Data Mining with Skewed Data

Chapter · February 2010

DOI: 10.5772/9382 · Source: InTech

CITATIONS
3

READS
476

3 authors, including:

Jorn Mehnen
University of Strathclyde
166 PUBLICATIONS **4,084** CITATIONS

SEE PROFILE

12

Data mining with skewed data

Manoel Fernando Alonso Gadi Grupo Santander, Abbey, Santander Analytics
Milton Keynes
United Kingdom

Alair Pereira do Lago
Depart. de Ciencia de Computacao, Inst. de Matematica e Estatistica Universidade de Sao Paulo
Brazil

Jörn Mehnen
Decision Engineering Centre, Cranfield University Cranfield, Bedfordshire MK43 0AL
United Kingdom

In this chapter, we explore difficulties one often encounters when applying machine learning techniques to real-world data, which frequently show skewness properties. A typical example from industry where skewed data is an intrinsic problem is fraud detection in finance data. In the following we provide examples, where appropriate, to facilitate the understanding of data mining of skewed data. The topics explored include but are not limited to: data preparation, data cleansing, missing values, characteristics construction, variable selection, data skewness, objective functions, bottom line expected prediction, limited resource situation, parametric optimisation, model robustness and model stability.

1. Introduction

In many contexts like in a new e-commerce website, fraud experts start investigation procedures only after a user makes a claim. Rather than working reactively, it would be better for the fraud expert to act proactively before a fraud takes place. In this e-commerce example, we are interested in classifying sellers into legal customers or fraudsters. If a seller is involved in a fraudulent transaction, his/her license to sell can be revoked by the e-business. Such a decision requires a degree of certainty, which comes with experience. In general, it is only after a fraud detection expert has dealt with enough complains and enough data that he/she acquired a global understanding of the fraud problem. Quite often, he/she is exposed to a huge number of cases in a short period of time. This is when automatic procedures, commonly computer based, can step in trying to reproduce expert procedures thus giving experts more time to deal with harder cases. Hence, one can learn from fraud experts and build a model for fraud. Such a model requires fraud evidences that are commonly present in fraudulent behavior. One of the difficulties of fraud detection is that fraudsters try to conceal themselves under a "normal" behavior. Moreover, fraudsters rapidly change their modus operandi once it is

discovered. Many fraud evidences are illegal and justify a more drastic measure against the fraudster. However, a single observed indicator is often not strong enough to be considered a proof and needs to be evaluated as one variable among others. All variables taken together can indicate high probability of fraud. Many times, these variables appear in the literature by the name of characteristics or features. The design of these characteristics to be used in a model is called characteristics extraction or feature extraction.

2. Data preparation

2.1 Characteristics extraction

One of the most important tasks on data preparation is the conception of characteristics. Unfortunately, this depends very much on the application (See also the discussions in Section 4 and 5). For fraud modelling for instance, one starts from fraud expert experience, determine significant characteristics as fraud indicators, and evaluates them. In this evaluation, one is interested in measuring how well these characteristics:

- covers (is present in) the true fraud cases;
- and how clearly they discriminate fraud from non-fraud behavior.

In order to cover as many fraud cases as possible, one may verify how many of them are covered by the characteristics set. The discrimination power of any of these characteristics can be evaluated by their odds ratio. If the probability of the event (new characteristics) in each of two compared classes (fraud and non-fraud in our case) are p_f (first class) and p_n (second class), then the odds ratio is:

$$OR = \frac{p_f/(1-p_f)}{p_n/(1-p_n)} = \frac{p_f(1-p_n)}{p_n(1-p_f)}.$$

An odds ratio equals to 1 describes the characteristics as equally probable in both classes (fraud and non-fraud). The more this ratio is greater/less than 1, the more likely this characteristic is in the first/second class than in the other one.

2.2 Data cleansing

In many manuals on best practice in model development, a chapter on data consistency checks, or data cleansing, is present. The main reason for this is to avoid waisting all the effort applied in the model development stage, because of data inconsistency invalidating the dataset in use.

Here we understand *data consistency checks* as being a set of expert rules to check whether a characteristic follows an expected behaviour. These expert rules can be based on expert knowledge or common sense. For example, a common error when filling in the date-of-birth section in a credit card application form is to put the current year instead of the year the person was actually born. In most countries an under sixteen year old can not have a credit card. Therefore, an easy way of checking this inconsistency is to simply calculate the applicant's age and check if it falls within a valid range. With more information available, more complex checks can be applied, such as, e.g. matching name with gender or street name with post code. In some cases the model developer has access to reports stored in Management Information Systems (MIS). If that is the case, it is a highly recommended idea to calculate key population indicators and compare these to portfolio reports. For example, in a credit card industry environment one can check the volume of applications; accept rate, decline rate and take-up rate

and others. It can also be useful to check the univariate distribution of each variable including the percentage of outliers, missing and miscellaneous values.
Having identified the characteristics that contain errors, the next step is to somehow fix the inconsistencies or minimise their impact in the final model. Here we list, in the form of questions, some good practices in *data cleansing* used by the industry that can sometimes improve model performance, increase generalisation power and finally, but no less important, make models less vulnerable to fraud and faults.

1. Is it possible to fix the errors by running some codes on the dataset? Sometimes wrong values have a one-to-one mapping to the correct values. Therefore, the best strategy is to make the change in the development dataset and to carry on with the development. It is important that these errors are fixed for the population the model will be applied to as well. This is because both developing and applying populations must be consistent, otherwise fixing the inconsistency would worsen the model performance rather than improving it;
2. Is a small number of attributes[1] (less than 5%) impacting only few rows (less than 5%)? In this case, one can do a bivariate analysis to determine if it is possible to separate these values into a default (or fault) group. Another option is to drop the rows. However, this tactic might turn out to be risky (see section about missing values);
3. Is the information value of the problematic attribute(s) greater than for the other attributes combined? Consider dropping this characteristic and demand fixing;
4. Is it possible to allow outliers? Simply dropping them might be valid if there are few or there are invalid values. Change their values to the appropriate boundary could also be valid. For example, if an acceptable range for *yearly income* is [1,000;100,000] MU[2] and an applicant has a yearly income of 200,000 MU then it should be changed to 100,000 MU. This approach is often referred to as truncated or censored modelling Schneider (1986).
5. Finally, in an industry environment, when an MIS is available, one can check for the acceptance rate or number of rows to be similar to the reports? It is very common for datasets to be corrupted after transferring them from a Mainframe to Unix or Windows machines.

3. Data skewness

A dataset for modelling is perfectly balanced when the percentage of occurrence of each class is $100/n$, where n is the number of classes. If one or more classes differ significantly from the others, this dataset is called skewed or unbalanced. Dealing with skewed data can be very tricky. In the following sections we explore, based on our experiments and literature reviews, some problems that can appear when dealing with skewed data. Among other things, the following sections will explain the need for stratified sampling, how to handle missing values carefully and how to define an objective function that takes the different costs for each class into account.

[1] possible values of a given characteristic
[2] MU = Monetary Units.

3.1 Missing values

Missing values are of little importance when dealing with balanced data, but can become extremely harmful or beneficial when dealing with skewed data. See how the example below looks harmful at first glance, but indeed exposes a very powerful characteristic.
Table 1 shows an example of a characteristic called Transaction Amount. By looking at the first line of the table one may conclude that the number of missing values is small (1.98%) and decide not to investigate any further. Breaking it down into fraudulent and legitimate transactions, one can see that 269 (32.5%) data items whose values are missings are frauds, which is nearly 9 times bigger than the overall fraud rate in our dataset (1,559/41,707 = 3.74% see Table 2).

Population	# transaction	# missing	% missing
Total	41707	825	1.98%
Fraud	1559	269	17.26%
Legitimate	40148	556	1.39%

Table 1. Fraud/legitimate distribution

Investigating even further, by analysing the fraud rates by ranges as shown in table 2, one can see that the characteristic being analysed really helps to predict fraud; on the top of this, missing values seem to be the most powerful attribute for this characteristic.

Trans. amount	# frauds	# trans.	Fraud rate
Missings	269	825	32.61%
0,100	139	14639	0.95%
101,500	235	10359	2.27%
501,1000	432	8978	4.81%
1001,5000	338	4834	6.99%
5001,+inf	146	2072	7.05%
Total	1559	41707	3.74%

Table 2. Transaction amount bivariate

When developing models with balanced data, in most cases one can argue that it is good practice to avoid giving prediction to missing values (as a separate attribute or dummy), especially, if this attribute ends up with dominating the model. However, when it comes to unbalanced data, especially with fraud data, some specific value may have been intentionally used by the fraudster in order to bypass the system's protection. In this case, one possible explanation could be a system failure, where all international transaction are not being correctly currency converted when passed to the fraud prevention system. This loophole may have been found by some fraudster and exploited. Of course, this error would have passed unnoticed had one not paid attention to any missing or common values in the dataset.

4. Derived characteristics

New or derived characteristics construction is one of, if not the, most important part of modelling. Some important phenomena mapped in nature are easily explained using derived variables. For example, in elementary physics speed is a derived variable of space over time. In data mining, it is common to transform date of birth into age or, e.g., year of study into primary, secondary, degree, master, or doctorate. Myriad ways exist to generate derived characteristics. In the following we give three typical examples:

1. *Transformed characteristics:* transform characteristics to gain either simplicity or generalisation power. For example, date of birth into age, date of starting a relationship with a company into time on books, and years of education into education level;
2. *Time series characteristics:* a new characteristic built based on a group of historical months of a given characteristic. Examples are average balance of a bank account within the last 6 months, minimum balance of a bank account within the last 12 months, and maximum days in arrears[3] over the last 3 months;
3. *Interaction:* variable combining two or more different characteristics (of any type) in order to map interesting phenomena. For example, average credit limit utilization = average utilization / credit limit.

5. Categorisation (grouping)

Categorisation (discretising, binning or grouping) is any process that can be applied to a characteristic in order to turn it into categorical values Witten & Franku (2005). For example, let us suppose that the variable *age* ranges from 0 to 99 and all values within this interval are possible. A valid categorisation in this case could be:

1. category *1*: if age is between 1 and 17;
2. category *2*: if age is between 18 and 30;
3. category *3*: if age is between 31 and 50;
4. category *4*: if age is between 51 and 99.

Among others, there are three main reasons for categorising a characteristic: firstly, to increase generalisation power; secondly, to be able to apply certain types of methods, such as, e.g. a Generalised Linear Model[4] (GLM) Witten & Franku (2005), or a logistic regression using Weight of Evidence[5] (WoE) formulations Agterberg et al. (1993); thirdly, to add stability to the model by getting rid of small variations causing noise. Categorisation methods include:

1. *Equal width*: corresponds to breaking a characteristic into groups of equal width. In the *age* example we easily break age into 5 groups of 20 decimals in each: 0-19, 20-39, 40-59, 60-79, 80-99.
2. *Percentile*: this method corresponds to breaking the characteristic into groups of equal volume, or percentage, of occurrences. Note that in this case groups will have different widths. In some cases breaking a characteristic into many groups may not be possible because occurrences are concentrated. A possible algorithm in pseudo code to create percentile groups is:

```
Nc <- Number of categories to be created
Nr <- Number of rows
Size <- Nr/Nc
Band_start [0] <- Minimum (Value (characteristic[0..Nr]))
//Dataset needs to be sorted by the characteristic to be grouped
For j = 1 .. Nc {
For i = 1 .. Size {
```

[3] Arrears is a legal term for a type of debt which is overdue after missing an expected payment.
[4] One example of this formulation is logistic regression using dummy input variables
[5] This formulation replaces the original characteristic grouped attribute for its weight of evidence

```
Value_end <- Value(characteristic[i+Nc*Size])
}
Band_end[j] <- Value_end
Band_start[j+1] <- Value_end
}
```

3. *Bivariate grouping*: this method corresponds to using the target variable to find good breaking points for the ranges of each group. It is expected that, in doing so, groups created using a bivariate process have a lower drop in information value, whilst it can improve the generalisation by reducing the number of attributes. One can do this in a spreadsheet by recalculating the odds and information value every time one collapses neighbouring groups with either similar odds, non-monotonic odds or a too small population percentage.

Next, we present one possible process of grouping the characteristic *age* using a bivariate grouping analysis. For visual simplicity the process starts with groups of equal width, each containing 10 units (see Table 3). The process consists of eliminating intervals without monotonic odds, grouping similar odds and guaranteeing a minimal percentage of individuals in each group.

Age	goods	bads	Odds	%tot
0-9	100	4	25.00	1.83%
10-19	200	10	20.00	3.70%
20-29	600	50	12.00	11.46%
30-39	742	140	5.30	15.55%
40-49	828	160	5.18	17.42%
50-59	1000	333	3.00	23.50%
60-69	500	125	4.00	11.02%
70-79	300	80	3.75	6.70%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 3. Age bivariate step 1/4

Age	goods	bads	Odds	%tot
0-9	100	4	25.00	1.83%
10-19	200	10	20.00	3.70%
20-29	600	50	12.00	11.46%
30-39	742	140	5.30	15.55%
40-49	828	160	5.18	17.42%
50-79	**1800**	**538**	**3.35**	**41.22%**
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 4. Age bivariate step 2/4

The result of the first step, eliminating intervals without monotonic odds can be seen in Table 4. Here bands 50-59 (odds of 3.00), 60-69 (odds of 4.00) and 70-79 (odds of 3.75) have been merged, as shown in boldface. One may notice that merging bands 50-59 and 60-69 would

Age	goods	bads	Odds	%tot
0-9	100	4	25.00	1.83%
10-19	200	10	20.00	3.70%
20-29	600	50	12.00	11.46%
30-49	**1575**	**300**	**5.23**	**32.97**%
50-79	1800	538	3.35	41.22%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 5. Age bivariate step 3/4

result in a group with odds of 3.28; hence resulting in the need to merge with band 70-79 to yield monotonic odds.
By using, for example, 0.20 as the minimum allowed odds difference, Table 5 presents the result of step two where bands 30-39 (odds of 5.30) and 40-49 (odds of 5.18) have been merged. This is done to increase model stability. One may notice that odds retrieved from the development become expected odds in a future application of the model. Therefore, these values will vary around the expectation. By grouping these two close odds, one tries to avoid that a reversal in odds may happen by pure random variation.

Age	goods	bads	Odds	%tot
0-19	**300**	**14**	**21.43**	**5.54**%
20-29	600	50	12.00	11.46%
30-49	1575	300	5.23	32.97%
50-79	1800	538	3.35	41.22%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 6. Age bivariate step 4/4

For the final step, if we assume 2% to be be the minimum allowed percentage of the population in each group. This forces band 0-9 (1.83% of total) to be merged with one of its neighbours; in this particular case, there is only the option to merge with band 10-19. Table 6 shows the final result of the bivariate grouping process after all steps are finished.

6. Sampling

As computers become more and more powerful, sampling, to reduce the sample size for model development, seems to be losing attention and importance. However, when dealing with skewed data, sampling methods remain extremely important Chawla et al. (2004); Elkan (2001). Here we present two reasons to support this argument.
First, to help to ensure that no over-fitting happens in the development data, a sampling method can be used to break the original dataset into training and holdout samples. Furthermore, a stratified sampling can help guarantying that a desirable factor has similar percentage in both training and holdout samples. In our work Gadi et al. (2008b), for example, we executed a random sampling process to select multiple splits of 70% and 30%, as training and holdout samples. However, after evaluating the output datasets we decided to redo the sampling process using stratified sampling by fraud/legitimate flag.

Second, to improve the model prediction, one may apply an over- or under- sampling process to take the different cost between classes into account. Cost-sensitive procedure Elkan (2001) replicates (oversampling) the minority (fraud) class according to its cost in order to balance different costs for false positives and false negatives. In Gadi et al. (2008a) we achieved interesting results by applying a cost-sensitive procedure.
Two advantages of a good implementation of a cost-sensitive procedure are: first, it can enable changes in cut-off to the optimal cut-off, For example, in fraud detection, if the cost tells one, a cost-sensitive procedure will consider a transaction with as little as 8% of probability of fraud as a potential fraud to be investigated; second, if the cost-sensitive procedure considers cost per transaction, such an algorithm may be able to optimise decisions by considering the product [probability of event] x [value at risk], and decide on investigating those transactions in which this product is bigger.

7. Characteristics selection

Characteristics selection, also known as feature selection, variable selection, feature reduction, attribute selection or variable subset selection, is commonly used in machine learning and statistical techniques to select a subset of relevant characteristics for the building of more robust models Witten & Franku (2005).
Decision trees do characteristics selection as part of their training process when selecting only the most powerful characteristics in each subpopulation, leaving out all weak or highly correlated characteristics. Bayesian nets link different characteristics by cause and effect rules, leaving out non-correlated characteristics Charniak (1991). Logistic Regression does not use any intrinsic strategy for removing weak characteristics; however, in most implementations methods such as forward, backward and stepwise are always available. In our tests, we have applied a common approach in the bank industry that is to consider only those characteristics with information value greater than a given percentage threshold.

8. Objective functions

When defining an objective function, in order to compare different models, we found in our experiments that two facts are especially important:

1. We have noticed that academia and industry speak in different languages. In the academic world, measures such as Kolmogorov Smirnov (KS) Chakravarti et al. (1967) or Receiver Operating Characteristic (ROC curve) Green & Swets (1966) are the most common; in industry, on the other hand, rates are more commonly used. In the fraud detection area for example it is common to find measures such as hit rate (confidence) and detection rate (cover). Hit rate and detection rate are two different dimensions and they are not canonical. To optimise a problem with an objective having two outcomes is not a simple task Trautmann & Mehnen (2009). In our work in fraud detection we avoided this two-objective function by calculating one single outcome value: the total cost of fraud;
2. In an unbalanced environment it is common to find that not only the proportion between classes differs, but also the cost between classes. For example, in the fraud detection environment, the loss by fraud when a transaction is fraudulent is much bigger than the cost to call a customer to confirm whether he/she did or did not do the transaction.

9. Bottom line expected prediction

The problem of finding the best model can be computationally expensive, as there are many parameters involved in such a search. For this reason, it is very common for model developers to get satisfied with suboptimal models. A question equally difficult to answer, in general, is how far we are from an optimum. We do not intend to respond to this question here; what we want to address is a list of ways to help the model developer to estimate a minimum acceptable performance before getting close to the end of the model development. In our fraud analysis we found two good options for estimating a bottom line for expected suboptimal cost: a first option could be the cost resulting from a Naïve Bayes model. It is important to notice that Naïve Bayes does not need any grouping, characteristics selection or parameter tuning; a second option could be to consider the cost from a first "quick and dirty" model developed using the method chosen by the model developer.

10. Limited resource situation

Many real-world application present limited resource problems. This can make the decision of what is the best model different compared to a model without restrictions. In a hospital, for example, there may be a limited number of beds for patients; in a telephone costumer service facility, there may be a limited number of attendants; in the fraud detection world the number of people available to handle manual transactions is in general fixed; and the number of transactions each member of fraud detection can handle per day is also fixed due to practical reasons. In such applications, being aware of the capacity rate becomes very important. It is also extremely important for the model outcome to indicate the probability[6] of the event rather than providing a simple yes/no response. By having the outcome as a probability, models can be compared using for example, cutoffs that keep the selecting rate equal to the capacity rate. In fraud detection, comparing models detection rate and hit rate fixing for example 1000 transaction to be investigated.

11. Parametric optimisation

Once we have the data and the optimisation criteria, the following questions have to be answered:

> Which classification method is recommended for producing the best model for any given application?
>
> Which parameter set should be used?

For instance, we can apply classification methods such as: Neural Networks (NN), Bayesian Networks (BN), Naïve Bayes (NB), Artificial Immune Systems (AIS) and Decision Trees (DT), Support Vector Machines (SVM), Logistic Regression and others. In fact, there is not a final and unique answer to this first question. Support Vector Machines, for instance, is known to be very effective for data with a very large number of characteristics and is reported to perform well in categorisation problems in Information Retrieval. However, our experience with SVM on fraud data did not meet our expectations. For many parameter sets, the method did not even converge to a final model and this behaviour for unbalanced data is reported to not be uncommon.

[6] Or be possible to transform it into a probability.

In order to assess methods many factors can be used including the chosen optimisation criteria, scalability, time for classification and time spent in training, and sometimes more abstract criteria as time to understand how the method works. Most of the time, when a method is published, or when an implementation is done, the method depends on parameter choices that may influence the final results significantly. Default parameters, in general, are a good start. However, most of the time, they are far from producing the best model. This comprises with our experience with many methods in many different areas of Computer Science. This is particular true for classification problems with skewed data.
Quite often we see comparisons against known methods where the comparison is done by applying a special parameter variation strategy (sometimes a parameter optimisation) for the chosen method while not fairly conducing the same procedure for the other methods. In general, for the other methods, default parameters, or a parameter set published in some previous work is used. Therefore, it is not a surprise that the new proposed method wins. At a first glance, the usage of the default parameter set may seem to be fair and this bias is often reproduced in publications. However, using default sets can be biased by the original training set and, thus, not be fair.
Parameter optmisation takes time and is rarely conduced. For a fair comparison, we argue that one has to fine tune the parameters for all compared method. This can be done, for instance, via an exhaustive search of the parameter space if this search is affordable, or some kind of sampling like in Genetic Algorithm (GA)[7] (see Figure 1). Notice, that the final parameter set cannot be claimed to be optimal in this case.
Unfortunately, this sampling procedure is not as easy as one may suppose. There is not a single best universal optimisation algorithm for all problems (No Free Lunch theorem - Wolpert and Macready 1997 Wolpert & Macready (1997)). Even the genetic algorithm scheme as shown in Figure 1 might require parameter adjustment. According to our experience, we verified that a simple mistake in the probability distribution computation may drive the results to completely different and/or misleading results. A good genetic algorithm requires expertise, knowledge about the problem that should be optimised by the GA, an intelligent design, and resources. The more, the better. These considerations also imply that comparisons involving methods with suboptimal parameter sets depend very much on how well each parameter space sampling was conduced.

12. Robustness of parameters

After the parameter optimisation has been conducted, it can be advantageous or desirable to have the optimised parameters independent from the training set, i.e. they can be applied to different datasets of the same problem. In this case we can call this parameter set *robust*.
When the parameter are not robust, the optimisation process is not as strong as expected since the obtained optimised parameter set has no or little generalisation power. In this case, in our experiments, we found that it is a good approach to sacrifice some prediction power in order to gain robustness in the parameter set. Note that a procedure using n-fold cross validation could lead to a parameter set that is more independent from a dataset. However, we choose to present a different approach which also generates robust parameter sets with more control of what is happening during the process. This procedure is based on repeated sampling from the development dataset into training and holdout samples. Then, we applied parameter

[7] One could also use some kind of Monte Carlo, Grid sampling or Multiresolution alternatives.

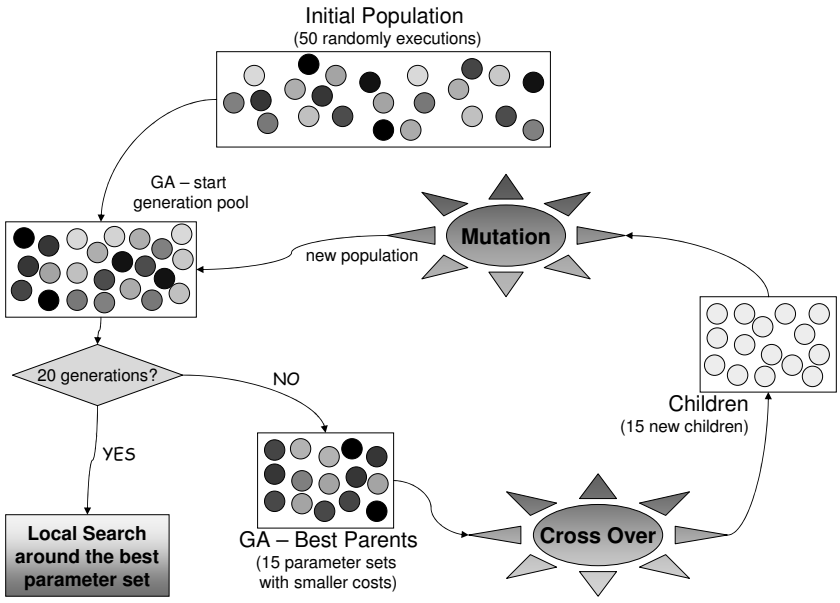

Fig. 1. Genetic Algorithm for parameters optimisation. We start with an initial pool of e.g. 50 random individuals having a certain fitness, followed by e.g. 20 Genetic Algorithm (GA) generations. Each GA generation combines two randomly selected candidates among the best e.g. 15 from previous generation. This combination performs: crossover, mutation, random change or no action for each parameter independently. As the generation goes by, the chance of no action increases. In the end, one may perform a local search around the optimised founded by GA optimisation. Retrieved from Gadi et al. Gadi et al. (2008b).

optimisation and choose the set of parameters which is the best in average over all splits at the same time.

In our work, in order to rewrite the optimisation function that should be used in a GA algorithm, we have used a visualization procedure with computed costs for many equally spaced parameter sets in the parameter space. After having defined a good optimisation function, due to time constraints, we did not proceed with another GA optimisation, but we reused our initial runs used in the visualization, with the following kind of *multiresolution optimisation* Kim & Zeigler (1996) (see Figure 2):

- we identified those parameters that have not changed, and we frozen these values for these respective parameters;
- with any other parameter, we screened the 20 best parameter sets for every split and identified a reasonable range;
- for all non-robust parameters, we chose an integer step s so the search space did not explode;

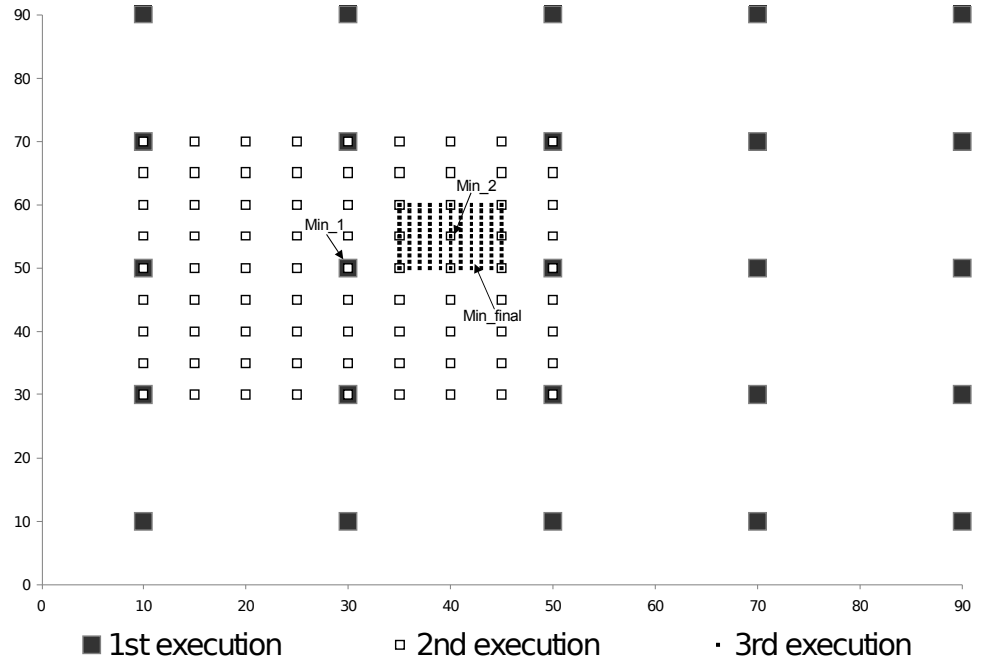

Fig. 2. An example of the multiresolution optimisation that was applied in order to find robust parameters. In this example one can see two parameters in the search space and three steps of this multiresolution optimisation. For the parameter represented in horizontal line, the search space in first step ranges from 10 to 90 with step size 20 and the minimum was found for 30. In the second step, the scale ranges from 10 to 50 with step size 5 and the minimum was found for 40. In third step, it ranges from 35 to 45, with step size 1, which is equivalent to a local exhaustive search in this neighborhood. Retrieved from Gadi et al. Gadi et al. (2008a).

- we evaluated the costs for all possible combinations according to the search space defined above and found the parameter set P that brings the minimum average cost among all the different used splits;
- if the parameter set P was at the border of the search space, we shifted this search space by one step in the direction of this border and repeated last step until we found this minimum P in the inner area of the search space;
- we zoomed the screening in on the neighborhood of P, refined steps s, and repeated the process from then on until no refinement was possible.

13. Model stability

In industry, generally the aim of modelling is to apply a model to a real situation and to generate profit, either by automating decision making where a model was not previously available or replacing old models by a new and improved one. For doing so, most model development processes rely on past information for their training. Therefore, it is very important to be able to assess whether or not a model is still fit for propose when it is in use, and to have a set of

actions to expand the model's life span. In this section we explore advantages of using out-of-time samples, monitoring reports, stability by vintage, vintage selection and how to deal with different scales over time.

13.1 Out-of-time:
an Out-Of-Time sample (OOT) is any sample of the same phenomena used in the model development that is not in the development window[8], historic vintages or observation point selected for development. In most cases in reality a simple split of the development sample into training and testing data cannot identify a real over-fitting of the model Sobehart et al. (2000). Therefore, the most appropriated approach to identify this change is either to select a vintage or observation point posterior to the development window or select this previously to the development window. The second approach gives the extra advantage of using the most up-to-date information for the development.

13.2 Monitoring reports:
the previous action, OOT, should be best done before the actual model implementation; after that, it becomes important to evaluate whether the implemented model still delivers a good prediction. For this purpose, it is crucial to create a set of period based monitoring reports to track the model's performance and stability over time.

13.3 Stability by vintage:
stability by vintage corresponds to breaking the development sample down by time within the development window and evaluate the model's performance in all of the different periods within the data. For example, if one has information collected from January 08 to December 08, a good stability by vintage analysis would be to evaluate the model's performance over each month of 2008. This tends to increase the chance of a model to be stable after its implementation.

13.4 Vintage selection:
many phenomena found in nature, and even in human behaviour, repeat themselves year after year in a recurrent manner; this is known as *seasonality*. Choosing a development window from a very atypical month of the year can be very misleading; in credit cards, for example, choosing only December as the development time window can lead to overestimation of expected losses since this is the busiest time of the year. Two approaches intend to mitigate this problem. Both approaches are based on selecting the broadest development window possible. One common window size is 12 months, allowing the window to cover the whole year. Please notice, there is no need to fix the start of the window to any particular month. The first approach corresponds to simply develop the model with this pool of observation points; it is expected for the model to be an average model that will work throughout the year. A second approach is to introduce a characteristic indicating the "month of the year" the information was collected from, or any good combination of it, and then to develop the model. As a result, one would expect a model that adjusts better to each observation point in the development window.

[8] Here we understand development window as being the period from where the training samples were extracted. This can be hours, days, months, years, etc.

13.5 Different scale over time:

Another common problem applies to the situation where characteristic values fall outside the training sample boundaries or some unknown attributes occur. To reduce the impact of this problem, one can always leave the groups with the smallest and biggest boundaries as negative infinite and positive infinite, respectively, for example, changing [0,10];[11,20];[21,30] to $]-\infty$,10];[11,20];[21,$+\infty$[. Furthermore, undefined values could always be assigned to a default group. For example, if for a numeric characteristic a non-numeric value ocurrs it could be assigned to a default group.

14. Final Remarks

This work provided a brief introduction to pratical problem solving for machine learning with skewed data sets. Classification methods are generally not designed to cope with skewed data, thus, various action have to be taken when dealing with imbalanced data sets. For a reader looking for more information about the field we can recommend a nice editorial by Chawla et al. **?** and three conference proceedings Chawla et al. (2003); Dietterich et al. (2000); Japkowicz (2000). In addition, good algorithm examples can be found in Weka Witten & Franku (2005) and SAS Delwiche & Slaughter (2008).

Perhaps, most solutions that deal with skewed data do some sort of sampling (e.g: with undersampling, oversampling, cost sensitive training Elkan (2001), etc.). These contributions are effective Gadi et al. (2008a) and quite well known nowadays.

This text provides recommendations for practitioners who are facing data mining problems due to skewed data.

Details on the experiments can be found at Gadi et al. (2008b) and Gadi et al. (2008a), which presents an application of Artificial Immune Systems on credit card fraud detection.

Finally, another subject explored in this work was the importance of parametric optimization for chosing a good classification method for skewed data. We also suggested a proceedure for parametric optimization.

15. References

Agterberg, F. P., Bonham-Carter, G. F., Cheng, Q. & Wright, D. F. (1993). Weights of evidence modeling and weighted logistic regression for mineral potential mapping, pp. 13–32.

Chakravarti, I. M., Laha, R. G. & Roy, J. (1967). *Handbook of Methods of Applied Statistics*, Vol. I, John Wiley and Sons, USE.

Charniak, E. (1991). Bayesians networks without tears, *AI Magazine* pp. 50 – 63.

Chawla, N. V., Japkowicz, N. & Kotcz, A. (2004). Special issue on learning from imbalanced data sets, *SIGKDD Explorations* **6**(1): 1–6.

Chawla, N. V., Japkowicz, N. & Kotcz, A. (eds) (2003). *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets*.

Delwiche, L. & Slaughter, S. (2008). *The Little SAS Book: A Primer*, SAS Publishing.

Dietterich, T., Margineantu, D., Provost, F. & Turney, P. (eds) (2000). *Proceedings of the ICML'2000 Workshop on Cost-Sensitive Learning*.

Elkan, C. (2001). The foundations of cost-sensitive learning, *IJCAI*, pp. 973–978.
URL: *citeseer.ist.psu.edu/elkan01foundations.html*

Gadi, M. F. A., Wang, X. & Lago, A. P. d. (2008a). Comparison with parametric optimization in credit card fraud detection, *ICMLA '08: Proceedings of the 2008 Seventh International*

Conference on Machine Learning and Applications, IEEE Computer Society, Washington, DC, USA, pp. 279–285.

Gadi, M. F., Wang, X. & Lago, A. P. (2008b). Credit card fraud detection with artificial immune system, *ICARIS '08: Proceedings of the 7th international conference on Artificial Immune Systems*, Springer-Verlag, Berlin, Heidelberg, pp. 119–131.

Green, D. M. & Swets, J. A. (1966). *Signal Detection Theory and Psychophysics*, John Wiley and Sons.
URL: *http://www.amazon.co.uk/exec/obidos/ASIN/B000WSLQ76/citeulike00-21*

Japkowicz, N. (ed.) (2000). *Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets*. AAAI Tech Report WS-00-05.

Kim, J. & Zeigler, B. P. (1996). A framework for multiresolution optimization in a parallel/distributed environment: simulation of hierarchical gas, *J. Parallel Distrib. Comput.* **32**(1): 90–102.

Schneider, H. (1986). *Truncated and censored samples from normal populations*, Marcel Dekker, Inc., New York, NY, USA.

Sobehart, J., Keenan, S. & Stein, R. (2000). Validation methodologies for default risk models, pp. 51–56.
URL: *http://www.moodyskmv.com/research/files/wp/p51p56.pdf*

Trautmann, H. & Mehnen, J. (2009). Preference-based pareto optimization in certain and noisy environments, *Engineering Optimization* **41**: 23–38.

Witten, I. H. & Franku, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*, Elsevier.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization, *Evolutionary Computation, IEEE Transactions on* **1**(1): 67–82.
URL: *http://dx.doi.org/10.1109/4235.585893*