Predicting Return to Work with Data Mining

Barry Senensky FSA FCIA MAAA
Jonathan Polon FSA

Claim Analytics
January 12, 2004

TABLE OF CONTENTS

Predicting Return to Work with Data Mining

Executive Summary

Claim Analytics was founded in early 2001, with the objective of using data mining tools to create new solutions for the insurance industry.

Data mining is the term for the type of analysis made possible by modern computers: using powerful processors to mine through large databases to reveal previously unsuspected or unquantified trends and relationships.

One of the first projects undertaken by Claim Analytics was to create a model to predict return to work for group insurance claimants with long term disabilities. This report, sponsored in part by the Society of Actuaries Health Section, is a case study on creating such a model.

Why Score Claims?

Automated claim scoring:

- Provides a fast, objective, consistent method of scoring (or ranking) approved claims, based on likelihood of recovery.
- Helps claims managers to optimize the allocation of all available resources, and to spend time where it is most productive.
- May be useful in developing more precise reserving and pricing assumptions.

The Model

The model used historic data to study and establish connections between

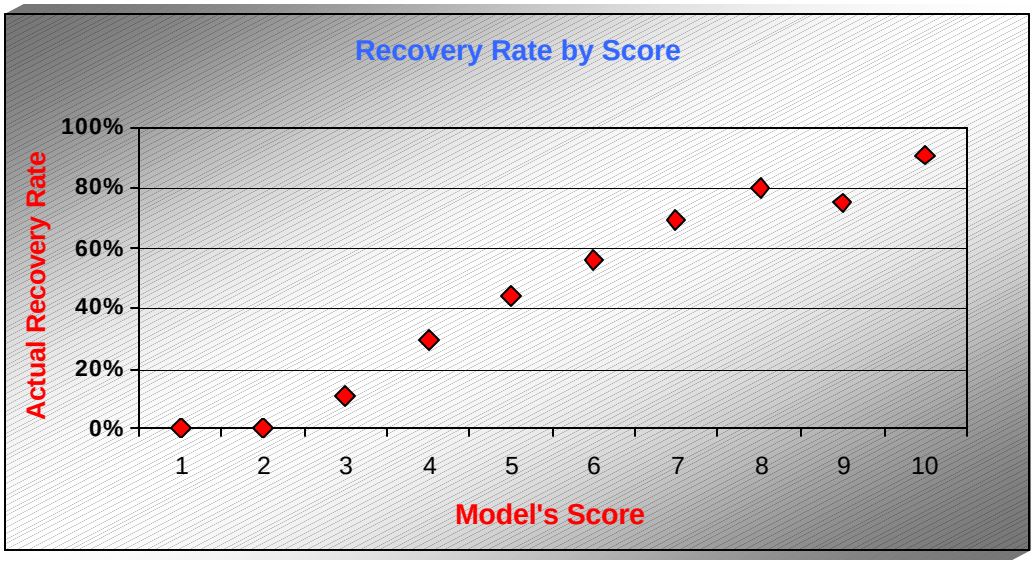
- Data known at the time of claim onset, and
- Whether the claimant returned to work within a stated time.

As the model made these connections, it built an algorithm to predict recovery.
It then scored each claim with a number from 1 to 10: the higher the score, the greater the likelihood of recovery within the stated time period.[1]

[1] We had originally intended to score claims from 1 to 99, but later decided that (i) the extra range of scores would be of little use to claims managers, and (ii) while the model does have predictive value, it would not reach the level of precision necessary to justify separating claims into 99 separate categories.

Once complete, the model was validated by having it make 'predictions' on historic claims. While the outcomes of these claims were known, they were not revealed to the model. In the model's view, these were 'new' claims.

As the chart below demonstrates, the model's predictions aligned very closely with the claimants' real-life return to work behavior. The ability of the model to predict recovery opens a whole new set of opportunities for claims management.



Introduction

Claims managers make daily decisions about how to manage the claims in their portfolio. Should they:

- Order an independent medical examination for Derek T.?
- Provide extensive rehabilitation services to Pat B.?
- Call Jacob Z. again, to monitor his progress?
- Have an investigator check out that suspicious-sounding bad back of Brenda B.?

Good claims managers have keenly developed instincts. They know the ins and outs of claims management. They are keenly aware of 'disability mindset' – the tendency for claimants to give up hope of recovery or to become too acclimatized to being off work. They are eager to practice early intervention in order to prevent its onset. They can boast of an impressive batting average in returning claimants to work. And they do all this without any assistance from decision support tools.

Yet modern decision-support tools can be powerful indeed. They predict weather, they detect fraud, they land jets. It would certainly be of value to develop one to offer decision-making support to claims managers.

Scope

This report offers a case study in how data mining tools can be used to build a model to predict likelihood of recovery for group insurance disability claimants. It includes a high-level summary of the results from modeling a specific set of data.

Definitions

The following set of definitions may be helpful in comprehending the terms and concepts used in this report.

Back Propagation of Error	A supervised learning method for neural networks where the computed error value is passed back through the neural network, leading to adjustments to the weights so as to prevent the same error from recurring.[2]
CART	Classification and Regression Trees. A method to construct a decision tree that is used to classify data based on the answers to a set of binary questions. Also known as recursive binary partitioning.
Data Mining	Using powerful modern processors to mine through large databases to reveal previously unsuspected trends and relationships.
Dataset - Testing	The second of three sets that the claims data should be divided into. As the testing dataset is not used in the training of a model, it provides a way to evaluate the generalization ability of a model. The testing data may also be used to compare and rank several competing models. The testing dataset is typically comprised of about 10% of the claims data.
Dataset – Training	The first of three sets that the claims data is divided into. The training dataset is the dataset on which data mining techniques are applied to train and develop a scoring model. It is typically comprised of about 80% of the data.

[2] Introduction to Neural Networks: Design, Theory and Applications, p. 261

Dataset - Validation	The final of the three sets that the historical claims data should be divided into. Once the final model is trained, the validation dataset (minus outcome information, to ensure a blind test) is passed through the model to ensure its ability to evaluate unseen data. The validation dataset usually includes about 10% of the claims data.
Error	The difference between the outcome predicted by the model, and the actual outcome.
Fitness Function	The goodness of fit measure, selected by the modeler, that the genetic algorithm aims to optimize.
Genetic Algorithm	Inspired by Darwin's Theory of Evolution, genetic algorithms are optimization tools that conduct a global search by comparing, combining and mutating several sub-optimal solutions to create a new – and hopefully improved – generation of solutions.
Gradient Descent	The process of incrementally reducing network error via changes to the weights.[3] Involves calculating the partial derivatives of the error with respect to the weights.
Neural Network	Statistical models of real world systems. Unlike traditional statistical models, yet similar to humans, neural networks learn by example. If the modeler has an idea as to what factors influence the outcome, but not how they do so, neural networks, given available data, are an appealing approach.
Training a network	The process of passing historical data (with known inputs and outputs) through the neural network so that it can detect and learn relationships.
Weights	Parameters

[3] Applying Neural Networks: A Practical Guide, p. 284

Data Mining

What is data mining? Simply, using sophisticated statistical tools to 'mine' through organizational databases to find patterns and trends. Why? Because these discovered patterns and trends can be used to predict human behavior.

The techniques of data mining are very powerful. They harness the speed and capacity of modern processors to the breadth of modern databases. Trends and truths that were previously unrecognized can suddenly come to light.

Using Data Mining for Disability Claims Management

Building A Model to Predict Recovery

Data mining techniques can be used to create a model to predict the likelihood of return to work for any given new claimant. The claims manager can use this information to help choose the best way to:

- Manage that claim.
- Optimize the use of scarce resources to facilitate return to work.

Which data mining techniques could be used to create the model?

We used three data mining techniques to create our model: CART (Classification And Regression Trees), neural networks, and genetic algorithms. We chose Salford Systems CART software to create tree-structured diagrams, and Brainmaker as our neural networks and genetic algorithms package. For information on our software choices, please see Appendix A.

CART

CART creates tree-structured classifications of data. These classifications help determine which factors have the most influence in bringing about a certain result.

In this project, we used CART to reduce the number of influencing factors we would feed into our predictive model. This helps to prevent the model from bogging down in less important information.

Neural Networks and Genetic Algorithms

While we commenced the project planning to use only neural networks, we discovered that genetic algorithms offered significant benefits as well. (More on this later.)

Neural networks and genetic algorithms are optimization tools. Both compare groups of facts to known outcomes, and *attempt to find relationships*. The human brain does something similar in its constant effort to find a link between cause and effect.

Imagine watching the Kentucky Derby and being able to predict which thoroughbred will win by:

- Tracking fifteen bits of data about each horse and jockey ,
- Assigning the databits different weights, and
- Arriving, after iterative efforts, at a non-linear algorithm to predict the winner.

Not likely.

This however, is what neural networks and genetic algorithms delight in. These two tools can measure *subtle* weightings of *many* characteristics, in *simultaneity.*

This is how they uncover relationships that traditional analytical tools fail to detect. It is why they are now used in many applications requiring pattern recognition, diagnosis and prediction. Banks use data mining tools to spot credit card fraud, tax officers to look for inconsistencies in tax returns, law enforcement agencies to detect money laundering, and marketers to analyze sales activity and model buying behavior.

Unlike most mathematical and/or programming tools, neural networks and genetic algorithms can deal well with data inconsistencies – such as one 40-year-old with a diagnosis of fibromyalgia returning to work within 8 months, and another … not at all.

Practical Focus

The focus of this report is practical, not theoretical. Our goal is to show how sophisticated data mining tools – available as packaged software – can be harnessed to improve disability claims management.

The remainder of this report is a case study of how we created a model to predict likelihood of recovery for group insurance LTD claimants.

Following is a one-page summary of the case study.

Case Study: One-Page Summary

Define the Goal

We decided that our model would predict likeliness of recovery within a given time span rather than time to recovery. As well as making this decision, we decided how we would evaluate the completed model.

Data Requirements

We started by trying to get a general idea of which factors influence recovery. Then we had to precisely define recovery – a surprisingly tricky task. We had to decide how many records were needed to create the model, and ascertain that the necessary data fields and records were available.

Data Preparation

We first split the data into three parts. The largest, 80% of the data, was used for training the model. We also set aside 10% for testing and 10% for final validation.

We then checked data quality, and reviewed the data fields with a knowledgeable user. Finally we had to devise a way of transforming raw medical data into quantifiable terms that the model could understand and evaluate.

CART

We used CART as an initial filter to key in on which data factors impact recovery most.

Building the Neural Network

Much more data preparation goes into preparing the data for a neural network than for CART. For example, we modelled 'age' in three ranges, and used fuzzy logic for the edges of the ranges. That is, we put claimants in 3 age buckets – 18-35, 36-50 and 51-65 – but used fuzzy logic on the bucket boundaries so that the model would recognize the similarity between a 35- and a 36-year-old claimant.

Neural Network Training Settings

Choosing neural network training settings is akin to making design decisions for the model. In this section we discuss the basis for which we made our design decisions. Choosing the settings is an iterative process; it takes time and a great deal of trial-and-error testing.

Determining Best Network

The experimentation involved in choosing training settings required training a neural network for each combination of settings selected. We then determined the best settings by comparing the various networks that we trained, using measures such as percentage of records correctly scored, R-squared, etc.

Genetic Algorithms

When we had finished training our network, we decided the results were not adequately precise; too many scores were lumped in the middle. We decided to try the 'Genetic Training Option' of our neural network software.

Genetic algorithms, while similar in function to neural nets, use different methods: neural nets 'learn' from data, while genetic algorithms 'evolve' to a solution. They can be used together, combining the strengths of both (genetic algorithms are capable of global search and not easily fooled by local minima, while neural networks can apply gradient descent to find the minimum point on the error curve). Using genetic algorithms, we produced a model that we felt was ready for final validation.

Validating the Model

We validated the completed model by comparing the model's predictions for the validation data to the real outcomes. The results were excellent. The model's predictions of return to work behavior allied closely with actual outcomes.

Case Study

This case study discusses how we created a model to predict likelihood of recovery for group insurance LTD claimants.

1 Defining the Goal

1.1 The goal

Our goal was to build a model to predict the likelihood that **a group insurance disability claimant would recover within 24 months of satisfying the elimination period**.

The model would assign each new claim a score from 1 to 10: the higher the score, the greater the likelihood of recovery. Our definition of 'recovery' was based on 'return to work.'

1.2 The benchmark

Before starting the project, we defined a benchmark for success. The idea behind the benchmark was that a model that was accurate at both ends of the scale had proven its ability to successfully score claims.

Our benchmark was that:

- 75% or more of claims scored with an 8 - 10 returned to work within 2 years.
- 5% or less of claims scored with a 1 - 3 returned to work within 2 years.

1.3 Why 24 versus 6, 12, or 60 months?

Theoretically, any number of months could have been selected. However, there is little practical value in selecting a horizon greater than 36 months as there are very few additional recoveries beyond that horizon.

We could have modeled likelihood of recovery within a shorter horizon, such as 6 or 12 months, but we preferred 24 months to ensure the data included a large number of recoveries. We also could have modeled the likelihood of recovery for a horizon up to 36 months. However, at 24 months the definition of disability often changes from "inability to perform one's own occupation" to "inability to perform any occupation." We felt different claims management strategies might be used to help claimants return people to

their own occupation rather than any occupation; a 24-month horizon allowed us to identify claimants who would be able to return to their own job.

In practice, it may be advisable to model the likelihood of recovery over 2 different horizons, one short (6 – 12 months) and one long (24 – 36 months), so as to provide an indication of which claimants are expected to recover quickly, which slowly, and which not at all.

1.4 Why not predict *expected time to recovery*?

Why not predict *expected time to recovery* rather than *likelihood of recovery within a specific horizon*? Two reasons: claims that never recover and outliers.

Claims That Never Recover Not all claims recover (some terminate due to death or retirement age), making mean recovery time impossible to define.

Outliers Outliers can have too great an impact on mean recovery time. As an example, say recovery occurs:

- In 20 months 80% of the time,
- In 120 months 20% of the time.

Mean recovery time is then 40 months. The "40 months" is a rather misleading statistic for claims managers. The manager's objective should be recovery within 20 months: recovery *does* occur by then, 80% of the time.

2 Create a Project Plan

A project plan was created to control and monitor the progress of the project. For each step of the project, our plan assigned an elapsed time, number of hours required, person responsible, and cost.

3 Data Requirements

3.1 Description of Dataset

The dataset used for this case study included two years of data comprising nearly 4,000 approved claims from one insurance company. Approximately half of the claims had recovered within two years of satisfying the elimination period.

3.2 Determine the factors that influence recovery

Here we tried to determine which factors might be likely to influence recovery, so that we knew what data we'd like to collect. Research, intuition, common sense, and discussions with claims professionals were all involved in deciding which factors to use in building the model. Note that it is only necessary to identify factors that ***may*** affect recovery – the model itself will decide whether, and/or how much.

Following is a sample of the types of resources we consulted to help us decide which factors to include in our model. This list is not exhaustive.

3.2.1 Published studies

The Canadian Institute of Actuaries produced a study of Canadian group LTD termination that focused on terminations due to recovery *and* death – not just recovery. However, to the extent a factor influences termination, it is likely it also influences recovery. Because the study didn't always specify whether a factor influenced recovery, mortality, or both, we decided to give all the factors consideration.

Their findings:

- Termination rates decrease with increasing age.
- Termination rates decrease with increasing time since disability.
- Male terminations are more likely to result from death (thus less likely to result from recovery) than females.
- Termination rates vary by diagnostic category. That is, high for accidents, low for mental / nervous disorders, low in the early years but high in the later years for HIV / AIDS.
- Risk status: termination rates are lower for ASO business compared to insured business.

- Region: Quebec relative termination rates are very high in the first year and then fall markedly. In the West, it's the opposite; relative termination rates start at a low rate and then increase.

- Termination rates vary by elimination period.

The CIA Group Life Waiver Study: Based on 1988 – 1994 Canadian Group LTD Termination Experience found that:

- Spikes in recovery rate occur at 24 months, both from date of disability and from end of the elimination period. This is related to the change in definition from own occupation to any occupation; some companies use 24 months from disability while others use 24 months from elimination period.

- Recovery rates are low in the first month after the elimination period (relative to age and time since disability)

The Canada Pension Plan (CPP) Experience Study of Disability Beneficiaries found:

- Males have a higher level of recovery – but the gap has shrunk in recent years.

- Insurer-specific claims practices have an effect: CPP recovery rates have fallen in recent years due to more stringent qualification requirements (a reason why there may be differences between companies).

- Insurer-specific claims practices: recovery peaks around the 2nd anniversary – attributed to reassessment activity.

- Diagnosis: more severe disabilities are less likely to recover.

- Age: younger more likely to recover.

- Secondary medical problems: less likely to recover. May be indicative of complex or serious health problems.

- Evidentiary requirements: length of time needed to provide evidentiary requirements associated with higher likelihood of recovery.

Butler, Hartwig and Gardner document several factors affecting Workers Compensation Claims (not the same as disability insurance, as it covers only work-related accidents or diseases, but similar):

- Gender: females tend to file more claims and claims of greater severity.

- Age: older workers have lower claim frequency but more severe workplace accidents.

- Economic condition: both claim frequency and claim severity fall as the demand for labor increases (proxied by Manufacturing Gross Product).

- Moral hazard: claim frequency and duration increase as benefit levels increase.

3.2.2 Advice of experts

Examples of advice we received:

- *Elimination period is important*
  Longer elimination periods decrease recovery rates. One reason may be that longer elimination periods preclude early intervention, allowing "disability mindset" to set in. Another may be that people with milder disabilities tend to recover faster, and may recover before satisfying the elimination period.

- *Labor Union*
  For people who cannot return to their original position, seniority requirements may result in difficulties in younger employees being redeployed to a different job (increasing duration of disability).

  Whether or not a claimant is a union member is information that may be unavailable. However, certain industries and/or occupations do have higher levels of unionization, so modeling 'industry' and/or 'occupation' is worthy of consideration.

- *Employer attitudes*
  Some employers are more willing to make special accommodations to help employees return to work. Employer attitude certainly won't be on the claims extract. However, it is possible that there is some homogeneity of attitudes within certain industries, and so 'industry' may be worthy of consideration. Also, if a large company accounts for a substantial portion of total claims it may be worthwhile to specifically identify that company in the model.

3.2.3 Common sense

We looked at the fields available to the model and tried to imagine a scenario for each field where it would be an important factor in recovery.

As an example, consider the field 'occupation.' If a claimant has arthritic fingers, and her occupation is 'dentist' she will have far more problems with her job than if her occupation is 'negotiator.' Clearly, occupation is an important factor, and should be included in the model.

3.3 What data is needed to determine if a claim has recovered?

It's tougher than it looks. First and foremost, we needed to define recovery.

A full return to work will certainly mean recovery. But consider the following examples:

- Graham gets a negotiated settlement of 12 months full pay. Is this a recovery?
    - If this is a recovery, what is the time to recovery? The date of the settlement? The time to recovery is probably not the date of settlement because it's unlikely Graham has recovered by this time. More likely he is well on his way to a likely recovery, or is unlikely to ever recover. A better choice may be to assume recovery will coincide with the date Graham has been paid through.
- Suzanne appears to have returned to health, but is not returning to work. Her insurer orders an independent medical exam, which certifies that Suzanne, could, indeed return to work. Her insurer cuts her off. Is this a recovery?
- Joan has returned to work part-time, but is unable to resume the weight of her previous full-time job. Is this a recovery?

All of the questions above, and others, must be answered before recovery can be clearly defined. In addition, the definition of recovery will be limited by the amount of available detail on claim termination provided in the database.

We decided to define recovery as any claimant that returned to work, or at least was considered able to return to work. If the insurance company terminated a claim because the medical evidence indicated a claimant was capable of return to work, we included the claim as a recovery - even if the claimant didn't return to work. Instances where a claim terminated due to death, retirement or contractual provisions (such as attaining maximum benefit period) were not considered to be recoveries.

3.4 How many records are required?

There is a tradeoff between the number of records and the number of variables that can be considered. Generally the more there is of the former, the greater can be the latter. (This issue is addressed further on in 'Building the Neural Network'.)

We recommend that the dataset contain a minimum of 500-1000 claims that recovered within the specified time horizon, and 500-1000 that did not. If the dataset is smaller than that, it may be necessary to use fewer variables than we did in our model, and/or to try and categorize claims into fewer scores, i.e. 1-3, or 1-5, instead of a 1-10 score.

As the number of cases of each scenario rises, the model achieves greater subtlety and specificity. The larger the database drawn upon, the more sophisticated and precise can be the results achieved.

3.4.1 How many years of data should be considered?

If you are working with a very large database, with enough records in one year to satisfy the requirements of the model, should you also use data from previous years?

There is a trade-off. Going further back will increase the number of records, and allow observations of recovery rates for different parts of the business cycle (good). However, claims practices may have changed over the years, making older data less relevant (bad). Going back further does permit both modeling recovery over longer horizons, and incorporating the business cycle as a factor influencing recovery.

3.5 Data availability

Some data fields you wish to use will be easily available, some unavailable, and some indirectly available, by calculating from available data. Age at time of disability, for example, can be derived from date of birth and date of disability.

4 Preparing the Data for Modeling

4.1 Split the data

Once the data was received, our first task was to split it into three sets: 'training,' 'testing' and 'validation.'

- *Training*
  The data used to build and 'train' the model, from which the model will learn how to predict recovery. It will usually comprise about 80% of the data.

- *Testing*
  This should be about 10% of the total data. It will be used to test the model once it nears completion. Testing data must not be the same as training data — neural networks can 'memorize,' so they must be tested on unfamiliar data. If the model does not test well enough to meet objectives, the modeler goes back to using the training data, and only returns to the testing data when the model is deemed to be ready for testing again.

- *Validation*
  This should also be about 10% of the total data. It must be kept **completely** separate from the rest of the data – not even looked at – during model building.

  The outcomes for the validation data should be withheld from the modeler. When testing of the model is complete, the modeler scores the validation data.

4.2 Validate data quality

Data quality was checked using a number of different tests.

4.2.1 Automated checking

The following methods of checking data quality can be automated, at least to some degree:

- Blank fields. Any fields left blank are suspect.

- Min and Max. For numeric-value fields (age, benefit amount, dates) we tested minimum and maximum values for each field, to ensure that all values fell within a reasonable range. For example, for the field 'age,' a minimum below sixteen or a maximum above 65 was suspect.

- Relationships. Many of the data fields are interrelated; we checked to see if these relationships were consistent with logical expectations. For example: 'Date Reported' should be later than 'Date of Disability;' 'Monthly Benefit Amount' should be less than 'Monthly Income.'

4.2.2 Manual checking

With a large database manually checking records is impractical to do in entirety, but spot checks are possible. Manual checks rely on applying common sense. For example, if occupation is 'teacher' and monthly income is $550,000, or if diagnostic category is 'mental nervous disorder' but diagnosis is 'colon cancer,' something is askew.

The main purpose of our spot check was to confirm that most records appeared to be correct (i.e. the data fields were consistent with each other). If many records show problems, the integrity of the entire database must be questioned.

4.2.3 What to do with questionable data

Our first step was to identify all of the data problems. Next was to decide which of the four following courses we would take with the problems encountered:

- Retrieve the proper value from paper files
- Estimate the proper value
- Exclude the suspect field from the model
- Discard the affected records.

The ideal next step would have been to correct all data problems by investigating and finding the correct values. Realistically, this may be too time-consuming, and often not even possible.

One solution is to estimate the proper value. Some software packages provide this service, using various analytical methods. For packages that do not, there are methods of estimating the values manually.

For example, say for 5% of the records, the value for the claimant's income is entered as 0. Even if the time is spent combing through paper files to find the actual income, it may be that this value was never even reported to the insurance company. A better approach may be to try and estimate the correct value for these records.

Our data file included the monthly benefit amount and the claimants' occupation. This is plenty of information to estimate income. For example, if the average claimant has monthly benefit equal to 52% of their income, this can be used as an initial estimate. There are some occupations that are very common in the claims database – for these there is additional information to use to refine the estimate.

We were trying to find relationships between several factors over thousands of records – our estimates of the correct values for a few fields on some of the records needed to be close, but not perfect. If a data field was suspect for a large proportion of the records, we excluded it from our analysis. Similarly, we excluded records with suspect data in several fields.

4.3 Understand the data fields

It is worthwhile to review each one of the fields used in the study with someone who is familiar with the database. Not only will there be fields that require explanation, there will be also be fields that appear to be straightforward, but are not. An actual user of the data can explain the quirks and characteristics of each field.

4.4 Transforming raw data for modeling

For data to be useful in a model, it has to appear to the model in a way that is easily measured and evaluated. The way in which data is collected and stored is *not* an exact parallel for the way in which it will be modeled.

Some data preparation decisions are easy. Gender is entered as either 'M' or 'F.' But for other fields – diagnosis, income, even age – a number of questions must be asked and answered before the data can be prepared and the modeling can begin.

The best way to illustrate the issues we encountered in transforming raw data to model-ready data may be by a series of examples:

4.4.1 Examples of transforming raw data

Age at time of disability. This was not always directly available from the data extract. However, given that date of disability and date of birth are both available, age at time of disability can easily be calculated.

Percentage of income replaced. If monthly salary and monthly benefit are both available, replacement income % can easily be calculated.

Did claimant recover within 24 months? This information was not usually directly available from the data extract. However, if date of disability, elimination period, current status, status date and status description are available, the desired measure can be determined.

4.4.2 Modeling 'diagnosis'

There were too many diagnoses (400+ in our dataset) to model each as a unique measure and still enable the model to detect recurring patterns and relationships. As well, many of these occurred only infrequently, not providing enough examples for the model to uncover trends.

We wanted to convert each diagnosis into a series of measures that the model could use to identify and compare how various diagnoses are similar to and different from each other. To clarify, for example:

> *Vision Impairment*
> Intuitively, we expected that claimants with illnesses resulting in vision impairment would be less likely to return to work than claimants with illnesses that did not. We assigned a number, from 0 – 10, to each diagnosis, where 0 indicated the diagnosis as unlikely to result in vision impairment and 10 indicated the diagnosis was very likely to result in severe vision impairment.
>
> *Other Measures*
> We considered other measures for diagnoses in a similar manner, such as:
>
> - Associated impact on fine motor skills
> - Associated impact on gross motor function
> - Likeliness of being treated with drugs.

Here is a sample of how we modeled one diagnosis:

Diagnosis	**Muscular Dystrophy**
Diagnostic Category (categorical variable)	Brain and Nervous System
Impact on Vision	0
Impact on Fine Motor Skills	9
Impact on Gross Motor Function	9
Likelihood of Treating with Drugs	2
Likelihood of Being Fatal	3

Now we were able to present 'diagnosis' to the model as a set of numerical measures. This enabled the model to see many examples of each combination of measures (e.g. terminality high, curability low, vision impact low) and thereby detect patterns.

5 Using CART as an Initial Filter

We used CART (*Classification and Regression Trees)*, as an initial filter to narrow down the number of factors we would use in our model, and ensure that these were the ones most likely to be useful in predicting.

5.1 What is CART?

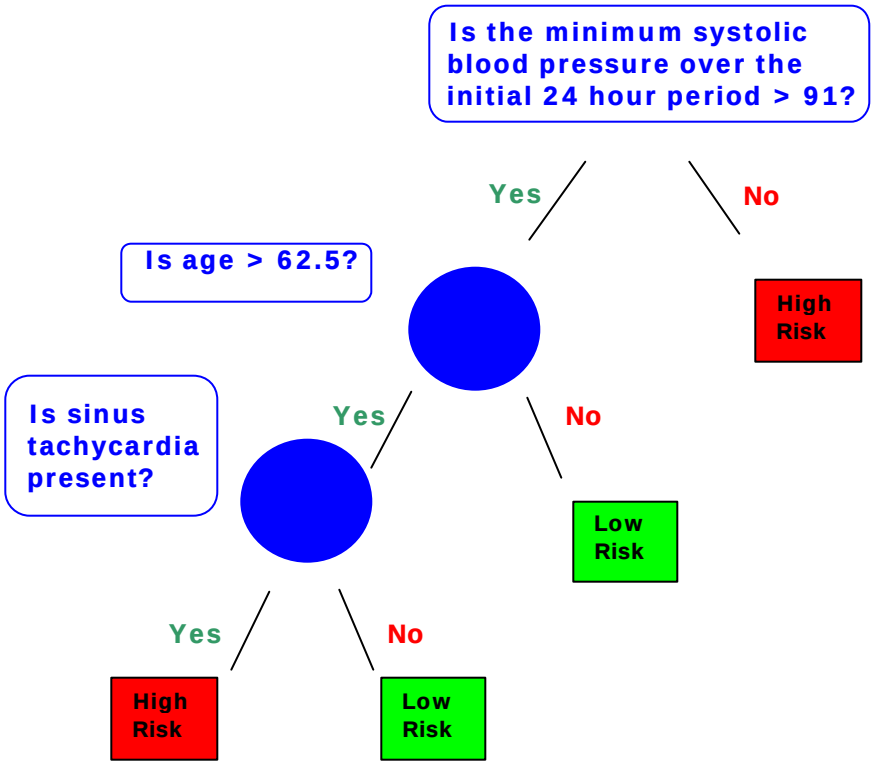
CART produces tree-structured classifications of data. These classifications are used to help determine which factors have the most influence in bringing about a certain result. While CART analysis is sophisticated, the tools it develops are easy to apply.

5.1.1 A CART study

A CART study was performed at the San Diego Medical Center, to see if a better method could be found of evaluating whether incoming heart attack patients were at high risk of dying. The then-current method involved taking 19 separate, often complex, measurements.

The CART study came up with a new system where **only three** yes/no questions were asked. This new system proved to be **more** accurate in predicting high-risk likelihood than its 19-measurement precursor.

CART-derived method of evaluating incoming heart attack patients



For more information on CART, see Appendix B.

5.2 Data preparation

Other than the fact that all data must by represented as numbers (e.g. for gender: female=1, male=2) there was little work to do to preparing the data for CART. NB: this is true for the software we chose, Salford Systems CART; it may not be true for other packages.

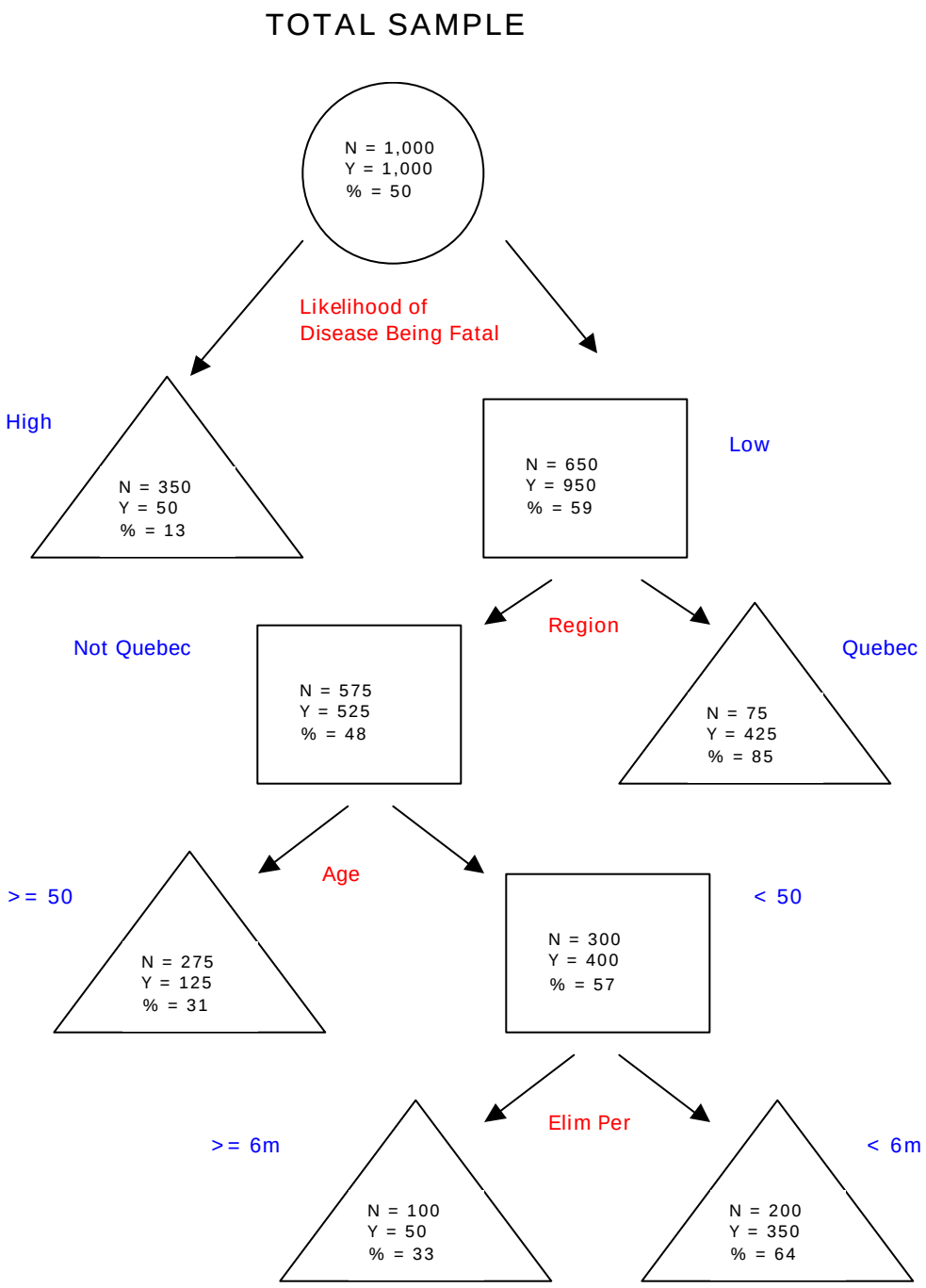
The tree-structured classifications produced using CART are not particularly sensitive to how data is prepared. Unlike other data analysis tools, such as linear regression or neural networks, CART is non-parametric (i.e. it does not determine its output by taking a *weighted sum* of inputs). Instead, CART partitions a dataset into binary groupings based on whether an input is greater than or less than a certain value. Thus the classification trees are unaffected by transformations of the input data.

5.3 Review CART results

We looked at the trees to get a feel for what factors were important (at the top of the tree), less important (bottom of the tree) and not important (off the tree). Interaction can be seen by following along the branches of the tree.

When viewing CART results, if anything looks to be counter-intuitive, it is wise to check the data to investigate. If the data proves to be in order, you may have made a very interesting discovery.

Following is a sample of how a CART tree may look. This tree is not based on real data and is for illustrative purposes only. Inside the boxes, *'N,' 'Y'*, and *'%'* indicate the number of claimants in this category that didn't recover, those that did recover, and the recovery rate, respectively. The square boxes represent nodes that are split further while the triangular boxes represent end nodes.


TOTAL SAMPLE
N = 1,000
Y = 1,000
% = 50
Likelihood of
Disease Being Fatal
High
N = 350
Y = 50
% = 13
Low
N = 650
Y = 950
% = 59
Region
Not Quebec
N = 575
Y = 525
% = 48
Quebec
N = 75
Y = 425
% = 85
Age
>= 50
N = 275
Y = 125
% = 31
< 50
N = 300
Y = 400
% = 57
Elim Per
>= 6m
N = 100
Y = 50
% = 33
< 6m
N = 200
Y = 350
% = 64

5.4 Analysis of CART results

The trees yielded a good deal of information about which factors were the primary and secondary factors influencing recovery. This helped us ensure that we:

a) Included the key factors in our model
b) Avoided the inclusion of less important factors that would introduce noise to the model and slow down run time.

For example, from the sample CART tree shown above we could make several observations. Keep in mind that an actual CART tree will likely have many more branches and provide much greater insight.

- Identifying diseases with high mortality rates will provide valuable information to the model.
- Claimants from Quebec have recovery rates that are much higher than average. Although it seems counterintuitive for one region to have such a different recovery rate, this is consistent with industry data.
- The tree splits on age at only one point (age 50). This may indicate that it is important to identify older (>= 50) claimants from younger (< 50) but less important – or maybe not important at all – to distinguish different ages within these two groups.
- Elimination period is likely of second order importance.
- Data fields that do not show on the CART tree are likely of lesser importance.

5.5 Next step is neural networks

At this point we had decided which factors were key, and should be included in the model. We were ready to begin building the neural network.

6 Building the Neural Network

6.1 What are neural networks?

Neural networks are statistical models of real world systems. Unlike traditional statistical models, yet similar to humans, neural networks learn by example.

6.1.1 Learning by example

Learning by example replaces traditional (if then … else) programming in solving problems.[4] This makes neural networks appealing for applications where:

- The rules are *not* known but a set of examples is available, from which a solution can be learned[5]
- The rules are not hard-and-fast – i.e. in the case of claims, a disability that keeps one 40-year-old worker away from her job for only 8 months, keeps another, with the same job, at the same place, away for two years.
- Non-linear relationships.

If you have a sense as to *what factors* influence the outcome but not *how* they do so, neural networks, given available data, are an appealing approach.

6.1.2 Uncovering unsuspected patterns

Neural networks uncover patterns and relationships that traditional analytical tools fail to detect. Traditional statistical methods pre-date automobiles, telephones, electric light, and, most importantly, electronic computing. They were developed under the constraint that they could involve only the number of calculations that could be performed by humans in a reasonable amount of time.

Neural networks were developed during – and inspired by – the electronic computing era. By harnessing our newfound power to make billions of calculations per second, they have allowed a new, more intensive approach to data analysis.

Many characteristics, in simultaneity

Neural networks can measure *subtle* weightings of *many* characteristics, in *simultaneity*. Turn to neural nets when you need to simultaneously and objectively:

- Analyze and correlate a number of different qualities,
- Evaluate their collective relationship to the end result, and
- Establish a weighted *non-linear* algorithm to predict.

[4] Fundamentals of Artificial Neural Networks, p. xiii
[5] Applying Neural Networks: A Practical Guide, p. 9

For a pictorial representation of neural networks, please see Appendix C.

6.2 Building a predictive model

To predict recovery, the essential capability the model must achieve is to distinguish claims that *do* recover from those that *don't*, based on claim characteristics. Once it can do this, it has the functionality required to score claims.

For each claim, the model applies a set of weights against input data factors, and estimates the likelihood of the claim recovering. The prediction is then compared to *what actually occurred*, and refined. Over many iterations, the model finds the set of weights to optimize how well its predictions match actual experience.

6.3 Preparing the data

For best results, most data fields should not be entered into the model exactly as they are received. They should be entered in a way whereby the model can best quantify and compare them.

Two general types of variables are used in the model:

Ordered A variable is called *ordered*, or *numerical*, if its measured values are real numbers (e.g. age or income), with a natural graduated order – $35,000 per annum is less than $50,000 per annum. Here the variable is represented as its actual value.

Categorical A variable is *categorical* if it takes values in a finite set not having any natural ordering (e.g. region or occupation class). Here we created several binary variables (one for each category), one with value 1 and the rest with value 0.

Note that one cannot set up just one variable that can take several values (one for each class), as the neural network will treat the input data as ordered. For example, if there are three occupation classes (executive, management, staff), we cannot represent this as one variable with a data field that takes on a value of 1, 2 or 3, as the neural network will think $1 < 2 < 3$. Instead, we must create 3 separate binary variables, each representing the claimants' membership in executive, management and staff, respectively. A claimant in management would have '0' as the value in the executive variable, '1' for management, and '0' for staff.

6.4 Data modeling techniques

The following is a discussion of some of the techniques that were used in preparing data for modeling.

6.4.1 Ranges

For some applications, you may get *better* results for numerical variables by having values correspond to ranges.[6]

Rather than representing age as one number (actual age), we decided to:

- Classify a claimant into one of 3 age categories (young, middle, old) and,
- Treat it as a categorical (unordered) variable.

Consider the cases of 21-year-old Ashley, 37-year old Bruce, and 53-year old Claire:

Their scaled ages (the scale maps 18-65 to a scale of 0-1) would be 0.06, 0.40 and 0.74, respectively:

- But when converted to the ranges of young, middle, and old:
    - Ashley gets 1 for young, 0 for middle, and 0 for old
    - Bruce gets 0 for young, 1 for middle, and 0 for old
    - Claire gets 0 for young, 0 for middle, and 1 for old
- As a result, instead of the difference in scaled age from Ashley to Bruce being 0.34 it is now a difference of 1 (on the young range), 1 (on the middle range) and 0 (on the old range).

The two differences of 1 on the young and middle ranges are much more meaningful to the neural network than the one difference of 0.34.

6.4.2 Fuzziness

Fuzzy logic is useful for modeling data groups with blurred boundaries. As an example of using fuzziness to deal with blurred boundaries, let us again consider the factor of 'age.'

Assume age is split into three categories: young (18-35), medium (36-50), and old (51-65). In this case the model will see the difference between an 18-year-old and a 50-year-old as identical to the difference between a 35-year-old and a 36-year-old.

Fuzziness alleviates this problem by allowing a value to be represented as partially of one set and partially of another. Using the example above, age 35 could be represented as 50% 'young' and 50% 'medium.'

Note that by using both ranges and fuzziness, we essentially put the age factor into three fuzzy buckets. We used buckets to ensure age differences received their due weight, and fuzziness to deal with any problems caused by the arbitrary boundaries between the buckets.

[6] Brainmaker Professional User Guide, p. 8-22

6.4.3 Scaling

Imagine a school report card where English was graded out of 1,000 but every other subject was graded out of 100. When the student's average was computed, the mark for English would carry undue weight. Scaling (AKA normalization)[7] of a dataset ensures that every variable operates *within the same range,* so that each variable contributes *the same proportion* to changes in network weights.

For example, there may be a situation where one variable covers a much larger range than another variable. Range of income might be $ 0 → $750K, whereas range of replacement income ratio could be 0 → 1.

In such cases, it is desirable to *scale* the input data before training. If not, the errors from the higher-valued variable will have a greater effect during training than those due to the lower value - their magnitude will be greater.

6.5 Data preparation examples

The following are examples of how we prepared a number of fields for the model.

6.5.1 Diagnostic category

Possible values for diagnostic category included 'malignant neoplasms,' 'musculoskeletal' and 'mental / nervous disorders' (we had 17 categories in total).

Representation may seem simple: create one binary variable for each possible diagnostic category (DC). But – this won't do. Neural nets learn by example. Some DCs are relatively rare. For these, there may not be enough examples from which to learn.

As well, there is a limit to how many variables one can include: the number of relationships that can be found is limited by the number of training examples available. If there are only 100 training examples, the network is unlikely to find useful relationships among 2,000 variables!

Our Approach: Rather than create a variable for each DC, create one for each of the most frequent DCs and lump the rest into a category called "OTHER."

[7] Applying Neural Networks: A Practical Guide, p. 31

6.5.2 Age

Clearly a numerical variable (range 18 – 65) but, through experimentation, we found results were better when age was represented as a categorical (and fuzzy) variable.

Our Approach: Represent as a categorical and fuzzy variable.

6.5.3 Region

There could be one category per state or province, but some states have few claimants; also, we didn't want to devote as many as 50 values to this variable.

Our Approach: Group provinces and states into a number of regions.

6.5.4 Income

Outliers can create major problems. While 92% of our claimants had an annual income between 20 – 100K, the maximum was 750K. If the scale is from 0 – 750K, then the difference between 20K and 100K appears quite small to the model (11% of the total range) so it is difficult for it to see the difference between these two values.

Our Approach*:*

- We considered using the natural log of salary. However the difference between Ln(20K) = 9.9 and Ln(100K) = 11.5 is 32% of the total range – an improvement but possibly not good enough.
- We considered omitting the few records that are outliers. This gets rid of the outlier problem but results in throwing out all records with high salaries.
- In the end we decided to categorize salaries (e.g. 0 – 30K, 30 – 60K, 60-100K, >100K) rather than model as a numerical variable as this approach tested the best.

6.5.5 Determining best data representation

As has been mentioned, the above are guidelines, but not rules, for determining the best way to represent data.

As a result, it is necessary to experiment and find which results in the best network.

6.6 Training the network

6.6.1 Choosing the settings

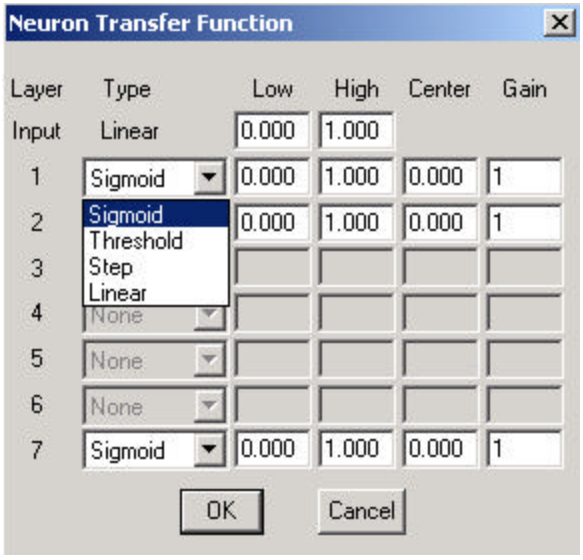
6.6.1.1 Introduction

Neural network software packages vary in how much design latitude they offer the modeler. Brainmaker, which was the software we used, offers a fair amount of latitude, and requires the modeler to make a number of decisions about the design of the neural network. The documentation refers to these as *training settings*.

There is no way of knowing in advance the best value for the settings – experimentation is required – but there are some rules of thumb to help determine the best range to test. We will cover how we dealt with the main settings in detail. Screen-shots from the Brainmaker software we used are provided for reference.

NB: The setting choices described below are made in preparation for training the model. Once these choices have been made, it is time to hit the 'RUN' key and start to experiment.

6.6.1.2 Choose activation transfer function



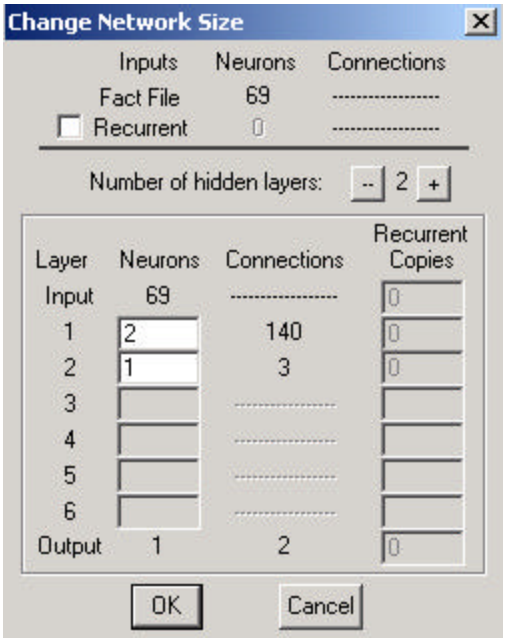
The input into the activation transfer function is always a weighted sum. The job of the activation transfer function is to squash – compact – the range. The following example assumes the modeler has chosen the Sigmoid Transfer function as the activation transfer function.

1. With two inputs: (1, 4) and two weights: (-1, 0.5) …
2. Our weighted sum would be (1 x -1) + (4 x 0.5) = -1 + 2 = 1.
3. ATF is sigmoidal ➔ output value = 1 / (1+exp (-weighted sum)) = 1 / (1+exp(-1)) = 0.731.

The Sigmoid Transfer Function, of those offered in Brainmaker, is the obvious choice for this project. The Brainmaker documentation recommends using the other options they offer only for experimentation. Moreover, the back propagation algorithm works quite well with nonlinear, continuously differentiable functions, such as this, and outputs a value between 0 and 1 (think of 0 as no recovery, 1 as recovery, and everything in between as varying degrees of likelihood of recovery).

Another commonly used transfer function is the hyperbolic tangent, which squashes the output into the range from -1 to +1. However, this activation function is not available in Brainmaker. Moreover, an output range of 0 to +1 seems much more natural for modeling recovery.

6.6.1.3 Setting – Network size



This is likely the most critical setting.

This refers to the number of layers and the number of weights in each layer. This in turn determined how many weights (parameters) there were in the equation.

There is a trade off between accuracy (more hidden layers) and ability to generalize (less hidden layers).[8]

[8] Applying Neural Networks: A Practical Guide pp. 51 - 52

The weights are simply parameters. To score a new claim, the neural network takes the cross product of the *parameters* and the *claim data* and then squashes the output using the activation transfer function. The neural network discovers patterns in data using back propagation and the weights are simply a summary of the patterns that were detected – i.e. the weights describe the data. More weights provide a more detailed description of the patterns while less weights provide a more general description of the patterns.

More weights are required for classification tasks with more complicated patterns. However, too many weights will lead to over-fitting. The neural network will "memorize" the training data – and classify it very accurately – but classify new data very poorly, as it will not generalize well.

The Brainmaker Professional User Guide advises:

- In general, the *more facts* [training examples] and the *fewer weights* you can use, the better.

- The minimum number of weights should be equal to the # of facts, divided by 10, minus the number of input weights, minus the number of output weights:
  [# of facts / 10] – Input weights – Output weights.

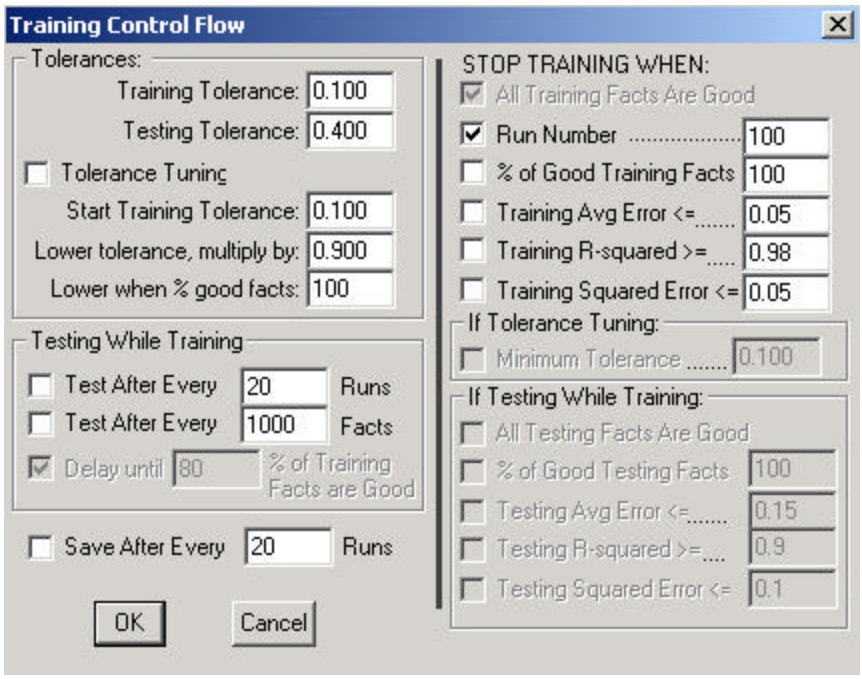
- The maximum number of weights should be equal to the number of facts, divided by 2, minus the number of input weights, minus the number of output weights:
  [# of facts / 2] – Input weights – Output weights.

The above guidelines provide a range for the vigorous experimentation that is a sine qua non of building a neural network.

With 2088 facts, 70 inputs, and 1 output, our minimum recommended number of weights was 139, and maximum 973.

Given that the fewer weights used, the better, and that the minimum recommended weights for our model was 139, our final model achieved a number of weights very close to optimal, at 145.

6.6.1.4 Setting – Training tolerance



Training Tolerance: How accurate the neural network output must be to be considered correct.

During training, the neural network cycles through the data one record at a time. Each pass at a record, the network compares the predicted output to the actual output and adjusts its weights to better approximate the actual output. If the difference between the predicted and actual output is *within* the error tolerance, the network *does not* adjust its weights (it has now passed the test), and moves on to the next record in the cycle.

"Tolerance" is defined as a percentage of the range of the output. For example, we were measuring likelihood of recovery on a scale of 0 (no recovery) to 1 (recovery) – so the range of the output was 1 (1 – 0 = 1).

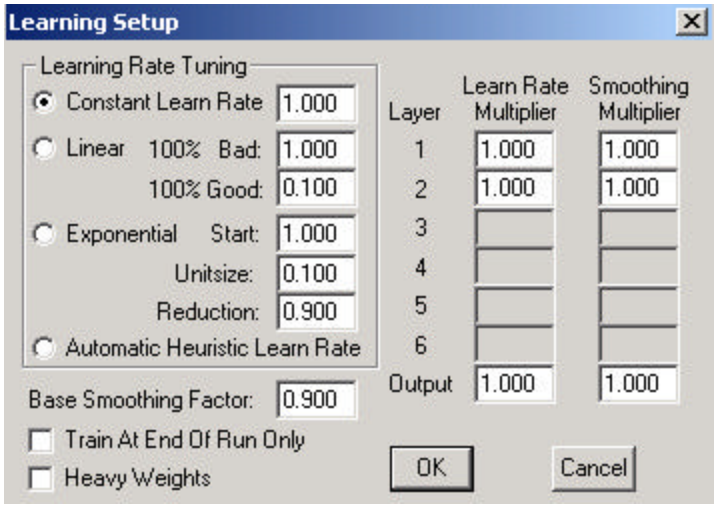
If tolerance is set at 0.1 and the *actual* output is 1.0 (recovery) then the network will not adjust its weights if the *predicted* output is between 0.9 and 1.0, because the difference is within the tolerance. This can also used as a "stop rule" for training: when x% of the training data has been predicted within the range of tolerance – stop training. However this is not an appropriate "stop rule" for this application because it is very unlikely that model will predict a large proportion of the training data within a reasonable tolerance.

It would be ideal to be able to achieve a training tolerance as low as 0.1. However, there will inevitably be contradictions in the training file (claimants who have very similar profiles but different outcomes). As a result it is unlikely that the model will be able to achieve a training tolerance as low as 0.1 on a large proportion of the training sample.

6.6.1.4.1 Tolerance tuning

Brainmaker has a related feature to training tolerance: *tolerance tuning*. This will automatically decrease the training tolerance once a specified % of the training data is accurately predicted (within the training tolerance) by the current state of the neural network that is being trained.

6.6.1.5 Setting – Learning rate tuning



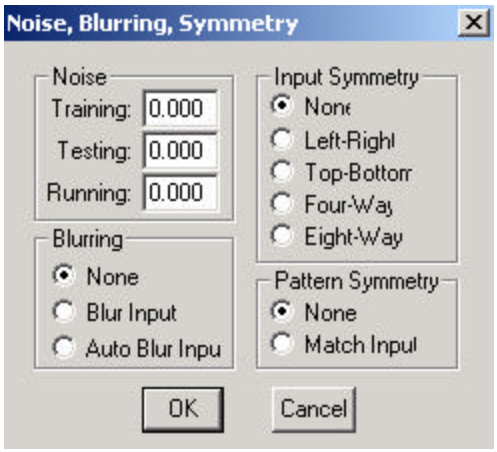
In the standard learning algorithm (i.e. back propagation), the learning rate specifies how large a change in the network weights should be made during training when there is a network error (significant difference between the model's prediction and the actual outcome).[9]

A large value for the learning rate will increase the learning speed, at first, but will also increase the risk that the network will have difficulty descending the error curve to the minimum point as it oscillates back and forth around the minimum taking too large a step each time.[10]

A small value for the learning rate slows down the learning speed and also increases the risk of becoming trapped in local minima. Learning rate tuning allows the learning rate to be decreased as learning progresses. There are several different algorithms for reducing the learning rate that can be experimented with.

[9] Brainmaker User Guide, pp. 10-20
[10] Applying Neural Networks: A Practical Guide, p. 64

6.6.1.6 Setting – Noise



The noise setting allows the system to add random fluctuations to each input value every time a training set (claim record) is presented to the network.

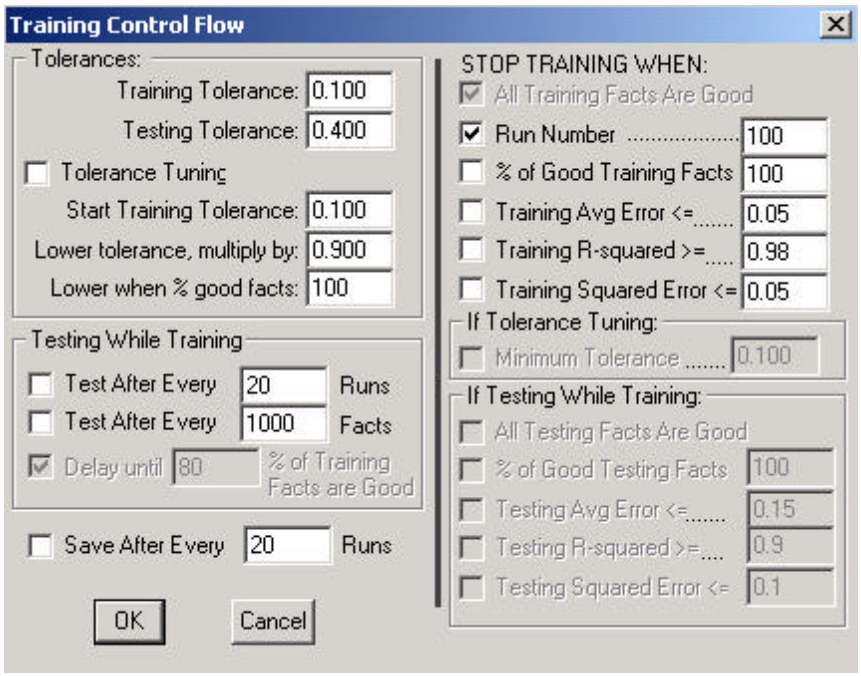
Adding noise can help prevent the network from "memorizing" the training set. The more noise you add, the more general your network becomes.[11]

However, if there is too much noise, the network may not learn at all – the best noise value makes it difficult, but not impossible, for the network to learn.[12] Noise will increase training time.

One option is to delay adding noise to network training until the preferred values for the other settings are determined.[13] Adding noise may help when there is a small training dataset.[14]

[11] Applying Neural Networks: A Practical Guide, p. 60
[12] Introduction to Neural Networks: Design, Theory and Applications, p. 201
[13] Brainmaker Professional User Guide, pp. 10-29
[14] Brainmaker Professional User Guide, pp. 3-22

6.6.1.7 Setting – Stop training rule



It is very unlikely the network will reach a point where it accurately predicts (within a reasonably small training tolerance) the correct output value for a large proportion of the training data. The data has too many contradictions – similar claims with different results – for the neural net to reach an adequately high level of accuracy.

Brainmaker offers a few other options:

- Training Average Error less than a specified amount
- Training R-squared bigger than a specified amount
- Training squared error less than a specified amount
- Pre-specified # of runs (i.e. cycles through the data).

The first three of these options are probably not useful for one reason. There are limitations to how well the network can train.

This leaves only one option: a pre-specified number of runs. How many runs is that best number? Experiment!

6.6.2 Determining best settings

There are no hard and fast rules for determining the best network settings, only guidelines. As a result, it was necessary to experiment with a wide range of both, to find which resulted in the best network.

First, it was necessary to select the activation transfer function. We found it worthwhile to perform this process multiple times (for multiple ATF's) and then to compare the resultant networks to see which was optimal.

Network size is probably the most critical of the network settings. To start, we kept the other settings constant (at reasonable values) and experimented to find the optimal network size. Then we continued to experiment to find the best combination of training settings.

Once Brainmaker has completed training a neural network, the result is a set of stored weights that can be applied to score data on new claims. The weights are available for viewing, but looking at the weights does not provide any insight into the patterns the neural network has uncovered.

Scoring new claims is very simple with Brainmaker:

- Load the neural network into Brainmaker.
- Highlight and copy the claims data from where it has been stored (e.g. an Excel spreadsheet).
- In Brainmaker, under the Edit menu, select Paste.
- Brainmaker automatically runs the copied claims through the loaded neural network, and then returns the resultant scores to the Windows clipboard.
- Return to Excel and select the cell just to the right of the last field of the first claim, and click Paste.
- The score for each claim will appear in the furthest column to the right.

6.6.2.1 Multiple Network Settings

The experimentation to determine best settings was one of the most demanding and time-consuming aspects of building the model. It required intensive effort, a flexible approach, and the commitment to going through the same process over and over again, with slightly different modifications each time. At this point, building the model becomes both art and science.

Each combination of training settings considered required training a neural network – so that we could determine the best settings by comparing the various trained networks. Only by training many neural networks with varied settings could the combination of settings that results in the best one be determined.

6.6.2.2 Multiple Networks for Each Setting

There is a random element in training a neural network. Training multiple networks using the same initial weights and network settings will result in different outputs. One way of coping with this randomness is to generate multiple networks with the same

settings – but different initial weights. Then, when comparing the various network settings, consider the average of the ranking measures (as described below) for each of the various network settings.

6.7 Ranking the networks

When comparing different networks, how did we rank them? The following were several measures that we used:

- The percentage of records correctly scored (within the training tolerance)
- Average error
- R-squared
- Root mean squared error.

6.7.1 Using the testing data

Should the ranking measures described above be used on the *training* data or is it preferable to pull out the *testing* data? We recommend using the testing data: we were more interested in how the network would fare when evaluating data it had not yet seen, and could not possibly have "memorized."

6.8 Choosing the best network

Once the best settings have been determined, the next step is to:

- Select the best neural network.
- Document what the best settings were.

Once this step was completed we found performance improved if, instead of using the best network (i.e. the one with the highest ranking using the ranking measures), we used a "committee of experts" approach where we trained eight networks with the same settings but different initial weights, then ran the testing data through all of the networks and took the average score.

6.9 Neural network results

When we had finished training our neural networks, we were not sufficiently satisfied with the results to proceed to the validation step, as too many of the claims were lumped in the middle, with scores between 4 and 7, and too few at either extreme.

Our neural network software, Brainmaker, offered an optional module titled: “Genetic Training Option.” The end result looked like a neural network, but with a different way of determining the weights. The same data input file as was used for training the neural network could be used for this option.

We decided to see if using the genetic algorithms options would make the predictive powers of the model more acute.

7 Genetic Algorithms

Genetic algorithms are similar in function to neural networks, and are assigned similar tasks. Both are optimization tools that try to find optimal weights for a specific function.

However, neural nets and genetic algorithms use different methods to arrive at a solution. Neural networks 'learn' from data, and attempt to build on the knowledge to improve results. Genetic algorithms, true to their name, 'evolve' to a solution.

7.1 Evolution vs. gradient descent

The *evolution* of genetic algorithms occurs in a 'survival of the fittest' environment. Multiple sets of weights are compared and considered in simultaneity. The sets of weights which best describe the data are allowed to 'mate' with each other.

By mating, we mean exchanging information (weights) – to generate new strings. This process is repeated with the new generation of strings. In essence, only the 'fittest' strings are allowed to survive.

Neural networks, by contrast, learn by *gradient descent*. They have a stronger ability to work towards an optimal solution, but can get bogged down in local minima, pursuing the best rendition of a sub-optimal solution.

Genetic algorithms have less strength in finding the optimal solution, as they are less efficient at crawling down the error curve. But they are capable of a *global* search, and are not easily fooled by local minima.[15] Their more random approach insures a greater likelihood of *discovering the best route* to the solution.

7.1.1 Using genetic algorithms and neural networks together to improve results

Hassoun[16] recommends using genetic algorithms to search the weight space of a neural network with a predefined architecture. That is, once you determine the activation transfer function and the best network size, use a genetic algorithm to find the weights, rather than a neural network.

The advantage of this approach is that genetic algorithms are capable of global search and not easily fooled by local minima.[17] The Brainmaker manual for the Genetic Training

[15] Fundamentals of Artificial Neural Networks, p. 452
[16] Fundamentals of Artificial Neural Networks, p. 452
[17] Fundamentals of Artificial Neural Networks, p. 452

Option recommends using the weights determined by the genetic algorithm as the initial weights for a neural network, and then using gradient descent to further improve performance.

This approach combines the strengths of both genetic algorithms – capable of global search and not easily fooled by local minima – and neural networks – applying gradient descent to find the minimum point on the error curve.

7.2 Genetic algorithms: basic concepts

7.2.1 Fitness function

The fitness function is a measure of how well a string (set of weights) describes the training data. Genetic algorithms use an optimization algorithm: they try to find the string that results in the fitness function being maximized.

Unlike neural networks – where only one training example is considered at a time –with genetic algorithms the fitness function is measured by considering the entire training dataset at once. An obvious choice for the fitness function would be a standard goodness of fit measure, such as R-squared or (1 – average error).

One limitation of Brainmaker's 'Genetic Training Option,' we found, is that it offers a very limited number of fitness functions. To overcome this limitation, we wrote our own genetic algorithms program. We were still able to use the same data input file as was used for Brainmaker.

7.2.2 Selection

Survival of the fittest is the basis for the selection process. It determines which strings will "mate" and pass their genetic information (weights) on to the next generation.

This is a stochastic process, where the likelihood of a string being selected is related to its relative value in the fitness function, that is, those strings that result in a higher value for the fitness function are more likely to be selected to mate.

The standard 'selection' choice is the *roulette wheel method.*[18] With this method:

- A string's probability of being selected is equal to the quotient of its fitness level divided by the sum of the fitness levels of all strings. There can be problems when one string is far superior to the rest of the population; because there is not enough genetic diversity, the other strings are very unlikely to get selected.

[18] Fundamentals of Artificial Neural Networks, p. 440

- Another way to do the roulette wheel is based on *rank* rather than *fitness* level. If the population size is 10, the fittest string is given rank 10 and the least fit string rank 1. The probability of selection for any string is equal to its rank divided by the sum of all the ranks.

7.2.3 Crossover

The crossover function is based on natural genetics. It determines how two selected strings will share their "genetic" information to generate a new (child) string. One way is to take some weights from one of the parent strings, and the other weights from the other parent string. Alternatively, one can also take a linear combination of the weights from both parents. This should be a stochastic process.

Crossing Sites

A common algorithm is to randomly select a *crossing site,* a point on the string where all genes (weights) to the left of the point are from one parent and the rest of the genes are from the other parents.[19]

Multiple crossing sites can be selected. The process begins with taking genes from one parent. At the first crossing site this stops, and genes are taken from the second parent. At the second crossing site, this stops, genes are taken again from the first parent, and so on.

Below is a representation of the crossover function with one crossover point.

Below is a representation of the crossover function with multiple crossover points.

[19] Fundamentals of Artificial Neural Networks, p. 441

Alternatively…

We did something a little different. For each gene in the child, we randomly (50/50 probability) took either the gene from the first parent or from the second parent. The crossover operator is the most crucial in obtaining good results – it is responsible for mixing the partial information contained in the strings in the population.[20]

7.2.4 Mutation

Mutation is also based on natural genetics, and should also be stochastic.

When a gene (weight) mutates, the value of the weight is changed. The value of the weight could change to a new, random value, or be changed by a random amount with a pre-defined distribution.

The purpose of mutation is to diversify the search and introduce new strings into the population in order to fully explore the search space.[21] Mutation should be applied with a small probability, to avoid destroying the highly fit strings in the population.[22] We used a mutation rate of 3% (a little higher than recommended in the book):

- 1.5% mutation rate to a random value. This adds diversity, by trying something completely different.
- 1.5% mutation rate to a random change in value (the size of the change was uniform from -0.5 to +0.5). This is a type of fine-tuning – can slightly adjusting a weight slightly improve the network?

7.2.5 Elitism

In using crossover and mutation, the best string may be lost. Elitism is a method which first copies the best string to the new population. It insures the best string from any generation is not lost, and can improve performance.

7.2.6 Population Size

Population size refers to the number of strings in a generation.

The trade-off is between having too few, and thereby exploring only a small part of the search space, and having too many, which decreases search speed. There are marginal diminishing returns to increasing population size.

[20] Fundamentals of Artificial Neural Networks, p. 442
[21] Fundamentals of Artificial Neural Networks, p. 443
[22] Fundamentals of Artificial Neural Networks, p. 443

We tried population sizes of 16 and 32. While 32 strings converged to a solution in fewer generations, each generation took longer to run. Final run times and results were very similar.

7.2.7 Stop Rule

The stop rule determines when the genetic algorithm will stop generating new strings. At this point, the fittest string from the final generation is the final solution.

A common approach would be to stop if the maximum fitness level does not improve by a specified amount over a specified number of generations (i.e. if r-squared does not increase by at least 0.005 over 25 generations).

7.2.8 Summary: General genetic algorithm

Following is a summary of the steps involved in building a genetic algorithm. This summary assumes that the problem has been defined, the data prepared, and the network size determined.

I. Define a fitness function
II. Define how many strings should be in the population (n)
III. Create initial population, using small, random weights
IV. Calculate the fitness level for each string in the population
V. Create new population
    a. Elitism: most fit string is replicated in next generation
    b. Repeat the mating exercise (n-1) times
        i. Based on fitness levels and selection algorithm, select a pair of strings to mate
        ii. For each mating pair perform crossover
        iii. For each mating pair perform mutation
VI. If the end condition is met, STOP – select the most fit string as your final set of weights
VII. If the end condition is not met, go back to step IV, using the newly generated population.

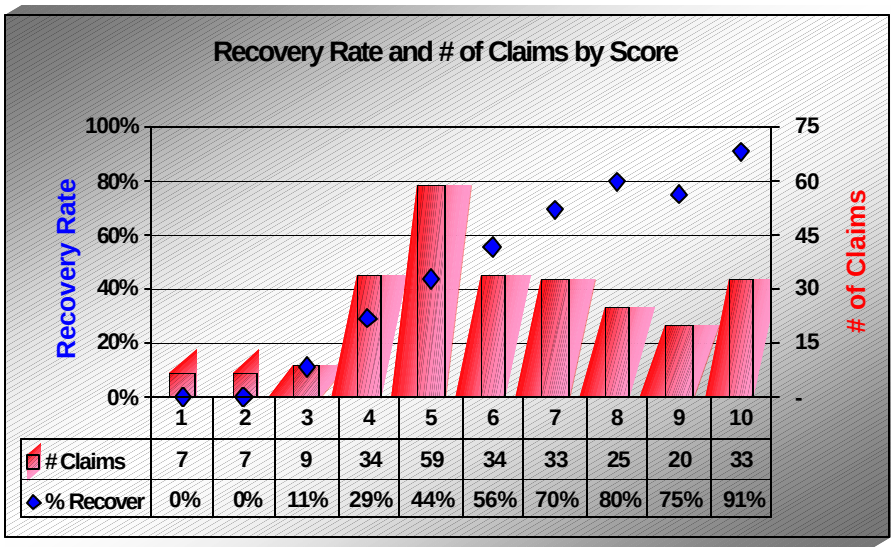
7.2.9 Results: Genetic algorithms

We had immense success with genetic algorithms, particularly when we used non-differentiable fitness functions to improve performance. Many variations have been developed for genetic algorithms, which decrease run time and/or explore more of the search space. These variations, however, are beyond the scope of this report.

8 Validating the Model

To measure our success, we simply compared the predictive scores the model had assigned to each claimant to what had actually happened.

We found a high correlation between the assigned score and the incidence of return to work. As indicated by the rising line of blue diamonds below, we had successfully created a predictive model.



8.1 Measures of Success

8.1.1 Benchmark

We looked at scores from 1-3 and 8-10 in aggregate, to see if the model had met the benchmark we had set:

- 75% or more of claims scored with an 8 - 10 returning to work within 2 years.
- 5% or less of claims scored with a 1 - 3 returning to work within 2 years.

It did:

- 83.3% of claims scored as an 8, 9, or 10 returned to work within two years.
- 4.3% of the claims scored as a 1, 2, or 3 returned to work within two years.

8.1.2 The Gray Area: Scores from 4 to 7

We also looked at the steepness of the change in return to work from scores numbered 4-7. We were very happy to see the consistent rise in this area, correlating a higher score with greater return to work. We were also pleased by the marked aggregate difference from bottom to top: **30%** of those we scored with a four returned to work within two years, as opposed to **70%** of those scored with a seven.

Being able to separate the slightly worse than average from the slightly better than average claims is very difficult for a human to do. It is a great deal more difficult than recognizing claims that are either end of the scale, very promising or very unpromising.

The ability of the model to clearly differentiate between 'gray area' claims may be one of its greatest attributes for claims management. Claims managers could use the model's scoring to then focus scarce resources and attention on those claimants in the gray area (6's and 7's) who are most likely to recover.

8.1.3 Summary of validation results

Our model did an excellent job of identifying claims with high potential for recovery:

(i) Aggregate recovery rate of the 8's-10's was over 80%
(ii) This high recovery rate was *not* simply because the model selectively gave high scores to just a very few claims (taking only the best of the best). Nearly half of all recoveries received a score in this range.

We were only slightly less successful in identifying low-potential claims (fewer claims were scored 1-3 than 8-10). However, those claims identified as poor did have a very low recovery rate (less than 5% - which is relatively lower than 83% is high).

One criticism that might be leveled at the model is that there were, proportionately, too many claims scored as a 5 or a 6. However, this is likely due to the scores being normally distributed. As well, it should be noted that the model showed both a definite difference between 5's and 6's, and consistent differences throughout the 4-7 range.

9 Model Limitations

A few limitations of the model should be noted:

- Claims management practices influence claim outcomes. Thus the patterns the model detects in the claims history are influenced by the practices of the company from which the data is drawn.
- The scoring system cannot be used to make payment decisions on specific claims, or to in any way replace well-trained professional claims personnel.
- The model cannot be used to make financial decisions, such as changes to the reserve basis, without additional testing and actuarial analysis.

10 Conclusion

We were delighted with the results of our data mining project. Our model proved itself able to accurately score claimants and predict recovery.

Scoring of disability claims can provide claims departments with a valuable tool to help with claims management.

Appendix A: Software Selection

In selecting software our bias was to choose software we had used in the past and with which we were already familiar. We did not undertake an exhaustive review of available offerings.

Classification Tree Software

There are a number of CART packages available at very low prices. However, some of these packages tend to produce code only, and not draw the actual decision trees.

This is a major drawback. It slows down the analytical process considerably, and makes learning more difficult. The software package we chose, Salford Systems' CART, does draw the decision trees. Additionally, this software:

- Uses the original CART code
- Is used by many large and established organizations around the globe.

Salford Systems' CART

Salford Systems' CART is the only decision tree system based on the original CART code developed by Stanford University and University of California at Berkeley statisticians; this code now includes enhancements that were co-developed by Salford Systems and CART's originators.

Applications

Industries using CART include telecommunications, transportation, banking, financial services, insurance, health care, manufacturing, retail and catalog sales, and education. Applications span: market segmentation, customer profiling, retention/attrition analyses, market segment profitability, campaign targeting, response prediction, credit card scoring, fraud detection, quality control, and biomedical research.

CART has been used and praised by senior managers at:

- Sears, Roebuck
- AT&T Universal Card Services
- Fleet Financial Group
- Chevron Information Technology
- A number of government and academic institutions.

Alternatives

Two alternative decision tree software packages are See5 from Rulequest Research and SPSS AnswerTree from SPSS Inc. We do not have experience working with either of these packages so we cannot comment on their quality. Our impression is that See5 does not provide drawings of the decision trees (only the rules) but that SPSS AnswerTree does. Both of these packages are much less expensive than Salford System's CART. See5 is currently priced at US$840, SPSS AnswerTree at US$1,495. CART starts at US$4,995 for a three-year license.

Neural Network Software

There are several neural network software packages available. Other than BrainMaker (from California Scientific Software) the only package we had previous experience with is BackPack (by Z Solutions). We chose BrainMaker for two reasons:

- It provided more flexibility in selecting the training settings, and
- It offered the genetic training option, which includes (i) the ability to train and compare several networks on the same dataset in batch mode and (ii) the genetic algorithm software.

BrainMaker starts at US$195 and prices go as high as US$995 for the system with all available options. It is the world's largest selling neural network development system, with over 25,000 sold. BrainMaker is easy to use, and includes manuals, tutorials, and example files.

Alternative neural network software packages include the aforementioned BackPack from Z Solutions (US$495) and NeuroSolutions 4.0 from NeuroDimension Inc. (US$795 – US$2,495). SAS offers two neural network products, JMP (US$995), which seems to be a rather watered-down neural network product, really just a high-end regression model, and Enterprise Miner, which we saw as a very large, corporate-level application, unsuitable to our project.

Genetic Algorithm Software

As discussed in the body of this paper, we decided to create our own genetic algorithm software.

The only commercial genetic algorithm software we explored was the Genetic Training Option of the BrainMaker neural network software. Our main reason for writing our own genetic algorithm software was to use fitness functions that were not available in BrainMaker. The limitation of our software is that it runs very slowly. We think performance will improve significantly once we convert the software from Microsoft Excel into C++.

Appendix B: Cart Methodology

CART methodology is technically known as binary recursive partitioning:

- *binary* because parent nodes are always split into *two* child nodes
- *recursive* because each child node becomes a *parent* node.

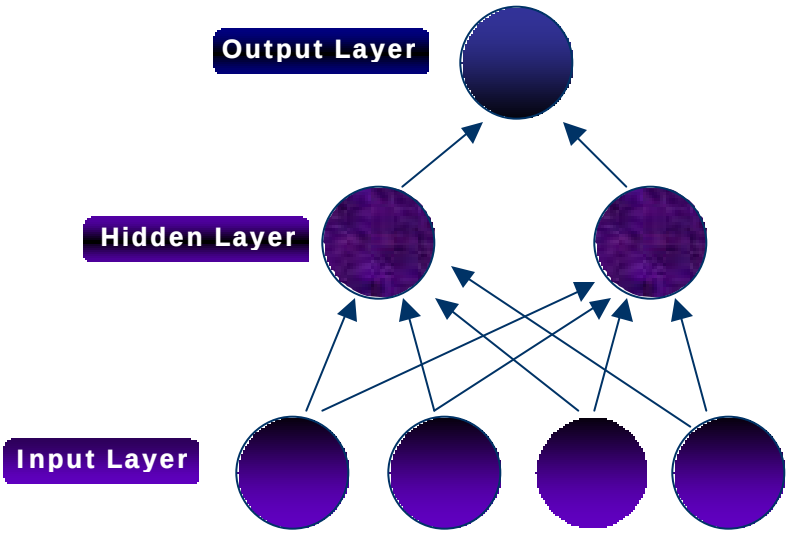
CART analyzes historical data using an exhaustive search methodology, considering every possible way of classifying the data sample into two subsets based on one criterion. All possible classification criteria are compared, with CART selecting the one that results in the optimal split of the data.

This process is repeated for each of the two subsets of data and continues for each subsequent subset until a pre-specified stop rule is reached. Then each terminal node is assigned to a class.

With this exhaustive search method a large number of possible data splits can be searched over, allowing the data modeler to include many questions / variables that may or may not turn out to be informative.[23]

[23] Breiman et al

Appendix C: Pictorial Representation of a Neural Network



Raw data is taken in at the input layer.

From there, it is passed to the hidden layer where the processing occurs. The hidden layer then passes the final result to the output layer.

Example: Neural Network Claims Scoring System

Data about a new claimant – age, gender, and type of injury or illness – would be received at the input layer. It would then be transmitted to the hidden layer, where the information would be processed. A return to work likelihood score would be sent from the hidden layer to the output layer. This score would then be relayed to the claims manager.

Bibliography

Breiman, L, Friedman, J, Olshen, R, Stone, C (1984) Classification and Regression Trees. CRC Press LLC (1998).

Butler, R, Hartwig, R, Gardner, H (1997) HMOs, Moral Hazard and Cost Shifting in Workers' Compensation. *Journal of Health Economics*, Volume 16, No. 2, April 1997, pp 191 – 206.

California Scientific Software (1988) BrainMaker Professional User's Guide and Reference Manual, California Scientific Software (1998).

Canadian Institute of Actuaries, Committee on Expected Experience – Group Life and Health (1998) Canadian Group LTD Termination Experience, 1988 – 1994.

Canadian Institute of Actuaries, Committee on Expected Experience – Group Life and Health (2001) Group Life Waiver Study Based on 1998 – 1994 Canadian Group LTD Termination Experience.

Hassoun, M (1995) Fundamentals of Artificial Neural Networks. The MIT Press.

Lawrence, J (1988) Introduction to Neural Networks: Design, Theory and Applications. California Scientific Software (1994).

Swingler, K (1996) Applying Neural Networks: A Practical Guide. Academic Press Limited.