot value. At the same time, using the binary target for classification is better than that of using multi-class prediction. A yes/no answer is much easier to give that an exact score value. If we have adequate training data, we might reduce the error and the system should perform better than the result obtained.

4.2 Classifier System

We followed the same steps and experimented on the classifier system. We concluded from Tables 4.4, 4.5 and 4.6 that the classifier system has trouble in learning this data set. Overall, there are 60 points, but it only correctly predicted 13 points, which is 21% accuracy. The classifier system with average data did slightly better. The accuracy is 43%, but still below the decision tree. For binary prediction, both the classifier system with full data and the classifier system with average data did well. The accuracy is 75% and 80% respectively. During the experiment we noticed that the reward scheme affected the performance of the system greatly. Two different schemes were tested:

Full Reward. For each prediction, if the prediction matches the target action, a maximum payoff value is rewarded. Otherwise, the reward is inversely proportional to the difference between the target value and the prediction, i.e. the greater the difference between target and prediction, the lesser the reward.

Binary Reward. If the prediction matches the target action, a maximum payoff value is rewarded; otherwise, the reward is zero.

In Figure 4.2 we compared the performance of the program based on full parameters . The y-axis denotes the prediction of the previous 50-iterations, which is an inverse function of the prediction error. The x-axis is the iteration number. From the

Table 4.4: Classifier system with full data and multiple-class prediction using binary representation.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	5	2	1		1	1	1			
	2	2	2	2					1		
	3			1		1	1		3		1
	4			3							
	5	1			1	1	1			1	2
	6	1		3			2				2
	7	1			2	1	1				1
	8						1	2	1		2
	9		1			1	1				
	10									1	1

Table 4.5: Classifier system with reduced data and multiple-class prediction using binary representation.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	7	1								
	2	2	3	1	1						
	3	1	1	5	1	1					
	4			3		2	1				
	5			1	1	2	2				
	6						2				
	7						3	1	1		1
	8							1	3	2	2
	9							1	1		3
	10										3

Table 4.6: Classifier system with full data and average data for binary prediction

	Target					
	Average Data			**Full Data**		
Prediction		0	1		0	1
	0	30	4	0	28	6
	1	6	20	1	8	18

graph, we can see that the reward scheme affects the performance of the program. Selecting the right reward scheme could help the system predict better. Notice that the program with full reward is slightly better than using binary reward. We thus used the full reward approach in the rest of our experiments.

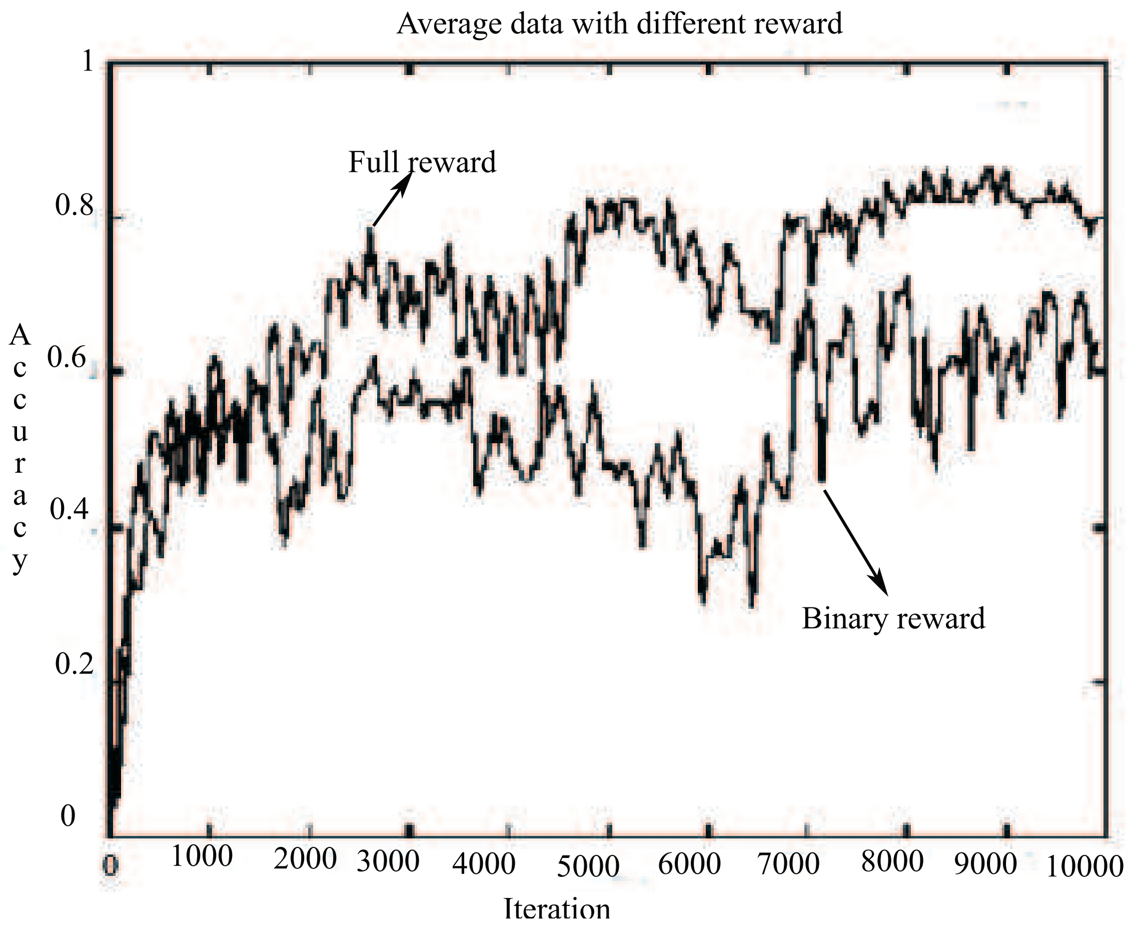


Figure 4.2: Comparison of reward scheme with average data set.

Overall performance of using full data is worse than using the reduced data set. A possible reason is that a classifier with full parameters as its condition part will suffer from searching in high dimensional spaces, at the same time, a large population has to be maintained in order to achieve better accuracy. This is particularly true since we used a binary representation.

Table 4.7: Artificial neural network with full data and multiple-class prediction.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	8	2	1							
	2	2	2	3							
	3		1	5	1						
	4			1	1	4	2				
	5				2		1				
	6			1			1		1		
	7						1	2	3	1	3
	8						2				2
	9										
	10						2		1		4

4.3 Artificial Neural Network

In this experiment, we used a leave-one-out cross validation approach in which we trained a network using ten patients, then, validated the network on one patient and used another one for testing. We manually adjusted the parameters based on repeated experiments.

Table 4.7 summarizes the performance of our artificial neural network with full data set and 10-class prediction with accuracy 31.7%. Table 4.8 summarizes the performance of our Artificial Neural Network with the average data set and multiple-class prediction with accuracy of 45%.

Table 4.9 compares the performance of our artificial neural network with average and full data for binary prediction with accuracy of 91.6% and 95% respectively. Clearly, we can see that both approaches did better than the classifier system approach, while they did worse than the decision tree method.

Table 4.8: Artificial neural network with average data and multiple-class prediction.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	7	1								
	2	3	3	4							
	3		1	7	1	1					
	4					2					
	5				1	2	2				
	6				1		2				
	7						3	1	2	1	
	8						1		3		2
	9							1			5
	10										2

Table 4.9: Artificial neural network with average and full data with binary prediction.

	Target					
	Average Data			**Full Data**		
Prediction		0	1		0	1
	0	31	2	0	31	0
	1	3	24	1	3	26

Table 4.10: Genetic Algorithm for finding the structure for artificial neural network with full data and multiple-class prediction.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	8									
	2	3	2	2							
	3			7	1						
	4			2	4	1					
	5					4	2				
	6						4				
	7						2	2	3		
	8								2	1	4
	9										5
	10										1

4.4 Genetic Algorithm for Finding ANN Structure

The performance of the Genetic Algorithm is promising. In the first attempt, it found a near optimal solution with a fitness value of 638.327, which corresponds to the mean squared error of 0.156 according to the fitness function we proposed. This is better than the results we obtained by manually setting the neural network parameters (by which, we got a solution with mean squared error 1.25).

From Tables 4.10, 4.11 and 4.12 we noticed that using the Genetic Algorithm for finding a better structure for the artificial neural network resulted in great success. Considering the simplicity of the Genetic Algorithm we are using (population size 20, generation 10, crossover probability 0.8, mutation rate 0.3, single-point crossover, and random mutation), we believe that applying the Genetic Algorithm in finding a set of parameters for the neural network is a promising approach.

Table 4.11: Genetic Algorithm for finding the structure for artificial neural network with reduced data and multiple-class prediction.

		Target									
		1	2	3	4	5	6	7	8	9	10
Prediction	1	9	1								
	2	2	3	4							
	3			7							
	4			1	2	3					
	5				1	2	6				
	6						3				
	7							2		1	
	8								3		1
	9								1		7
	10										1

Table 4.12: Genetic Algorithm for findig structure for artificial neural network with average and full data with binary prediction.

	Target					
	Average Data			**Full Data**		
Prediction		0	1		0	1
	0	34	0	0	34	0
	1	0	26	1	0	26

Chapter 5

Future Work

The data type that we are dealing with is a long vector of real values. The question is whether or not each factor significantly contributes to the final prediction. A simple statistical analysis (Ott [8]) has been done on the data and we discovered that there are several factors that do not provide any help in predicting the target value. A feature extraction routine can be used to exclude the nuisance factors.

The way to measure the performance of the system needs to be reconsidered. In our experiments, we count how many times the system guesses correctly. Another reasonable approach could be to count the correct guesses and compute the standard deviation of the wrong guesses the system makes, since we want the guesses as close as possible to the targets.

Skurichina and Duin [6] studied weak classifiers with small data sets. They concluded that a classifier constructed using a relatively small number of observations from data with a large number of features is biased. They also suggested readers using techniques like bagging, boosting and random subspace to adjust the performance of the classifier.

In our case, in order to improve the performance of the classifier system, we need to improve the expressiveness of the classifier. This is particularly true since there is a small data set involved.

The accuracy of the neural network predictions may be improved by using more than one output node. We have limited our experiments to one output node for speed considerations. It may be worthwhile to further experiment with networks with multiple output nodes.

We need to start thinking about how to speed up the whole process. Training a neural network is time consuming, and applying a Genetic Algorithm search for a neural network structure takes a long time. Constructing the classifier system is also time-consuming. Currently, all this is done offline. It will be much better if the system could be adaptive to the new examples. If the retraining process can be done quickly, we could afford to retrain the system for better accuracy. Methods like least squares analysis have been extensively studied in previous research (Rasheed et al. [5]). They demonstrate high efficiency with yet reasonable accuracy. These methods could be successfully apply in this domain as well. The quantity and quality of the data greatly affect the performance of the program. We will work closely with researchers at the veterinary school of UGA to obtain more data.

Finally, a good continuation to this work is to design a user interface that could take a description of the patient and convert it into the right data format for prediction. We also intend to design an image processing toolkit to view and identify the bone fracture. Finally, a recommendation system will be built on top of the evaluation system to design treatments by running a design optimizer (probably a Genetic Algorithm) whose objective function is the evaluation module.

Chapter 6

Conclusion

This thesis has presented a comparison of several methods in assessing bone fixation proposals. Decision trees did very well in binary prediction and also reasonably well in 10-class prediction. In our experiment, we got 91.7% and 86.7% accuracy by using decision trees for binary prediction with full data and average data respectively and 73.3% and 66.6% for 10-class prediction by using full and average data respectively.

The classifier system with full data did poorly in 10-class prediction, with accuracy of only 21%, it did well with average data and gave 43% accuracy. For the binary prediction, the accuracy is 75% and 80% by using full and average data respectively. Due to the small sample size, the classifier set that we constructed through training could be biased. The influence was that the prediction for an unseen example was not always accurate. Most of the time, the prediction exactly matches the target, but some times, it gives different results.

The artificial neural network did very well for binary prediction. In our experiments, a very simple network was able to get 31.7% and 45% accuracy for 10-class prediction by using full and average data, and 95% and 91.6% accuracy for binary prediction respectively.

Using the Genetic Algorithm to find the structure of a neural network shows a great deal of promise. We were able to get 100% accuracy in binary prediction by using both full and average data. For 10-class prediction, we obtained 63.3% and 53.3% accuracy by using full and average data respectively.

Table 6.1: Multiple-Class Prediction Comparison

	Decision Tree	**Classifier System**	**A.N.N.**	**G.A.A.N.N.**
Reduced Data	66.6%	43%	31.7%	53.3%
Full Data	73.7%	21%	45%	63.6%

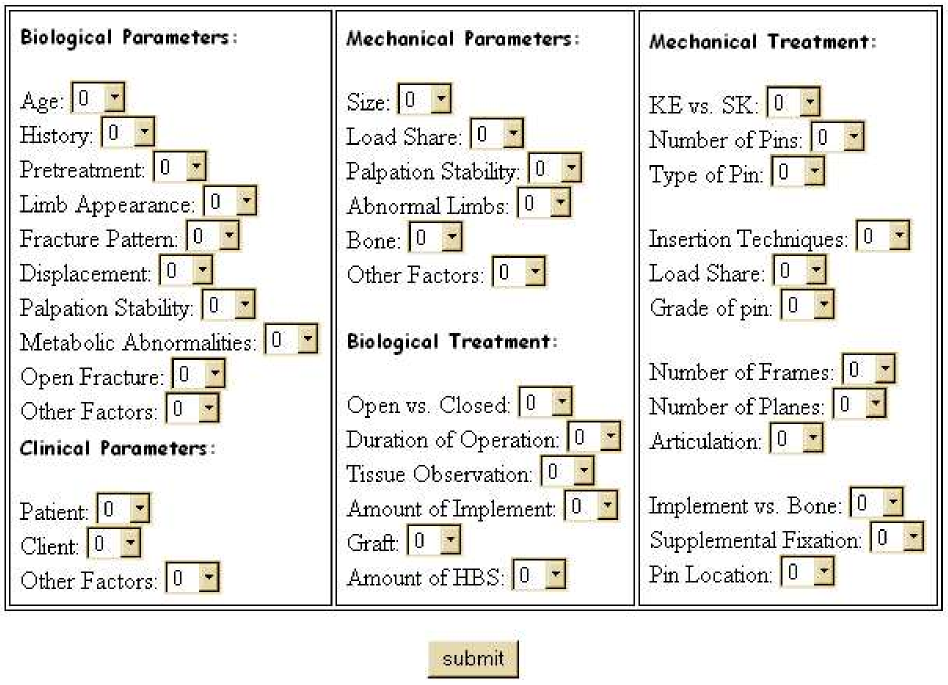
Table 6.2: Binary Class Prediction Comparison

	Decision Tree	**Classifier System**	**A.N.N.**	**G.A.A.N.N.**
Reduced Data	91.7%	80%	91.6%	100%
Full Data	86.7%	75%	95%	100%

There is one shortcoming about neural network is that it is not human readable. There is no obvious way to tell which factor is more important than the others, while a decision tree can be easily converted to a set of if-then rules. Comparison of performance for each method is presented in Tables 6.1 and 6.2.

We also created a simple CGI enabled web page [1] to let users evaluate their proposed methods. Once a user selected his/her proposal and submitted the form, the CGI program will invoke the actual program–a neural network constructed ahead of time, and the output of the neural network displays on the web page. A snapshot of the input form is shown below.

[1]https://machinelearning.dnsalias.net/FPAS.php



Biological Parameters:	**Mechanical Parameters:**	**Mechanical Treatment:**
Age: 0	Size: 0	KE vs. SK: 0
History: 0	Load Share: 0	Number of Pins: 0
Pretreatment: 0	Palpation Stability: 0	Type of Pin: 0
Limb Appearance: 0	Abnormal Limbs: 0	Insertion Techniques: 0
Fracture Pattern: 0	Bone: 0	Load Share: 0
Displacement: 0	Other Factors: 0	Grade of pin: 0
Palpation Stability: 0	**Biological Treatment:**	Number of Frames: 0
Metabolic Abnormalities: 0	Open vs. Closed: 0	Number of Planes: 0
Open Fracture: 0	Duration of Operation: 0	Articulation: 0
Other Factors: 0	Tissue Observation: 0	Implement vs. Bone: 0
Clinical Parameters:	Amount of Implement: 0	Supplemental Fixation: 0
Patient: 0	Graft: 0	Pin Location: 0
Client: 0	Amount of HBS: 0	
Other Factors: 0		

Figure 6.1: Web page for users to evaluate their proposal

Bibliography

[1] Potter W. D. Bone-fixes: An external skeletal fixation expert system. In *The 2000 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, Las Vegas, 2000.

[2] Goldberg David. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman, Massachusetts, 1989.

[3] Harold Edward Dale Jr. Bone-fixes: An expert system for external skeletal fixation. Master's thesis, The University of Georgia, 2000.

[4] Quilan J.R. *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.

[5] Rasheed K., Vattam S., and Ni X. Comparison of methods for using reduced models to speed up design optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, 2002.

[6] Skurichina M. and Duin P.W. R. Bagging and the random subspace method for redundant feature space. In *Multiple Classifier Systems*, pages 1–10. Springer, 2001. Second International Workshop, MCS 2001 Cambridge, UK.

[7] Palmer R.H. *Preoperative Decision-Making Mobil: The Fracture-Patient Assessment Score (FPAS)*. The University of Georgia College of Veterinary Medicine and Georgia Center for Continue Education, March 24–26 2002. 10th Annual Complete Course in External Skeletal Fixation.

[8] Ott. R.L. and Longnecker M. *An Introduction to Statistical Methods and Data Analysis.* Duxbury Press Pacific Grove, California, 2001.

[9] Lehr M. Rumelhart D., Widrow B. The basic ideas in neural networks. *Communications of the ACM 37(3)*, pages 87–92, 1994.

[10] M. Smith. *Neural Networks for Statistical Modeling.* Van Nostrand Reinhold, New York, 1993.

[11] Mitchell T. *Machine Learning.* WCB/McGraw-Hill, New York, 1997.

[12] Butz V.Matin. Xcsjava 1.0: An implementation of xcs classifier system on java. Technical report, Illinois Genetic Algorithms Laboratory, 2000. IlliGAL Report No.2000027.

[13] S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation 3(2)*, pages 149–175, 1995.

[14] Michalewicz Z. *Genetic Algorithms + Data Structure=Evolution Programs.* Springer, New York, 1996. 3rd Edition.