PREDICTION OF AUDITORS OPINION BASED ON A CLEAN OPINION

Can hidden information be detected through an evolutionary approach?

by

Per-Anton Rønning

Summary in Norwegian

Formålet med denne studien er å undersøke om en kan trene opp prediksjonsregler ved hjelp av en evolusjonær tilnærming, som kan identifisere foretak der revisor vil finne grunn til merknader i kommende år. Basis for prediksjonen er regnskapstall fra årsregnskapet og kodede revisoranmerkninger. Regnskapene er valgt ut slik at de i prediksjonsåret er fri for merknader, mens det i påfølgende er en eller annen revisormerknad (også inkludert "ingen merknad").

Problemstilingen er: Kan vi i en evolusjonær tilnærming avdekke informasjon i et regnskap uten merknad som antyder problemer under utvikling, som revisor vil oppdage og anmerke i det påfølgende år? I så fall vil en ha basis for et "Early warning" system som vil være av betydning når en skal bedømme selskapets drift og utvikling. Dette kan både støtte revisor i hans arbeid, samtidig som de interessenter som skal kredittvurdere selskapet har et verktøy å støtte seg på. Revisoranmerkninger gir uttrykk for revisors oppfatning av selskapets stilling og drift. Dersom disse anmerkningene er entydig negative, vil dette avspeile mulige alvorlige finansielle problemer. Selskaper som er i en slik situasjon kan risikere å gå konkurs. Situasjonen kan friste til økonomisk kriminelle handlinger, eller kan være resultat av slike handlinger. Revisoranmerkningene vil aldri entydig kunne gi informasjon om hvilken situasjon selskapet er i. Men de kan brukes som en proxy (tilnærming) på at noe er galt - eventuelt som en proxy på at selskapet ikke har observerbare problemer.

Basert på 1994 som det valgte prediksjonsår forsøker studien å utvikle regler som kan predikere for 1995 hvilke klasse av revisoranmerkning som vil bli gitt: Enten "intet vesentlig å bemerke", eller merkander som viser at der er forhold som revior ikke er tilfreds med.

En vil gjenom dette se om årsregnskapet inneholder informasjon som kan predikere potensielle problemer, selv om revisor i inneværende år ikke har funnet noe å sette fingeren på. I så fall vil både revisor ,bankenes kredittvurdering samt skattemyndighetene ha et nyttig hjelpemiddel for å kunne velge ut bedrifter som bør settes på en observasjonsliste for tettere oppfølging, før en kritisk situasjon oppstår. Tap kan begrenses ved at en i forkant observerer potensielle problemer, og deretter går til nødvendige omstruktureringer for å løse dem. På denne måten kan dette være et verktøy til å forebygge både konkurs og konkurskriminalitet, eller annen økonomisk kriminalitet for den del.

Datamaterialet strekker seg fra 1988 til 1997, og er levert av Dun & Bradstreet. Det foreligger kodede revisoranmerkninger for perioden 1992-1997.

1. Introduction

The purpose of this study is to investigate the results of an evolutionary approach to predicting auditors opinions in the “result year” based on financial statements in the “year of prediction”. The financial statements in the prediction year are chosen among companies where the auditor found that everything was in order. The financial statements in the result year are organized in two classes: Class 0 contains companies with no serious remarks on the part of the auditor, while class 1 contains companies that have received an explanatory auditors opinion of some degree (see below). The idea behind this study is to examine whether we can predict which class of auditors opinion (0,1) the company will belong to in the result year, based on a prediction year financial statement belonging to class 0. If this is successful, we have shown that there is some hidden information in the financial statements that the genetic programming algoritms can be trained by evolution to make use of. The result is a predicted opinion class in the result year with a degree of accuracy that is reasonably satisfactory. A prediction hitrate of 50% is the poorest result to be expected. There is always a 50-50% chance to be right. If we, based on this model can demonstrate hitrates, say 30 % above this level (64-65%) , we will term this reasonably satisfactory. If this result is obtained, we have tool of analysis which enables us to put class 1 companies on an observation list. The auditor will then make a note of this, and start reexamining certain facts , and bring his results of his examination to the attention of the management. This may be serious financial problems in an early stage of development, it may be (perhaps weak) signs of possible economic crime (systematic drain of cash hidden as ordinary business expenses, or just plain embezzlement), it may be a situation that in the end will cause the company to go bankrupt. At this early stage it will be difficult to determine the real causes behind the predicted results. But placing the company on an observation list will enable the interested parties (owners, board of directors, executive management, auditors, bankers, credit rating companies, tax authorities., suppliers and so forth) to follow up the company more closely.

This study is employing genetic programming as an adaptive learning technology. This means that rules for predicting possible problems in the year to come are evolved gradually based on the ideas of evolution, i.e. natural selection, genetic crossover and mutation. The rules providing the best fit, i.e. producing the greatest number of correct hits will survive in the natural selection process. The mediocre or poor ones will be replaced by a combination of their more successful competitors in a large population of such rules.

Over the past 25 to 30 years much research has been done to develop models in order to predict bankruptcy based on financial data. This effort is surveyed in Jones (1987).

Explicit prediction of bankruptcy is outside the scope of this study (it will be left to another study) , but this topic has been studied at length, and should be surveyed briefly in this report.

Altman (1968) introduced this field of research by employing discriminant analysis, and since then it has been subject to extensive studies by means of a multitude of techniques, such as: multivariate conditional probability models, recursive partitioning models, linear programming, expert systems, neural networks, chaos theory and rough set theory (McKee, 1998). A common approach to develop such models is to review the literature to identify a large set of financial or non financial variables with proven predictive power, and among those, identify a reduced set. One might use either judgement or mathematical analysis to do this, and the resulting set will then be input variables in the bankruptcy prediction model.

A problem is that many models have been built, using different variables and different forms to specify the relationship between the variables. Although some models predict accurately up to 90 % , no generally accepted model for bankruptcy prediction based on underlying financial indicators has emerged so far. This failure seems to incur a significant number of social costs due to bankruptcy, that might otherwise be avoided if a generally applicable theory were available. (McKee, Lensberg, 1998)

In our study, we let the evolutionary process perform a natural selection for us, i.e. from one step to the next in the process of selecting a reduced set of critical variables, the variables that were selected for inclusion in the model less than 4 out of 10 times over 10 independent runs is eliminated, and the remaining ones will be included in then next setup.

2. An Evolutionary Approach

The evolutionary approach in our study is implemented by means of genetic programming, which implies gradually evolving a set of programs which compete for "best fitness". These programs may be closed form mathematical formulas, or they may be algorithms written in some programming language. The basic idea was introduced as genetic algorithms, which is a slightly different approach (Holland, 1975). Genetic algorithms are based on a "bit string" of fixed length, in which the numbers can take on the values 0 or 1, and represents a possible individual that might survive to the next generation. The bit string is evaluated by calculating a value based on the 0's and 1's . In some models the place of each single number in the string represents some phenomenon, which is or is not taken into account according to the value 1 or 0, thus modelling individual behaviour. In other models the bit string is just evaluated and converted to a decimal number which represents some individual behaviour. In order to perform natural selection one must have some fitness measure, by which the individuals are ranked. The better fitness, the greater the probability of survival.

The natural selection mechanism randomly picks out a given number of individuals in groups of 3, ranks them by the fitness criterion, and lets the best fit ones reproduce, i.e. the "loser" in the ranking list of 3 is replaced by some randomly chosen combination of the two "parents".

In genetic algorithms a combination of the two parent's bit strings create a new individual which may or may not do better than the replaced one, and (hopefully) also do better than the parents. This is imitating the crossover event in natural reproduction. It is also common to make a random choice between crossover and mutation. In case of mutation only one "parent" is chosen, and some of its characteristics are changed to create the "offspring". This process is gradually creating a new population, and the speed of this change is subject to parameter settings exogenous to the model. (Koza 1992)

Genetic Programming takes this one step further, inasmuch as the bit string is substituted by a closed form mathematical expression, or a sequence of (machine language) program statements. The genetic program then becomes a collection of primitive functions such as +,-,*,/, Max(), Min() etc., which operate on variables and constants. The variables must be updated continually in the model (from a source that depends on the overall structure of the model - usually some externally generated input data), while the constants do not change. Typical expressions might be

1) $F = V(1)*0.2341 + (V(2)/V(3))*4.5521$
2) $F = V(4)*11.18 - (V(2)-V(3))/0.9863$,

where V(n) indicate variables. At some stage in the algorithm, 1) and 2) might be the parents in a crossover operation, to (randomly) create a new individual

3) $V(1)*0.2341 + (V(2)-V(3))/0.9863$,

which might prove better fit than the replaced individual, or even the parents.
Since these closed form expressions easily may be represented by a sequential computer program (which is executed one statement after another) , one might as well generate programs and perform the same kind of operations on them. That is, the programs are exposed to natural selection by means of the fitness criterion, and the program code is manipulated by means of imitated crossover and mutation operations.

This is in fact what we do in this study, and for this purpose we have chosen to use the program package "Discipulus" from Register Machine Learning Technologies, Inc., which is well suited to study the problem at hand.

3. The Model

The target of prediction is auditors opinion class 0 and 1, based on a coded value of the auditors opinion in the result year. Class 0 represents the set of codes 0-3, and class 1 represents the set of codes 4-9. The prediction is based on a set of financial key indicators in the previous year (the prediction year). The chosen financial indicators will be presented below. The auditor's opinion indicator is a number from 0 to 9 , as listed below:

0 = Clean (unmodified)

1 = Unqualified opinion with explanatory paragraph

2 = Qualified opinion - scope limitation

3 = Both 1 and 2

4 = Adverse opinion

5 = Disclaimer

Explanatory paragraphs - could be symptoms of severe problems

6 = Some shareholder(s) have been granted unsecured loans in violation of applicable law

7 = Payroll tax deductions not put into a separate bank account

8 = Documentation and internal control routines lacking

9 = Equity capital is lost, additional capital influx is necessary to legally continue the business

The data set has been supplied by Dun and Bradstreet, containing year end accounts for nearly 200.000 companies for the period 1988 -1997. Coded auditor's opinions are available for the period 1992-1997.

We have chosen 1995 as the *result year.* We have selected the set of financial indicators from 1994 financial statements of companies with auditors opinion =0, the *prediction year*. Furthermore, we have selected 1994 companies so that half of them have a 0-3 in 1995 (category "healthy in 1995") and half of them have a 4-9 code (category "failed [1)] in 1995").

[1)] By "failed" we do not mean bankrupt or insolvent, we just mean failed to pass the auditors examination receiving a clean opinion

This way of modeling may give answers to the question: Do the financial statements contain information that will enable us to predict next years auditors opinion, though the auditor has found no reason to make any comments this year? To obtain our results we divide this data set in two, the first is the training data set on which the prediction rules are evolved, the second is the validation set, so that the evolved rules can be tested "out of sample": How well do they perform on a data set on which they have not been trained?

The healthy and failed companies are paired according to two criteria, i.e. the "industry code" and within industry code the "company size". Company size is measured by $\log_{10}$ of total assets according to the balance sheet. The point in this is to create a data set where pairs of companies with 4-9 and 0 in 1995 are as similar as possible, both according to industry and size. To conclude, each pair of "healthy" and "failed" companies have been assigned a random number between 0 and 1 by which they are ordered, so that the industry codes and company sizes are as diverse as possible in the two subsets.

The "4-9" situation (class 1) may indicate severe problems in the conduct of business, although many paths may lead to this situation. The market conditions may suddenly take a turn for the worse. Examples might be the quick and possibly lasting drop in prices of crude oil in 1998, or the sudden interest rate hike in Norway during 1998, increasing the financial costs for all leveraged companies. Furthermore it may be obsolete or inadequate business strategy, poor management, major embezzlement (a crime) etc.

Whichever the cause, the "4-9" situation is sufficiently tangible to motivate some preventive action. If we are able to predict this situation before it occurs, we would place the company on an observation list for closer follow up. We might then substantially increase the chance of avoiding a situation which leads to the loss if the company's equity capital. We might also be in a far better position to discover criminal activity. This would be of particular interest to the vendors, the credit institutions and the tax authorities, but also to the owners who risk losing their stock investment. To bring this about, we now put Dicipulus to work on our model.

Dicipulus works with the two data sets presented above, one for training and one for validation. We are starting with the following set of financial indicators (V(.) is the variable number in the model)

V(0) = (Cash Flow)/(Total debt)
V(1) = (Cash balance)/(Short term debt)
V(2) = (Equity capital)/(Total capital)
V(3) = (Profit before taxes + Financial cost)/(Total capital)
V(4) = (Financial cost)/(Gross Revenue)
V(5) = (Short term debt)/(Total debt)
V(6) = Log_{10}(Total capital) - measuring company size
V(7) = Ln(number of years in business)
V(8) = (Drawing rights + Accounts Payable)/Total debt
V(9)=(Losses on Acct. receivable)/(Accounts Receivable)
V(10)=(Assets serving as collateral for loans)/Total Capital

The Discipulus target indicator = 0 if AO-1995 = 0-3,
= 1 if AO-1995 = 4-9

Some comments about the variables:

V(0)

Cash flow is taken as the total "source of capital" computed from the changes in the balance sheet from 1993 to 1994. Depreciation has already been deducted from the assets in the balance sheet, so no accumulated depreciation on the liabilities side of the balance sheet exists. This indicator measures the ratio "liquidity to total debt" , and expresses the company's ability to handle it's leverage.

V(1)

is a liquidity indicator, showing the percentage of cash available to cover short term debt. The smaller this number is, the more vulnerable the company is.

V(2)

is an indicator of how solid the company is. The greater the number the better the bankruptcy resistance if the company should run at a loss for some period of time.

V(3)

measures the return on the total amount of capital employed in the company. It shows the return regardless of financing.

V(4)

measures the ratio of the cost of debt financing compared to profit before taxes. It is a way to compare the "return on debt" to the return on equity capital.

V(5)

measures the ratio short term/long term debt. A hypothesis is, that a company in financial trouble tends to finance its operations with (expensive) short term debt.

V(6)

Some preliminary experiments on American data showed that the Log10(.) rather than the Ln(.) was a good measure of company size in the model

V(7)
is based on the supposition that the younger the company, the greater the risk to run into financial problems

V(8)
shows whether short term debt plays an important role or not in the leveraging of the company.

V(9)
indicates the quality of accounts receivable.

V(10)
indicates the percentage of assets that are reserved as collateral for certain loans

The *target indicator* can, as we see, take on the two values 0 and 1. When predicting there is given a threshold parameter of 0.5, which means that prediction values p in the range of $0 \leq p < 0.5$ are interpreted as 0, and values in the range $0.5 \leq p \leq 1$ are interpreted as 1.

Dicipulus now creates a number of machine language programs based on drawings (10000 different programs in our experiment) form a uniform distribution. That means: It draws random numbers between 0 and 1, each number having the same probability to be drawn. Then it chooses between a set of available program instructions, (+,- , * , /, ...) a set of input variables v(0), v(1), ... a set of constants and a set of working (intermediate) variables f(0), f(1), ... It chooses (at random) which of these elements are to be included and in what sequence. The programs are executed based on the variables that are actually included (there is no guarantee that all variables will be included in one particular program). Dicipulus keeps track of the total number of trials and the total number of correct classifications. A wrong classification causes a “penalty”. Those programs producing the highest number of hits are considered “best”, and these program have the highest probability of being “parents” to the next generation of programs. The offspring programs are created according the genetic programming method as presented above: Natural selection, crossover and mutation.

Dicipulus reports a variety of results. But we wish to focus on the percentage training and validation hitrates. Dicipulus also displays the best program from the training dataset, and which one of these is doing best on the validation data set. These programs are “de-compiled” from machine language to C++, so what you se is C++ code. One should not have too high expectations as to the readability of this program code. It is not a product of a human

programmer, so there are no explanatory comments or any other guidelines to make it easy to understand right away. Instead it is created with only one goal in mind: Efficiency.
The programs may be restricted in length by setting the appropriate parameter. There is no guarantee that a long program with many statements will work better than a shorter one.
So the length of the evolved code in our experiments is not at all a problem. But the programs seem to perform their task in a slightly odd fashion, although one should not be fooled by this. And Discipulus works very fast indeed.

4. Organizing the experiments

Let us recapitulate the dataset selection criteria:
1) First, we select all companies with a "4-9" in 1995, and a "0" in 1994.
We are looking for the "first offenders" in 1995, this year is the one showing the biggest number of "9" companies. We are then looking for a "0-3" match for every one of the "4-9" companies by the following criteria:

2) The match should have "0" in 1994, the same industry code (It turned out that this criteria was always fulfilled, so it was not necessary to select from "the closest possible" industry code) , and

3) it should have the minimum difference in company size compared to the "4-9" company, measured by $\log_{10}$ of total capital.

1)-3) put in another way: For each "failed" company, then within the same industry code, select the "healthy" company most equal in size as the matching "healthy" company.

4) These pairs are then randomly distributed to the training and the validation file, in order to avoid systematic selection as to industry code and company size in the two files.
5) As a technicality all companies with target value 0 are output before the companies with target value 1, for reasons of output readability.
The resulting dataset based on these criterion consist of 532 companies in the training dataset, and the same number in the validation dataset.

The experiments are carried out as follows: The chosen setup of input variables are run 10 times in Dicipulus. Each run will lead to a different result, due to the choice of the random number seed. The random number seed is controlling the random number sequence, and a fixed seed would lead to the same result over and over again, given a fixed number of tournaments, or “generations” in the genetic programming sense. More specifically, different seeds will lead to different choices of variables, constants, program instructions and the order in which they are to be included in the program as it is built. The reason why we perform this variety of runs is to ascertain model robustness. It would not be acceptable if the training and validation hitrates were significantly different from one run to the next. We measure the success by convergence in results over the 10 runs.

Having finished 10 runs for one setup, we examine the frequency of inclusion of the variables in these 10 best training programs. We select the variable with fewest occurrences to be eliminated from the next variable setup. In this way we are approaching the simplest possible model with a good average predicting power. What program is “best” is left to the natural selection mechanism, and we acknowledge the financial indicator variables included in the program as the ones making this particular program the winner. The set of variables included is also a result of the crossover and mutation operations.

The experiment has been carried out 10 separate runs on 4 different variable setups, each new setup containing fever variables than its predecessor.

5. The results

The tables showing a summary of the results are presented in this paragraph. The total set of variables is as follows:

Var.	Definition	Abbreviation
V0	CashFlow/Total Debt	CFlow/Debt
V1	Cash/Short Term Debt	Cash/S.T.D
V2	Equity Capital/Total Capital	ECap/TotCap
V3	(Result before taxes + Financial Costs)/Total Capital	ResTF/TotCap
V4	(Financial costs)/(Gross Revenue)	FinCost/G.Rev

V5	(Short Term Debt)/(Total Debt)	S.T.D/Debt
V6	Log_{10}(Total Capital)	Log_{10}(TotCap)
V7	Ln(Years in Business)	Ln(YiB)
V8	(Drawing rights + Accounts Payable)/(Total Debt)	(DR + AP)/Debt
V9	(Losses on Accounts Receivable)/(Accounts Receivable)	(Loss AR)/AR
V10	(Assets put up as collateral)/(Total Capital)	(Coll.)/(TotCap)

When reporting the results the abbreviations are used as reference to the variables.

5.1 Setup 1

Setup 1 consists of the whole set of variables V0 - V10, as defined in 5.

The result the 10 runs of Setup 1 is:

(A "1" to the right of the variable marks that it is included in the resulting program)

Setup 1		Run#											
Variable def.	Var.#	1	2	3	4	5	6	7	8	9	10	Sum	
CFlow/Debt	V0	1	1	1	1	1	1	1	1	1	1	10	
Cash/S.T.D	V1	0	0	0	0	0	0	0	1	0	0	1	(Excl)
ECap/TotCap	V2	0	1	1	1	0	1	0	1	1	1	7	
ResTF/TotCap	V3	1	0	0	0	1	0	1	0	1	0	4	
FinCost/G.Rev	V4	1	1	1	0	0	1	1	1	1	1	8	
S.T.D/Debt	V5	0	1	0	1	0	1	1	1	0	0	5	
Log10(TotCap)	V6	0	0	0	1	0	0	1	0	1	0	3	(Excl)
Ln(YiB)	V7	0	0	0	0	1	0	0	0	0	0	1	(Excl)
(DR + AP)/Debt	V8	1	1	0	1	1	1	0	0	1	1	7	
(Loss AR)/AR	V9	1	0	1	0	0	1	1	1	1	1	7	
(Coll.)/(TotCap)	V10	1	0	1	1	1	0	0	0	1	1	6	
Training Hitrate		62.4	65.6	64.7	63.7	65.8	64.7	68.2	65.2	64.8	64.5	65.0	
Validation "		58.5	59.0	61.5	57.7	59.0	58.0	56.0	56.0	57.9	60.2	58.4	

All variables included in less than 40% of the cases are chosen for elimination before the next Setup. Variables V1,V6 and V7 are therefore marked for exclusion before setup 2. Based on the training data set we are able to predict correctly in 65% of the cases, which is 30% better than the "neutral" 50%-50% case.

5.2 Setup 2

Setup 2		Run#										
Variable def.	Var.#	1	2	3	4	5	6	7	8	9	10	SUM
Cflow/Debt	V0	1	1	1	1	1	1	1	1	1	1	10
ECap/TotCap	V1	1	0	1	1	1	1	0	0	1	1	7

ResTF/TotCap	V2	1	0	1	0	1	1	0	1	1	1	7
FinCost/G.Rev	V3	0	1	1	0	1	1	1	1	0	1	7
S.T.D/Debt	V4	0	1	0	0	0	1	0	1	0	0	3 (Excl)
(DR + AP)/Debt	V5	1	1	1	1	0	0	1	1	0	1	7
(Loss AR)/AR	V6	1	1	1	1	0	0	1	0	1	0	6
(Coll.)/(TotCap)	V7	1	1	1	1	0	0	1	1	1	1	8
Training Hitrate		65.2	63.5	64.3	64.3	67.5	67.3	64.7	64.8	65.6	63.9	65.1
Validation “		59.6	55.5	61.8	60.2	57.0	57.0	60.9	60.0	63.7	61.3	59.7

Setup 2 shows a slight increase in average training hitrate (0.1 points) and a more substantial increase in average validation hitrate (1.3 points) V4 is excluded ,and the remaining variables are included in setup 3.

5.3 Setup 3

Setup 3		Run#										
Variable def.	Var #	1	2	3	4	5	6	7	8	9	10	SUM
CFlow/Debt	V0	1	1	1	1	1	1	1	1	1	1	10
ECap/TotCap	V1	1	1	1	0	0	0	1	0	0	1	5
ResTF/TotCap	V2	0	1	0	0	0	0	0	1	1	0	3 (Excl)
FinCost/G.Rev	V3	0	1	1	0	1	1	0	1	1	1	7
(DR + AP)/Debt	V4	1	1	0	1	1	1	1	1	1	0	8
(Loss AR)/AR	V5	1	1	1	1	0	0	1	0	1	1	7
(Coll.)/(TotCap)	V6	1	1	0	1	1	1	1	1	0	1	8
Training Hitrate		65.0	65.2	65.8	60.7	66.2	66.0	63.5	64.8	63.5	66.7	64.7
Validation “		61.8	59.8	57.0	56.6	56.6	61.1	61.5	57.1	59.6	63.7	59.5

The average hitrates are falling back slightly, (0.4 - 0.2 points) , but the they largely stay on the same level. Variable V2 is now eliminated, and the next setup is our final setup.

5.4 Setup 4

Setup 4		Run#										
Variable def.	Var.#	1	2	3	4	5	6	7	8	9	10	SUM
ECap/TotCap	V0	1	1	1	1	1	1	1	1	1	1	10
CFlow/Debt	V1	0	1	0	0	1	1	0	1	0	0	4
FinCost/G.Rev	V2	1	1	1	1	1	1	1	1	1	1	10
(DR + AP)/Debt	V3	1	1	1	1	1	1	0	1	1	0	8
(Loss AR)/AR	V4	1	1	1	1	1	0	1	1	1	1	9
(Coll.)/(TotCap)	V5	0	1	1	1	1	0	1	1	1	1	8

Training hitrate	66.0	63.9	65.6	63.9	63.7	63.9	65.0	63.5	65.0	64.5	64.5
Validation hitrate	60.5	61.1	59.2	64.3	58.6	60.0	61.1	61.3	61.5	64.8	61.2

Setup 4 gives us the smallest difference between training and validation hitrates, and no variable is included in less than 40% of the cases. The differences between the two hitrates over the setups are:

Setup 1: 65.0 - 58.4 = 6.6

Setup 2: 65.1 - 59.7 = 5.4

Setup 3: 64.7 - 59.5 = 5.2

Setup 4: 64.5 - 61.2 = 3.3

The elimination of variables seems to have the positive effect that we are evolving rules that perform more and more alike on the training and validation datasets.

Runs 4 and 10 stand out, as the validation hitrates are slightly above the training hitrates. We have evolved two prediction rules which perform just as well on the out of sample validation data as they do on the training data.

5.4.1 A closer study of Run Number 10 and Run Number 4

Run number 10 is attracting attention because of high hitrates both in training and validation. The resulting Genetic Programming algorithm (the training algorithm) has been processed by means of "Matematica", giving the following result:

1) $x = -(V0*(2.082 + V0))/(2*(0.6918 - 0.0268*V0 - 0.0248*V0^2 -1.025*V2)^2) + 2*V2$

2) $y = 0.511 + 0.0234*(x+V4+2*V5)$

We observe some interesting non-linear properties in the x function, where $x = f(v0,v2)$ while $y = g(x,V4,V5)$ is linear in x, V4 and V5.

A negative x will contribute to a value of $y < 0.5$, i.e. a prediction of "no problems ahead", while a positive x will contribute to $y \geq 0$, i.e. a prediction of potential problems.

The graph shows that high equity capital percentage leads to a decreasing x when V2 (Financial Cost/Gross Revenue) is increasing.

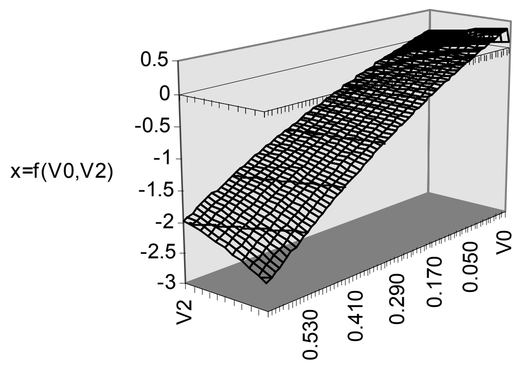
As V0 (the Equity Capital Percentage) is decreasing, we observe that this relationship is reversed from certain points of V0 on: An increasing V2 will now contribute to an increasing x instead. This seems to indicate that financial costs related to gross revenue (incurred as a result of leveraged investments) does not cause potential problems as long as the equity capital percentage is above a certain level. Potential problems will, however, be increasingly present as v2 is increasing while V0 is decreasing.

We have analyzed the x function and present a graph of it in fig. 5.4.1-1

Run number 4 is attracting similar attention as run number 10. The problem is that the evolved "training" algorithm is too complex to be resolved as a closed form expression.

This is not, however, the case with the "validation" algorithm. This algorithm is relatively simple : y=(-V0*0.0468)/((0.0892*V0^2-V5 *V0*0.0936)^2+ 0.3997+V3)+0.5183
Choosing variable V5 (Collateral/TotalCapital) as parameter, and letting V0 and V3 ((DrawingRights+AccountsPayable)/TotalDebt) vary, the graph of y = f(V0,V3,V5) will display the distinct features presented in fig. 5.4.1-2
Increasing V0 creates an increasing "buffer zone" for predicting "no problems" as 3 increases. An increasing V3 will meet the distinct barrier displayed in the graph, in which case y will increase above 0.5, which means predicting a potential problem. The higher V0 the longer V3 can travel along the V3 axis before the barrier is encountered.
These two models (fig. 5.4.1-1,5.4.1-2) seem to display an analogy. In run 10 "financial costs" seems to play the same part as the sum of short term debt items plays in run 4. Decreasing V0 will create a decreasing distance to a barrier where the model switches form predicting "no problems" to predicting "potential problems ahead". Increasing V2 (run 10) or V3 (run 4) is triggering the change in prediction *ceteris paribus*. A closer look at V3 / V4 indicates that they are mirror images of each other. Increasing financial costs V3 is *ceteris paribus* a function of increasing leverage. In this case V3 contains debt items with relatively short time to maturity, and is usually is carrying higher financial costs than long term debt items. Taking this into consideration, the similarity in the two models make sense. The sum of financial costs is relatively sensitive to increase in short term debt items, therefore the GP system has discovered two alternative paths, using two different, but closely interrelated variables to evolve (in principle) similar models of prediction.

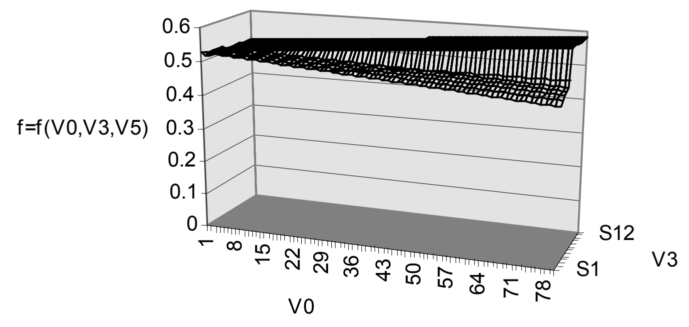
Fig. 5.4.1-1



x= -(V0*(2.082 + V0))/(2*(0.6918 - 0.0268*V0 - 0.0248*V0^2 -1.025*V2)^2) + 2*V2
y = 0.511 + 0.0234*(x+V4+2*V5)
V0=EquityCapital/TotalCapital
V2=FinacialCosts/Revenue
V4=ReceivablesWriteoff/AccountsReceivable

V5=Collateral/TotalCapital

Fig. 5.4.1-2



y=(-V0*0.0468)/((0.0892*V0^2-V5 *V0*0.0936)^2+ 0.3997+V3)+0.5183
V0=EquityCapital/TotalCapital
V3=(DrawingRights+AccountsPayable)/TotalDebt
V5=Collateral/TotalCapital - introduced as parameter

6. A comparison with the Logit statistical model

A logit regression is based on a single equation where the dependent variable, Y_t is a 0-1 dummy variable. It is a maximum likelihood estimation model where the cumulative distribution function is used to define choice probabilities, given by:

$$P_i = 1/(1 + e^{-V})$$

where:

P_i is the estimated probability

$$V = c + \sum_{i=0}^{N-1} (b_i \cdot v_i)$$

c and b_i are the estimated coefficients as described below,

v_i are the observed variable values,

N is the number of observations.

The idea behind this model is to choose, as estimates of c, b_i , the values of c, b_i that maximize the probability of obtaining the sample that is actually observed.

We have run a logit regression in the Shazam program package on the training data set in setups 1-4, and we obtained the following results, based on 532 companies.

6.1 Logit run setup 1

```
VARIABLE     ESTIMATED    STANDARD    T-RATIO
NAME         COEFFICIENT  ERROR
V0           -0.15948     0.07962-2.00290
V1            0.00402     0.00585 0.68604
V2            0.03190     0.08662 0.36829
V3           -0.43330     0.52503-0.82530
V4            0.39029     0.38035 1.02610
V5            0.03461     0.37083 0.09333
V6           -0.29347     0.14998-1.95680
V7            0.12887     0.17864 0.72139
V8            1.23350     0.39540 3.11960
V9            0.33974     0.29690 1.14430
V10           1.10530     0.45731 2.41710
CONSTANT      0.10537     0.70514 0.14943
```

6.1 Logit run setup 1 (continued..)

```
             PREDICTION  SUCCESS TABLE
                   ACTUAL
                   0        1
             0     163      111
PREDICTED    1     103      155

NUMBER OF RIGHT PREDICTIONS        =       318.00
PERCENTAGE OF RIGHT PREDICTIONS    =       0.59774
```

6.2 Logit run setup 2

```
VARIABLE     ESTIMATED    STANDARD    T-RATIO
NAME         COEFF        ERROR
V0           -0.13142     0.07166-1.83400
V1            0.03735     0.08115 0.46024
```

```
V2          -0.63142     0.51852-1.21770
V3           0.35188     0.37865 0.92929
V4          -0.01800     0.36629-0.04914
V5           1.11770     0.38791 2.88130
V6           0.31342     0.29421 1.06530
V7           0.89699     0.43242 2.07440
CONSTANT    -0.33447     0.29498-1.13390
      PREDICTION  SUCCESS TABLE
                  ACTUAL
                  0      1
            0     158    114
PREDICTED   1     108    152

NUMBER OF RIGHT   PREDICTIONS=      310
PERCENTAGE OF RIGHT PREDICTIONS     =      0.58271
```

6.3 Logit run setup 3

```
VARIABLE    ESTIMATED    STANDARD    T-RATIO
NAME        COEFF        ERROR
V0          -0.13197     0.07103-1.85810
V1           0.03764     0.08101 0.46461
V2          -0.63519     0.51279-1.23870
V3           0.35434     0.37712 0.93961
V4           1.10920     0.34759 3.19120
V5           0.31386     0.29402 1.06750
V6           0.90503     0.40023 2.26130
CONSTANT    -0.34634     0.16920-2.04690

      PREDICTION  SUCCESS TABLE
                  ACTUAL
                  0      1
            0     158    114
```

```
PREDICTED    1       108      152

NUMBER OF RIGHT PREDICTIONS           =       310
PERCENTAGE OF RIGHT PREDICTIONS       =       0.58271
```

6.4 Logit run setup 4

```
VARIABLE     ESTIMATED    STANDARD    T-RATIO
NAME         COEFF        ERROR
V0            0.03339     0.07930 0.42105
V1           -0.11654     0.07152-1.62930
V2            0.37849     0.39466 0.95902
V3            1.13930     0.34676 3.28550
V4            0.31952     0.29228 1.09320
V5            0.91707     0.40154 2.28390
CONSTANT     -0.43096     0.15534-2.77440
```

6.4 Logit run setup 4 (continued...)

```
             PREDICTION  SUCCESS TABLE
                         ACTUAL
                         0        1
                     0   161      114
PREDICTED            1   105      152

NUMBER OF RIGHT   PREDICTIONS=     313
PERCENTAGE OF RIGHT PREDICTIONS       =       0.58835
```

6.5 Summary

When comparing the logit hitrates (Percentage of right predictions) and the GP training hitrates we see the following:

Setup	Logit hitrates	GP Training hitrates (Average)
1	60	65.0
2	58	65.1

3	58	64.7
4	59	64.5

GP obtains the highest score in all cases. An additional comment should be made at this point: Preliminary experiments showed an interesting result as to robustness in the two models. Running the analysis on a dataset containing significant out-liers (a small number of extreme values far away from the average value, and far outside one standard deviation) showed that GP discovered these anomalies and adjusted the algorithm accordingly, while the logit model showed notably lower hitrates. When removing the out-liers the logit result improved significantly, while the GP results hardly improved at all. This is a noteworthy property of the GP method: It can be expected to be robust against invasion of out-liers in the data.
GP develops ways around the anomalies and encapsulates them, so to speak.

Then, the next question to examine is: How well will the logit model perform on the *validation* data set, based on the estimated coefficients from the *training* set in setup 4?

That is: We want to analyze

$$P_i = 1/(1 + e^{-V})$$

where

$$V = -0.43096 + 0.03339 \cdot V0 + 0.11654 \cdot V1 + 0.37849 \cdot V2 + 1.1393 \cdot V3 + 0.31952 \cdot V4 + 0.91707 \cdot V5$$

and

V0,V1,--,V5 are values from the *validation* dataset. This was accomplished by means of an MS Excel model, computing a P_i value on each data line (one line per company).

The P_i value is then transformed as follows:

$$P_i' = \begin{cases} 1 & \text{when } P_i >= 0.5 \\ 0 & \text{when } P_i < 0.5 \end{cases}$$

Each P_i' is then compared to the [0,1] target value in the validation file, and whenever P_i' = Target Value a "hit" is counted, otherwise a "no hit" is counted.
The result of this model is a hitrate of 0.5714, a bit less than the result on the training data set.

7. Conclusion

The results of this study show that it is possible to predict with a reasonable degree of correctness that companies that this year have passed the auditors examination with flying colors, the next year will receive au unfavorable opinion, and that this years financial statements contain information that put us in a position to make such predictions. This result is expected to be of interest to the auditor's community, to bankers and vendors following up the credit worthiness of a client , as well as to the tax authorities, and the credit authorities. If we are in a position to predict adverse or explanatory auditor's opinions the following year, we have also put ourselves in a more favorable position when it comes to avoiding financial problems. We have then managed to create a short-list of companies that deserve scrutiny beyond average measures.

References

Holland, J.H. 1975, Adaptation in Natural and Artificial Systems. *University of Michigan Press*

Jones, F.L., 1987. "Current techniques in bankruptcy prediction" *Journal of accounting Literature.*

Koza, J.R., 1992. "Genetic Programming: On the Programming of Computers By Means of Natural Selection." *Massachusetts Institute of Technology.*

McKee, T.E., 1998. A Mathematically Derived Rough Set Model for Bankruptcy Prediction. *Collected Papers of the 7th Annual Research Workshop On Artificial Intelligence and Emerging Technologies In Accounting, Auditing and Tax,* Editor: C.E.Brown. Artificial Intelligence/Emerging Technologies section of the American Accounting Association.

McKee, T.E., Lensberg, T., 1998. Using A Geneitc Algorithm To Obtain A Causally Ordered Model From A Rough Sets Derived Bankruptcy Prediction Model. *A preliminary result reporting and discussion paper, the Norwegian School of Economics and Business Administration.*