
Machine Learning Techniques for the Evaluation of External Skeletal Fixation Structures

Ning Suo

Artificial Intelligence Center, The University of Georgia, Athens, GA 30602 USA

SUONING@UGA.EDU

Khaled Rasheed

Walter D. Potter

Computer Science Department, The University of Georgia, Athens, GA 30602 USA

KHALED@CS.UGA.EDU

POTTER@CS.UGA.EDU

Dennis N. Aron

College of Veterinary Medicine, The University of Georgia, Athens, GA 30602 USA

DARON@VET.UGA.EDU

Abstract

In this paper 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 prediction 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 more time in training. A Genetic Algorithm is also employed to find the best structure of the Neural Network. Experimental results for the different methods are presented and compared.

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. Figure 1 shows a Type Ib fixator. It contains one frame with three planes, which secure the bone fracture, yet allow the animal to move around moderately.

Researchers and veterinarians at the Veterinary School of UGA 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 al in teaching by allowing students to test their proposed treatment plans through this system before going through expensive experiments.

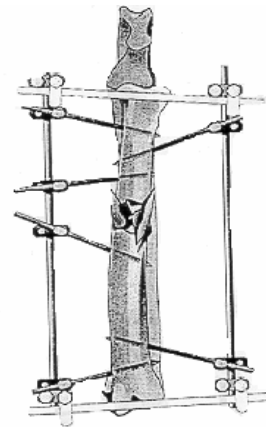


Fig 1. An external skeletal fixation device--Type Ib fixator

It can also be used to demonstrate the bone fracture 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 treatment itself. The biological features of the wound and fixation mechanics selection constitute this category.

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. That final balance score is the training target.

Table 1. Part of the Scoring System.

BIOLOGICAL		MECHANICAL	
AGE	3	SIZE	4
HISTORY	2	LOAD	6
PRETREATMENT	0	BONE	8
LIMB	2	CLINICAL	
FRACTURE	4	PATIENT	6
DISPLACEMENT	5	CLIENT	7
PALPATION	2	OTHER	4

2. Related Work

Previously, Potter (2000) and Dale (2000) designed an expert system called "Bone-FixES" to analyze and evaluate treatment plans. An expert system is a computer program that utilizes a human expert's experience to tackle real-world problems. It is suitable for domains with complex symbolic representation and finite elements. 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, with a finite element analysis feature built in, 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.

Decision tree learning (Mitchell 1997) is a widely used and practical machine learning technique for inductive inference. 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 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.

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

We continued this project by taking 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 paper presents a brief description of our proposed methods for multiple-class prediction. The paper then presents a number of experiments. Following that, we conclude the paper with a discussion of the results and future directions.

3. Proposed Methods

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 structure 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 treatment, there are 35 parameters that contribute to the final score. One obvious way is to take all 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. In order to accomplish that, we have to transform those variables using a mathematical formula. We take the average of the parameters in our experiment, which shrinks the parameters into 4 key factors. The validity of both approaches is checked in the experiment.

We then explain each method in detail in the remainder of this section.

3.1 Decision Tree Revisited

We continued the decision tree approach by taking See5.0 as our premier testing tool. See5.0 is an implementation of 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 “blank sample” that the other methods can be compared with.

Different methods were taken to maximize 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 group. 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 that more than two classes may exist. We constructed the decision tree with 11 patients (each patient with a 5 treatment proposal, 55 cases in total), and left one patient for testing. Different features such as: cross-validate, boost, pruning, and winnow attributes are combined together. At the same time, we conducted training on both the full parameters and the reduced parameters approaches.

3.2. 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.

We adopted the idea of classification based on accuracy by Stewart W. Wilson (1995). Instead of using strength, it uses prediction accuracy as its fitness. A Java implementation of this algorithm by Martin V. Butz (2000) can be found on the Illinois Genetic Algorithms Laboratory web page. We added a class to send our problem to the system, and we rewrote the genetic algorithm to meet our own purpose. 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. 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 a statement, in our case a string describing the new patient, a classifier that matches this statement will post its message to the environment. If more than one classifier fires, the classifier with 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.

Apart from that, following Stewart W. Wilson’s suggestion (2001), we revised XCS to a classifier system that could take real values as input and produce real-valued output.

3.3 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 is much simpler, but most of the time, it is not. The fact is that our brain can handle these situations surprisingly well. That is why artificial neural networks came to the rescue. 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.

A simple backpropagation neural network (Smith 1993) that includes momentum and learning rate features was implemented for testing. Users can specify the value for momentum, learning rate and spread. Spread is a value that controls the variance of the network. If there is a difference between an output and the actual target within the spread value, we consider the prediction to be correct.

In the experiment, we kept on training the network until we reached pre-selected epochs since we wanted to see the stability of the network. We repeatedly ran this network with different settings and training epochs. By looking at the result of previous runs, we determined the stopping epochs based on testing error. Then we retrained a network with the same parameters but used the stopping epochs discovered in the previous runs.

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*.

3.4 Genetic Algorithms Finding the Structure for the Neural Network

In order to find the better structure of a network, we turn to the help of genetic algorithms. This is a particular search problem where the search space is all possible configurations of networks.

Genetic Algorithms (GAs) (Goldberg 1989, Michalewicz 1996) 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 a population maintenance strategy.

Generational Genetic Algorithm: We selected a generational GA since it is easy to maintain population diversity.

Representation: We considered that the number of nodes in the hidden-layer, momentum value, learning rate value and the spread value that controls the variance of the network are the most important factors. A 4-tuple real valued vector forms the representation.

Selection Methods: A simple proportional selection method was adopted to provide enough selection pressure.

Genetic Operators: Single point crossover and a random mutation have been tested in the program.

Fitness Function: In our case, instead of an actual mathematic formula, the fitness function is the artificial neural network simulator that we constructed before. Upon evaluating the fitness of a point, the simulator is evoked and a value 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 network configuration. Each solution obtains its fitness value by the initial fitness evaluation—constructing a network with the specified configuration and training the network followed by testing, the mean squared error of the prediction set is returned for fitness. During the search process, a pair of candidates is randomly selected, then crossover and mutation is 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 prediction 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. A

stopping detector was implemented as well. From the previous study, we considered 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. A global minimum is the best point 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 or equal to the global minimum, it indicates either the network is in a local minimum or the start 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 changed within the specified epochs (i.e. the current average error got smaller than the global minimum) the program continues the training, otherwise, training 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.

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 for prediction, one for validation, and the other cases for training. We repeated this method for all the data. Then, we constructed a table for each method to summarize the over all performance of each method. In the table, the horizontal direction stands for the target value and the vertical direction for prediction value. The more the data is concentrated on the shaded diagonal line, the better the performance of the method. Our final conclusion and discussion follows.

4.1 Decision Trees

The first method that we used was the decision tree. We repeatedly constructed a tree classifier followed by testing it on both full parameters and reduced parameters. 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 noticed that different aggregation methods would affect the accuracy of the prediction. A better 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.

From Table 2, 3 and 4, we concluded that the result obtained by using average data for training and testing is the same as the result from using the full data set. Our guess is that the tree rules are more towards treatment rather than patient information.

At the same time, using the binary target for classification

```

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)

```

Fig. 2. A decision tree generated by using reduced parameters with 10 classes prediction

is better than that of using multi-class prediction. The reason might be due to the complexity in learning the difference between the lower target class and developing tree rules. If we have adequate training data, we might reduce the error and the system should perform better than the result obtained.

Table 2. Decision tree with full data and multiple class prediction

		TARGET									
		1	2	3	4	5	6	7	8	9	10
P R E D I C T I O N	1	9	1	1							
	2	1	3								
	3		1	8	2	1					
	4				1						
	5			1		4					
	6						7	1	2		
	7							2			
	8			1			1		3		2
	9										
	10									1	7

4.2. Classifier System

We can conclude from Table 5 that the classifier system has trouble in learning this data set. Over all, there are 60 points, it predicted correctly only 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 Rewarding. For each prediction, if we match the target action, a maximum payoff value is rewarded.

Table 3. Decision tree with full data and average data for binary prediction

		TARGET				
		AVERAGE DATA		FULL DATA		
P R E D I C T I O N		0	1	0	1	
	0	30	2	32	6	
	1	3	25	2	20	

Table 4. Decision tree with average data and multiple class prediction

		TARGET									
		1	2	3	4	5	6	7	8	9	10
P R E D I C T I O N	1	10	3	3							
	2		2								
	3		1	6	1	2					
	4			1	1		1				
	5					3		1			
	6						7		1		1
	7										
	8				1			1	3		
	9										
	10							1	2		8

Table 5. Classifier system with full data and multiple class prediction using binary representation.

		TARGET									
		1	2	3	4	5	6	7	8	9	10
P R E D I C T I O N	1	5	2	1		1	1	1			
	2	2	2	2					1		
	3			1		1	1		3		1
	4			4							
	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

Otherwise, a reverse function is introduced--reward value is proportional to the maximum payoff. The greater the difference between target and prediction, the lesser reward gained. A Penalty function is introduced as well.

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

In Figure 4, 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 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. In Figure 5, we compared the performance of the program based on full parameters with two different reward schemes.

Notice that the program with full rewarding is slightly better than using binary rewarding. Overall performance of using full data is worse than using the reduced data set.

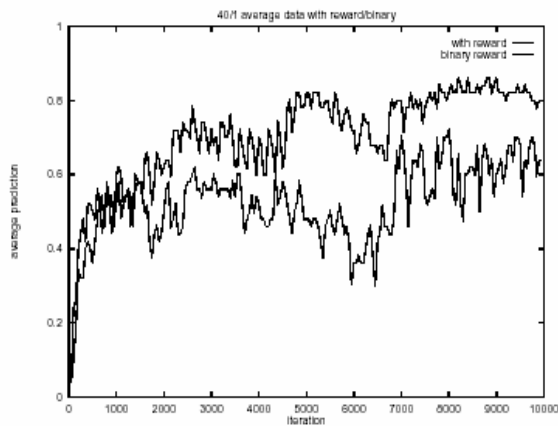


Fig. 4. Comparison of reward scheme with average data set. (top is full reward scheme, bottom is binary scheme).

The 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 in the case where we use a binary representation.

We also introduced a partial match concept for an integer representation in the prediction process to make the system generalize better. The rate of partial match is defined by a variable, which is given by the user.

4.3 Artificial Neural Network

In this experiment, we used a leave-one-out cross validation approach in which we trained a network using 10 patients, then, tested the network on 1 patient and another one for prediction. We manually adjusted the parameters based on repeated experiments.

Table 6 summarizes the performance of our artificial neural network with full data set and 10-class prediction with accuracy 31.7%. Table 7 summarizes the performance of our artificial neural network with the

Table 6. Artificial neural network with full data and multiple class prediction

		TARGET									
		1	2	3	4	5	6	7	8	9	10
P R E D I C T I O N	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						2		1		4
	10										

Table 7. Artificial neural network with average data and multiple class prediction

		TARGET									
		1	2	3	4	5	6	7	8	9	10
P R E D I C T I O N	1	7	1								
	2	3	3	4							
	3		1	7	1	1					
	4					2	1				
	5				1	2	2				
	6				1		2				
	7						3	1	2	7	
	8						1		3		2
	9							1	1		5
	10										2

Table 8. Artificial neural network with average and full data with binary prediction

		TARGET					
		AVERAGE DATA			FULL DATA		
P R E D I C T I O N		0	1		0	1	
	0	31	2	0	31	0	
	1	3	24	1	3	26	

average data set and 10-class prediction with accuracy 45%. Table 8 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 both approaches did better than the classifier system approach, while they did worse than the decision tree method.

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).

Table 9. Genetic algorithm in 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
P R E D I C T I O N	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

From Tables 8, 9 and 10, we noticed that using the genetic algorithm for finding a better structure for the artificial neural network resulted in great success. Consider 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 going to be a good approach.

5. Future Work

The data type that we are dealing with is a long vector of real values. The question is whether or not the factors equally contribute to the final prediction. A simple statistical analysis (Ott 2001) 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 will be used to exclude the nuisance factors.

The way to measure the performance of the system needs to be reconsidered. In our experiment, 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 (2001) studied weak classifiers with small data sets. They concluded that a classifier

Table 10. Genetic algorithm finding the structure for artificial neural network with average data and multiple class prediction

		TARGET									
		1	2	3	4	5	6	7	8	9	10
P R E D I C T I O N	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 11. Genetic algorithm in finding the structure for artificial neural network with full data and binary prediction.

		TARGET				
		AVERAGE DATA		FULL DATA		
		0	1	0	1	
P R E D I C T I O N	0	34	0	0	34	0
	1	0	26	1	0	26

constructed using a relatively small number of observations from data with a large number of features is biased. They also suggested to use 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 classifier for the classifier system. This is particularly true since there is a small data set involved.

We need to start thinking about how to speed up the whole process. Training a neural network is time consuming, and applying 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. A possible solution is to use hybrid approaches. Methods like least squares analysis have been extensively studied in our previous research (Rasheed et al. 2002), it demonstrates high

efficiency with yet reasonable accuracy. Using a quadratic least squares method instead of a neural network along with the genetic algorithm might give better results.

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.

We intend 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 GA) whose objective function is the evaluation module.

6. Final Remarks

This paper 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 multi-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. It is also possible to generate a continuous prediction by the program but, to a veterinarian, discrete prediction makes more sense than a continuous prediction. For instance, a prediction of 5 is better than a prediction of 4, but 4.9 and 4.3 might be confusing.

The classifier system with binary representation did poorly in 10-class prediction, with accuracy of only 21%, but the real valued classifier system did well, we got 43% of 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 success. 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. The shortcoming is that the network 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.

Acknowledgements

The authors present special thanks to Marc Schenkel for helping us prepare the data and standard format and his numerous contributions to this project.

References

- Butz V.M. (2000) XCSJava 1.0: An Implementation of XCS Classifier System on Java. *IlligAL Report No. 2000027*
- Dale H.E. (2000) Bone-FixES: An External Skeletal Fixation Expert System. *M.S. Thesis.*
- Goldberg D. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. *Addison-Wesley*
- Michalewicz Z. (1996) Genetic Algorithms + Data Structure=Evolution Programs Third Edition. *Springer*
- Mitchell T. (1997) Machine Learning. *WCB/McGraw-Hill*
- Ott. R.L. and Longnecker M. (2001) An Introduction to Statistical Methods and Data Analysis. *Duxbury Press Pacific Grove, CA*
- Palmer R.H. (2002) Preoperative Decision-Making Mobil: The Fracture-Patient Assessment Score (FPAS). *10th Annual Complete Course in External Skeletal Fixation, 55-62.*
- Potter W.D. (2000) Bone-FixES: An External Skeletal Fixation Expert System. *METMBS'2000*
- Quilan J.R.(1993) C4.5: Programs for Machine Learning. *Morgan Kauffman.*
- Rasheed K. and Vattam S. and Ni X. (2002) Comparison of Methods for Using Reduced Models to Speed Up Design Optimization. *The Genetic and Evolutionary Computation Conference (GECCO'2002)*
- Skurichina M. and Duin P.W. R(2001) Bagging and the Random Subspace Method for Redundant Feature Space. *Multiple Classifier Systems. Second International Workshop, MCS 2001 Cambridge, UK. Springer, 1-10.*
- Smith, M. (1993) Neural Networks for Statistical Modeling. *Van Nostrand Reinhold, NY*
- Wilson, S.W.(1995). Classifier fitness based on accuracy *Evolutionary Computation, 3(2), 149-175.*
- Wilson, S.W.(2001) Function approximation with a classifier system. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), L. Spector et al, eds. Morgan Kaufmann, San Francisco, CA, pp. 974-981*