

Chapter 8

Telia: Detection of frauds in a Media-on-Demand system in an IP network

8.1 Problem description

Henrik Jacobsson

Telia has performed field pilot tests on a Media-on-Demand (MoD) system for downloading and paying for movies and television over an ADSL connection. The problem is that the service may invite morally non-scrupulous people to take advantage of the system for their own fraudulent purposes. Telia is interested in developing an automated system for detecting these frauds. There is a large number of possible frauds that can be conducted. However, for practical purposes, Telia has selected a small number of known frauds, feasible in terms of detectability, to be studied in this project. These frauds are:

Illegal redistribution. This is when a user downloads movies or other broadcasted material (in a legal or illegal way) and later redistributes them to other users.

Excess download. This is when the user has tampered with the billing system, making it possible to download movies or TV without paying for it.

Subscription fraud. A user registers himself under false identity and manages to use the services of the media-on-demand system without paying for it.

In addition to detecting these frauds, Telia wants to detect new fraud types. This is hypothetically possible by detecting behaviours deviating from normal behaviour and then further analyze these behaviours to establish whether they are fraudulent or not.

8.1.1 Fraud indicators

Telia lists a number of possible indicators for the three kinds of frauds being studied.

Illegal redistribution. If the ratio between downloaded and uploaded data volumes is suspiciously high in general, this may be an indicator that the user redistributes movies. If the user is uploading a lot of data at the same time as he/she is downloading a live event, this is an even stronger indicator. If the user is spending a lot of money on an unreasonable number of movies, it is reasonable to

believe that the user is (financing this by) selling them. Since the redistribution is more likely to be focused on a specific service, an abnormal increase in usage of a service together with an increased traffic is considered as an indicator. If the received and transmitted data is of the same kind (*e. g.* by showing similar statistics or burst patterns), this may indicate redistribution fraud.

Excess download. If more data is transmitted than the user is billed for, excess download is probably in progress; alternatively, a billing system error has appeared.

Subscription fraud. If a new user deviates substantially from the average new user and uses services for an excessive amount it may be an indicator that the user ID is used by someone who will not pay for the services.

However, due to lack of time, not all of the indicators above are supported by the data. The indicators used must all be derivable from the data described in the following text. The extraction and interpretation of such indicators belongs to the task that should be solved by the learning systems.

8.2 Data description

Henrik Jacobsson

8.2.1 The data collection

The original idea was to use real data, *i. e.* data from the pilot tests Telia is conducting. The collection of real data was problematic since there were unfortunately very few users (only 12 users, many of which did never order any movies) and it was difficult to explicitly define *when* exactly a fraud is being committed. This data was used only at SICS (see section 8.3.2). To provide more data, Telia has created a simulator in which thousands of users can be simulated in a controlled way.

We choose to use 518 simulated users for training of which 18 are fraudulent (6 of each fraud type). The same number of users is used for the test set. The simulator was run for 6 months of simulated time. The fraudulent users were programmed to behave as normal users until they start their fraudulent behaviour some time during the 3 first months. Unfortunately, the simulator did not log *when* a user turns fraudulent. Therefore we filtered out the first three months, where there is an uncertainty if the user is fraudulent or not. This means that, in practice, the problem was reduced to a *classification* of users.

8.2.2 The raw data

The original simulated data distributed by Telia will only be described briefly since no classification will be based on this data directly. In summary, the data can be said to be “difficult” in many ways. It is sampled at different parts of the system and at different rates. Some of the data is sampled regularly (*e. g.* router statistics) and some is generated at the triggering of some events (*e. g.* ordering of movies). This type of irregular representation of time is not suitable for the majority of methods used in this project; yet, this situation reflects the actual situation of an operating telecom system. Moreover, it is difficult to identify users since the identity of a user is not represented in the same way in all files. To separate users from each other is very important since we want to separate fraudulent from normal users.

8.2.3 Refined data

The difficulties of identification and resampling of data with a synchronous time representation were overcome and it resulted in a refined dataset. This dataset was divided into one sequence for each separate user. Each row in the sequence represents one half hour of simulated time where everything that was *started* within that interval is included.

Column #	Attribute	Description
1	Line number	—
2	Time	Time after 2001-05-05 00:00:00 in seconds. Denotes the starting point of the interval.
3	User ID	A unique user ID.
4	Order requests	Ordered movies during that interval.
5	Delivery Notification	Delivered movies during that interval.
6	Billing Notification	The number of movies the user has been billed for during the interval.
7	Downloaded (DL)	Downloaded bytes during the interval.
8	Uploaded (UL)	Uploaded bytes during the interval.
9	UD-ratio (UD)	$UD = \begin{cases} UL/DL & \text{if } DL \neq 0 \\ UL/\overline{DL} & \text{otherwise} \end{cases}$ where \overline{DL} is the average downloaded bytes for all intervals.
10	Downloaded unicast	Downloaded Unicast-packets.
11	Uploaded unicast	Downloaded Unicast-packets.
12	UD-Ucast-ratio	The same procedure as for attribute 9.
13	Hour	Hour of the day the interval started, <i>i. e.</i> an integer in between 0 and 23
14	Active	1 if user is active in the interval, 0 otherwise.
15	No fraud	1 if the user never commits any fraud, 0 otherwise.
16	Illegal redistribution	1 if the user commits illegal redistribution, 0 otherwise
17	Billing fraud	1 if the user commits billing fraud, 0 otherwise (same as Excess download described above).
18	Break in fraud	1 if the user commits break in fraud, 0 otherwise (same as Subscription fraud described above).

Table 8.1: The attributes used extracted from Telia’s data. The last four attributes represent the target of the domain and the first three are not used for the classification.

A description of the attributes is found in table 8.1. Since some information may occur in the original data that are not transferred into the suggested representation there is a risk that some helpful information may have been deleted. This is especially critical for detecting novel fraud types. There are, for example, assumptions regarding the identification of users that work fine for the normal case but may not be applicable if someone starts hacking other user accounts or if the IP-addresses are changed, which may happen since dynamic IP-addresses are used.

8.3 The model used at SICS

Daniel Gillblad

8.3.1 Supervised and unsupervised fraud detection

Fraud detection can be performed in two fundamentally different ways. Either the fraud types are known, and data classified into fraudulent, perhaps with a fraud type tag, or non fraudulent is available. It is then at least hypothetically possible to construct a classifier that will classify new data patterns into fraudulent or non fraudulent, perhaps even with a very simple set of rules specified by someone who knows how these frauds are usually committed. The other possibility is that there is no knowledge about what fraudulent behaviour might look like, or perhaps that we want to be able to detect new types of frauds. This makes the task much harder, since we cannot extract any information on what the frauds might look like from data. Instead, we have to try to build a model of normal, non fraudulent behaviour. When new data patterns deviate significantly from this model, we can classify it as a possible fraud.

In the first scenario, where we have labelled data available for training of a learning system, we can use a supervised learning algorithm to construct our classifier. Supervised learning is only possible when labelled data is available. When this is not the case, we have to rely on unsupervised learning, *i. e.* there are no labelled examples to be learned by the system. This is the situation in the second scenario, where we just want to build a general model of the data that represents normal behaviour. The second scenario can be regarded as generally more difficult and sensitive to the choice of model. On the other hand it is, if it is performing with sufficient precision, perhaps even more useful to a company than a normal classifier trained on labelled data.

The supervised training method uses a mixture of Gaussians, with one Gaussian for each fraud type. The model is trained by estimating each Gaussians parameters on the corresponding data, *i. e.* the Gaussian that represents fraud type one is estimated from all data categorized as fraud type one, the Gaussian that represents fraud type two is estimated from all data belonging to fraud two and so on. When a classification is made, the input pattern is presented to each Gaussian. The corresponding fraud type of the Gaussian with the highest likelihood for the pattern is the classification result.

In the unsupervised case, a mixture of Gaussians is trained on non fraudulent data by the Expectation Maximization algorithm (see section 2.6.7). The resulting distribution is an estimate of the real distribution of non fraudulent data. If a new pattern has a very low likelihood of being drawn from this distribution, it can be considered to be a possible fraud or at least deviant behaviour.

8.3.2 The data used to train the models

The pilot test data set distributed by Telia contained fraud, but without any good indication of when and by whom the frauds were committed. To be able to use the dataset for testing fraud detection, the data was labelled for all users and time steps using a simple scheme based on the hints given about when the different frauds took place. The data was labelled as

1. Inactive
2. Active
3. Redistribution
4. Excess download
5. Subscription fraud

Patterns were only labelled as frauds at the time they could be considered to be committed, resulting in a rather low number of fraudulent patterns. This and the fact that each type of fraud is committed only by one user makes it impossible to divide the dataset into one training set and one testing set for

	Inactive	Active	Redistribution	Excess download	Subscription fraud
Inactive	23292	1995	0	0	0
Active	81	28730	5	0	0
Redistribution	0	0	153	0	0
Excess download	0	0	0	4	0
Subscription fraud	0	0	0	0	2

Table 8.2: Supervised prediction results on pilot test data

cross validation. One separate dataset was constructed though for training of the unsupervised model. This dataset contains every second day from all users, with all fraudulent behaviour filtered out.

Not all available data was used for training the models. Only order requests, delivery notification, billing notification, downloaded bytes and uploaded bytes were used, both on pilot test data and on simulated data. In both cases, the uploaded and downloaded bytes were also transformed by the logarithm to the power of three to compensate for dynamics in the data.

In the simulated data case, the amount of data was simply too large to train the models in a reasonable amount of time. Therefore, a subset of the users in the training set were selected for the training data. This subset includes all six fraudulent users of each fraud type and 18 randomly selected normal users, *i. e.* the training data set consists of 18 fraudulent users and the same number of normal users. For the unsupervised training, only the non-fraudulent users were used.

8.3.3 Results on pilot test data with supervised training

A mixture model was trained supervised over all pilot test data as described above. The model was then tested on the same dataset, with the results shown in table 8.2. The rows of the table show the classification made by the model, the columns the true value, and the each value show the number of patterns that fall into that category. Thus, if the prediction was perfect, all values except the diagonal would be zero. From the table we can see that the classifier sometimes confuses active and inactive behaviour. This is not much of a problem in this experiment though, since both types are non-fraudulent. The classifications of subscription frauds and excess download are without errors. Some slight mistakes are made for the redistribution fraud, classifying redistribution as normal active behaviour.

8.3.4 Results on pilot test data with unsupervised training

The unsupervised model contained three Gaussians and was trained on every second day of data from all users, with all fraudulent examples removed from the training dataset. The reason for training on every second day of data was to test if the model could generalize the information about the normal distribution to the whole dataset. The model was then tested on the complete dataset, with the results shown in figure 8.1. The upper plot shows the log likelihood of each pattern, while the lower shows the fraud type. The plots show that fraud types 3 and 4 (excess download and subscription fraud) causes significantly lower log likelihood than the non-fraudulent patterns. Redistribution fraud on the other hand does not show any lower log likelihood and could not be detected by the system. This is probably not a problem with the model, but rather a problem with the labelling of the data. The patterns marked as redistribution fraud are based on very basic assumptions that might very well prove to be wrong. In fact, later information suggested that there were probably no real redistribution fraud in the used data set at all.

8.3.5 Results on simulated data with supervised training

A mixture model was trained as described above on the reduced data set. The classification is made for each time step, not one general classification for one user. The results of the classifications are shown in

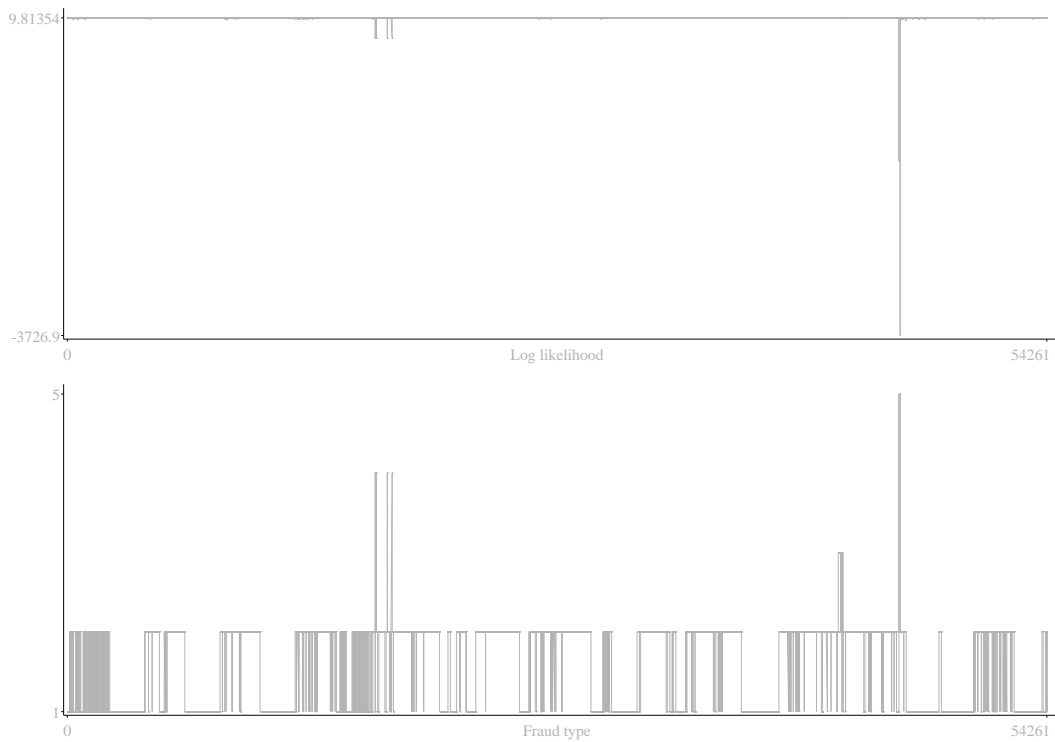


Figure 8.1: Log likelihood and fraud class type for pilot test data

	No fraud	Redistribution	Excess download	Subscription fraud
No fraud	2310670	0	105	56
Redistribution	694	584	2	0
Excess download	0	0	0	0
Subscription fraud	21	0	0	22

Table 8.3: Supervised prediction results on simulated test data

table 8.3. The layout of the table is the same as for table 8.2, except that the active and inactive classes in table 8.2 have been joined to one no fraud class.

If we assume that a user is fraudulent if he or she has been classified as having fraudulent behaviour at any point in time, we can translate the classification per time step to a classification per user. We can then calculate sensitivity and specificity, as described earlier, for all types of fraud. The result is shown in table 8.4.

The table does not show any results for the excess download fraud. It turned out after the experiments that in the test set, all data marked as excess download frauds were incorrectly labelled and should really belong to the illegal redistribution fraud. The results in table 8.4 compensates for this.

8.3.6 Results on simulated data with unsupervised training

The unsupervised model again contained three Gaussians and was trained on the selected normal users. The model was tested on the whole dataset and the results are shown in figure 8.2. As before, the upper plot shows the log likelihood and the lower plot the fraud type, here denoted 1 for no fraud and 2 to 4 for the different fraud types. All fraud types show significantly lower log likelihood in the model. The model also produces some dips in the log likelihood even when there is no fraud, but they are smaller than the ones produced by fraudulent behaviour. This has one exception though. In the beginning of

	Sensitivity	Specificity
No fraud	0.96	1.0
Redistribution	1.0	1.0
Subscription fraud	1.0	0.24

Table 8.4: Supervised prediction results on simulated test data, as sensitivity and specificity.

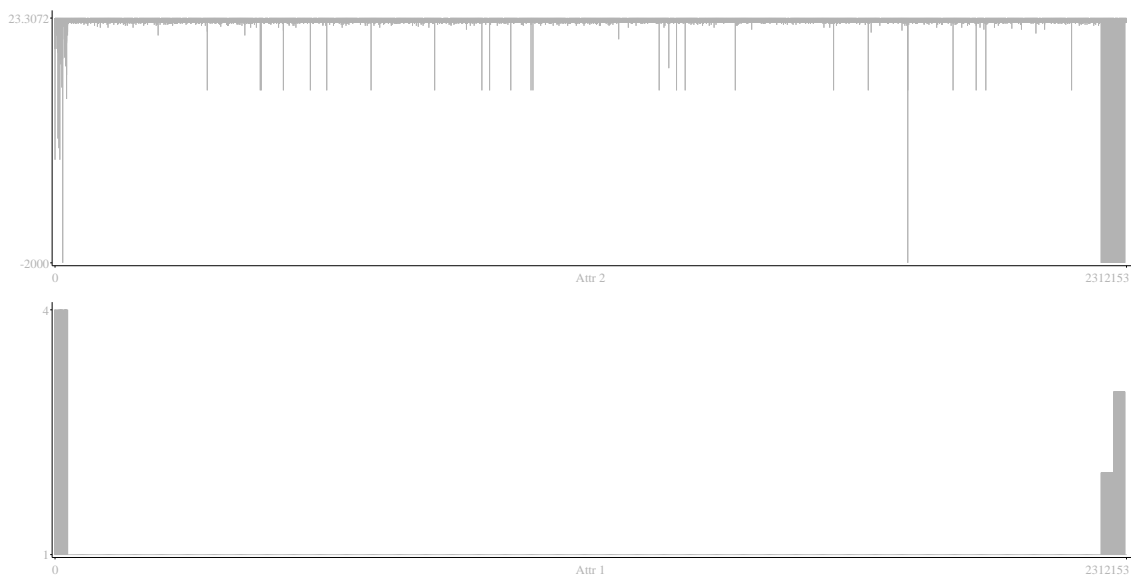


Figure 8.2: Log likelihood and fraud class type for simulated test data

the last quarter of the data there is one large dip in the log likelihood, but the pattern is not marked as fraudulent. The reason is that at that time, the user is billed for a movie that has not been ordered or delivered. This makes the pattern stand out from normal behaviour, although it is not the customer that behaves differently but the movie provider.

8.4 The model used at Halmstad University

Mikael Bodén

8.4.1 Summary

A set of feed forward and recurrent neural networks was trained to continuously classify user actions into four classes: “no fraud” (1), “illegal redistribution” (2), “billing fraud” (3) and “break-in fraud” (4). Recurrent architectures are more suited for the task and outperforms feed forward architectures consistently. The best models are able to correctly classify fraudulent behavior but are unable to distinguish (3) from (2).

8.4.2 Background

From the original 11 input variables, 5 were selected on basis of suspected importance (numbers of ordered, delivered and billed movies, ratio of download/upload multicast data, and ratio of download/upload unicast data).

Fraudulent behavior was suspected to be best identified by taking into account temporally extended patterns of activity. Two strategies for enabling a model to find such patterns were implemented in this study. First, averages of input values were collected for various intervals (1 [no averaging], 10, 50, 100, or 200). That is, for any instance in time the input values were a result of a set of earlier input values. Secondly, feed back activation were available to recurrent networks.

User input data was presented sequentially for 40 steps incremented each instant with the selected interval. Since user activity was limited to about 4000 tuples, only 20 sweeps were used for the 200-interval setup. The starting point was chosen randomly for each user (both for training and testing). The user specific target class was made available to the network at all times.

All networks were equipped with 5 inputs (one for each input variable) and 4 outputs (one for each class, designated to be “true” if activation was 1 and “false” if activation was 0). Two feed forward networks were tested. One single layered network (ffnh0; no hidden units) and one network with 5 hidden units (ffnh5). Three recurrent networks were tested: one with 5, one with 10 and one with 15 fully recurrent hidden units (srnh5, srnh10 and srnh15).

8.4.3 Model adaptation

Training and testing data each consisted of about 500 user activity scenarios. Of the 500, 18 were fraudulent (6 of each fraud class). Only the 18 fraudulent users plus another 30 “legal” users were used for directly adapting weights of the networks (the training set). One user from each fraud class plus another 5 normal users (not the same as in the training set) were selected as a validation set and used for choosing a suitable stopping criterion for training. The networks were trained for a maximum of 100000 users chosen in random order. Tests were performed on the full data set intended for testing.

Weight adaptation was based on the gradient of a maximum likelihood cost function defined over training data. In the case of recurrent networks, the gradient was propagated “backwards” for 5 time steps. The output function on output units in all networks was the so-called softmax function, which normalizes outputs so that they can be interpreted as probabilities. Forced classification was based on the output unit with the highest activation.

Various learning rates were tested. Good performance for all network types was found – with the exception of the single layered network (which never performed well) – when the learning rate was around 0.001 and our study focuses on those results. Other learning rates gave similar performance.

	Sensitivity	Specificity
No fraud	0.98	0.98
Illegal redistribution	0.37	0.27
Billing fraud	0.00	0.00
Break-in fraud	0.51	0.35

Table 8.5: Average performance of all tested recurrent networks.

	Sensitivity	Specificity
No fraud	0.95	0.97
Illegal redistribution	0.14	0.29
Billing fraud	0.00	0.00
Break-in fraud	0.36	0.05

Table 8.6: Average performance of all tested feed forward networks.

	Sensitivity	Specificity
No fraud	1.00	1.00
Illegal redistribution	0.91	0.48
Billing fraud	0.00	0.00
Break-in fraud	0.98	0.83

Table 8.7: Average performance of all recurrent networks trained using averages over 100 time steps.

	Sensitivity	Specificity
No fraud	0.97	0.98
Illegal redistribution	0.47	0.50
Billing fraud	0.29	0.16
Break-in fraud	0.29	0.04

Table 8.8: Average performance of all feed forward networks trained using averages over 100 time steps.

8.4.4 Results

Two performance measures were identified: sensitivity (the ratio between true positives and true positives plus false negatives; informally, the degree of within-class members classified correctly; 1 is best, 0 is worst) and specificity (the ratio between true positives and true positives plus false positives; informally, the degree of misclassifications of outside-class samples; 1 is best, 0 is worst). To test generalization, all measurements were done of the test data.

With no consideration to average interval, recurrent networks outperformed the feed forward networks (see Table 8.5 and 8.6).

Best classification performance was achieved when averages over 100 time steps were used as inputs (see Table 8.7 and 8.8).

The networks confuse class 3 with class 2. None of class 3 members were correctly classified (zero sensitivity of class 3) and about half of those classified as class 2 were misclassifications of class 3 members (about half specificity of class 2). It later turned out that this misclassification was actually due to mislabeled data, see section 8.7.1.

The classification performance on class 1 is excellent. Class 4 is classified correctly most of the time. When misclassification occurs class 4 members still end up as being fraudulent (2).

As only a limited number of simulations were performed, no significant differences were identified between networks with different number of hidden units.

8.5 The model used at DSV

Lars Asker

8.5.1 Summary

Two classifiers were trained to classify the data into four classes. The four classes were labeled “no fraud” (1), “illegal redistribution” (2), “billing fraud” (3) and “break-in fraud”(4). The first classifier used ordered Separate and Conquer (Ordered-SAC), and the second used Divide and Conquer with the Minimum description length principle to select split points (DAC-MDL). The data was divided into one training and one test set. Each set consisted of 518 examples of which 500 were normal users, and 6 from each of the 3 fraudulent classes.

8.5.2 Data preparation

We decided to represent the data by using the summary information that was available for each user. The summary files contained the following information.

- (UID) User id
- (LOC) Physical location
- (SID) Subscriber id
- (FT) Fraud type
- (AC) Activity
- (OR) Total number of ordered films during the interval
- (DN) Total number of delivered films during the interval
- (BN) Total number of films that were paid for during the interval
- (MOR) Maximum number of ordered films
- (MDN) Maximum number of delivered films
- (MBN) Maximum number of films that were paid for
- (TADL) The total average of downloaded bytes
- (TAUL) The total average of uploaded bytes
- (TADLU) The total average of downloaded unicast packages
- (TAULU) The total average of uploaded unicast packages
- (TAUDR) The total average ratio between uploaded and downloaded bytes
- (TAUDUR) The total average ratio between uploaded and downloaded unicast packages
- (AADL) The average of downloaded bytes during active periods
- (AAUL) The average of uploaded bytes during active periods
- (AADLU) The average of downloaded unicast packages during active periods
- (AAULU) The average of uploaded unicast packages during active periods
- (AAUDR) The average ratio between uploaded and downloaded bytes during active periods
- (AAUDUR) The average ratio between uploaded and downloaded unicast packages during active periods

- (NOI) The number of intervals
- (IV) The interval length

The variables containing information about User id, Physical location, and Subscriber id, were removed, and the remaining variables were used to represent the data.

8.5.3 Results

Both methods were able to correctly distinguish between normal and fraudulent users while Ordered SAC incorrectly classified 5 examples of class 3 (billing fraud) as class 2 (illegal redistribution), and DAC-MDL incorrectly classified 6 examples of class 3 as class 2. This fact (together with the results from the other groups) led us to suspect that an error had occurred while labeling the training data (see section 8.7.1). If the labels of the 6 training examples that were labeled as belonging to class 3 were changed to class 2, DAC-MDL classified all the training example correctly while Ordered SAC misclassified one example.

The model created by Ordered SAC could be represented as the following decision list:

```
IF (BN < 4) THEN <class 3>
ELSE IF (AC < 29.6) THEN <class 4>
ELSE IF (TAUL > 8075434.91) THEN <class 2>
ELSE <class 1>
```

where BN = ‘The total number of films that were paid for during the interval’, AC = ‘Activity’, and TAUL = ‘The total average of uploaded bytes’.

The model created by DAC-MDL could be represented as the following set of rules:

```
IF (AAUL =< 2480.58) AND (BN =< 5.5) THEN <class 3>
IF (AAUL =< 2480.58) AND (BN > 5.5) THEN <class 1>
IF (AAUL > 2480.58) AND (AC =< 36.557) THEN <class 4>
IF (AAUL > 2480.58) AND (AC > 36.557) THEN <class 2>
```

where AAUL = ‘The average of uploaded bytes during active periods’, BN = ‘The total number of films that were paid for during the interval’, and AC = ‘Activity’.

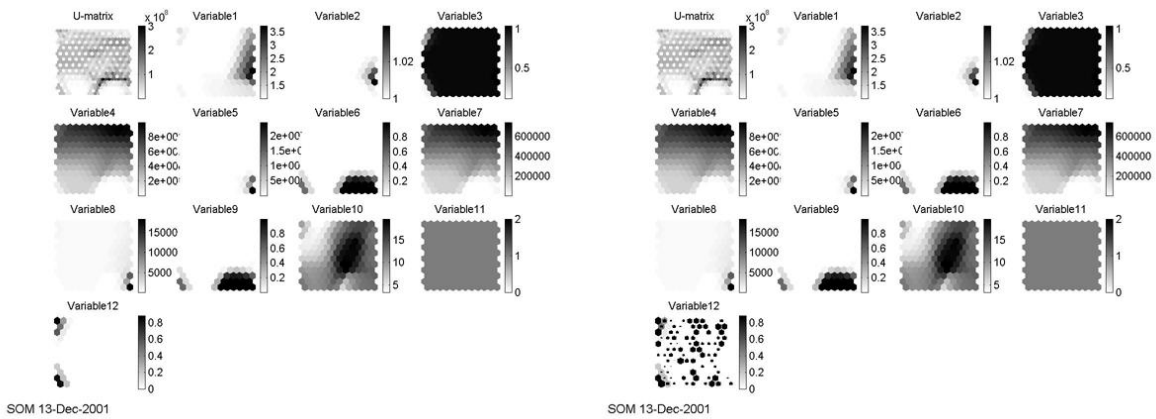


Figure 8.3: There are two maps represented by a U-matrix showing the distance between map units, measured in the input space, and by maps color coded by the values of each input dimension as represented by the map unit prototype vectors. The map to the right has the best-matching units, bmu's, marked. Each sample has a bmu on the map and the size of each marker is proportional to the number of samples it resembles most relative to the other units.

8.6 The model used at Mitthögskolan

Mikael Hall and David Martland

8.6.1 Initial try

The distributed data was quite big, some 5 million samples in total, we only took a randomly chosen subset. Each of the users in the train and the test portion (1036 files in total, with about 4000 samples each originally), contributed 250 samples. The target in this data had one target with range 1-4, representing no fraud, illegal redistribution, billing fraud and break in fraud respectively. This target was preprocessed into an orthogonal form, so that it was represented by four variables, each of range 0-1, where '1' indicates fraud. The training part contained eighteen cheaters, six of each type.

The self-organizing feature maps did not give other useful clues other than that the labelling was done constant for each cheating user, regardless of him/her being active or not. There seemed to be some correlating behavior only in a small subset of the samples labelled fraud. Also the test portion was useless in the sense that it contained the same cheaters as in the train part.

8.6.2 Second try

Because of the problems mentioned above we did some filtering of the data and we changed the goal from modelling to that to illustrate the use of self-organizing feature maps. First, we argued that the billing fraud type is only interesting when the user has received a movie. Thus we only work with samples in which a delivery has occurred. Perhaps we can see if the maps reveal the obvious clue that no payment was done. We also have tried to capture other clues, but due to time shortage, no other filtering has been investigated properly and no other fraud type. The self-organizing feature map without any magnification with regards to the target (billing fraud) is given in figure 8.3. The variables are in ascending order number of movies ordered, number of movies delivered, number of movies payed for, number of bytes downloaded, number of bytes uploaded, upload/download-byte ratio, number of downloaded unicast-packages, number of unicast packages uploaded, upload/download-unicast ratio, hour (0-23), active and a Billing Fraud indicator. The variables are preprocessed to have mean zero and variance one.

As can be seen in figure 8.3, the third variable shows no payment when fraud occurs. We also see that the user of course is active (it is included for a purpose explained below). The two corners at the left in both the fraud and payment variable display fraud. Now look at variable six. It seems that fraud

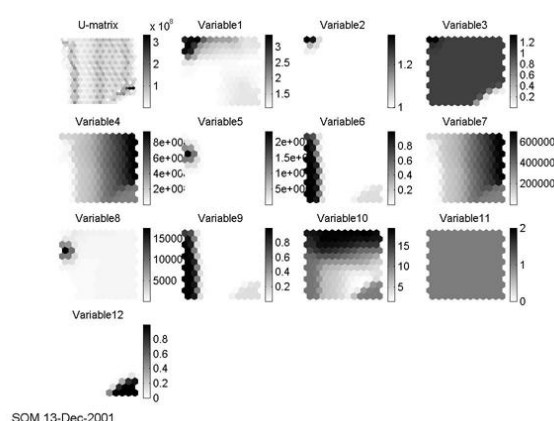


Figure 8.4: Here the target, variable twelve, has been multiplied by two.

occur both when the number of bytes downloaded is mid-high and low. However, by multiplying the target, variable twelve, we may force the map to develop a description of the data which is more tied to fraud behavior. Hopefully this will reveal weaker correlation behavior. We know that a larger part of the map will place them self according to the fraud variable. The other dimensions of the prototype vectors will form a weighted mean, and so describe the mean value of the values in the corresponding input dimension, given the values of target, as well as the neighboring prototype values and so on. Variable six now says that the number of bytes when fraud is committed is medium, so a mean of the previous values is described in the prototype, see figure 8.4. This behavior is important, since we cannot from one map only tell what they describe. By changing the variances we get more information. In this case we did not gain more details about the target, but at other times such gain may result. The target can be neglected to a high degree. By giving it more power (variance), we gain in that the features found must be given by the behavior of the target to a higher degree. Tendencies involving variables with low magnitudes, may appear in a map if they get help to be heard- the map units neglect other samples than before variance adjustment. Of course one would like to further examine the clues obtained, but self-organizing feature maps are still effective in giving initial, although rough, clues to work with. When dimensionality is high, this can lighten the workload and bring the user closer to the data quicker.

We conclude by remarking that the features shown is given within the boundaries given by the samples. A constant dimension stays constant, as can be seen in variable eleven. This means that correlating behavior clues are only given if that variable allows it.

The maps were computed using the SOM Toolbox, which for use with Matlab can be obtained free at <http://www.cis.hut.fi/projects/somtoolbox/>.

8.7 Discussion

Anders Holst and Henrik Jacobsson

8.7.1 Mislabeled test data

As can be seen above, all methods fail to identify class (3), Excess download, in the test set. Most methods seemed to confuse them with class (2), Illegal redistribution. This was considered strange, since there is no similarity between those two fraud types.

After all the experiments had been completed it was discovered that all cases labeled as class (3) actually were generated as class (2) and then mislabeled as class (3). This explains the confusion of the learning systems. When compensating for this mislabeling in the result tables above, it turns out that most of the methods indeed manage to classify the frauds in the test set almost 100% correct.

8.7.2 Real versus simulated data

This task has many aspects in common with for example detecting and diagnosing faults or disturbances in an industrial process. Fraud is, just as a malfunction, hopefully relatively uncommon in the real world situation, at least compared to the vast amount of non-fraudulent cases. This means that even if the total amounts of data seem very large, the number of fraudulent cases may be quite small. Also, since there are so many different ways to commit a fraud, many of which may not even have been conceived of yet, each type of fraud may have occurred only one or a few times. This again is similar to process diagnosis, where the number of ways something can go wrong are almost unlimited, but most of the things that can go wrong has never occurred in the real data.

This is of course very fortunate for the industrial plant or system, but unfortunate for the learning systems. To be able to learn a fraud or a fault, there should be enough examples to allow for a statistical significant analysis of what the relevant features of it are. To provide the learning systems with enough data, it may then be necessary to use a simulator to produce this data. In the simulator any number of serious faults or any conceivable frauds can be produced, without disturbing the real system.

However, simulated data has its own problems. The whole point with the simulator is of course that it should look indistinguishable from real data, from the learning system's point of view. Our experience is that in practice this is never the case. Building a simulator is in the first place a very complex and time consuming task, and simplifying assumptions have to be made all the time. For example, when simulating a process plant, physical entities like pressure and temperature may behave deterministically, like they would in an ideal world, and chemical reactions go with the speeds they ought to go. This means that the simulator is often deterministic – if started the same way the results will look exactly the same. The real world is not deterministic in the same way. Even if all input parameters are set exactly the same, there is a lot of noise. In a simulator the output of a variable should ideally be completely stable during steady state of the process, but in reality all measurements constantly vary, in the best case around a stable value. Also, sensors may not be perfect, but fluctuate somewhat or even drift, so some of the variation may come from the this. Further, dust and dirt may come into the process and disturb the ideal chemical circumstances. Clearly, a learning system trained on the ideal data from such a simulator would not recognize the situations encountered in the real world.

In the situation with frauds, it is even more complicated, since the behavior of human beings must be simulated. This can clearly not be done deterministically, but should be done with some statistical model. Constructing this statistical model to make it realistic is of course very difficult. Real data can be used to estimate the parameters of the model, but the hard task is to figure out which parameters to estimate. And then again, how should the parameters of fraudulent users be estimated, when the number of real cases to estimate them from is so small?

This fraud detection data illustrates these problems in a direct way. Note that, as mentioned above, writing a good realistic simulator is extremely hard, and the simulator here must in light of this be considered a good first step. Nevertheless, we found several discrepancies between the real and simulated data:

- Too homogeneous users.

All users in the simulated data are quite similar, ordering about the same number of movies every week. In the real data there are of course several different user types, some almost never ordering any movies, to those ordering a lot, and those having periods of high activity and are passive in between. If we got it right, this property of the simulator is mainly due to the short time in which it had to be constructed, and it was originally planned that it should use several user profiles. Still, in the general case it is important to span up all possible behavior types if the learning systems should perform well.

- Too clean download and upload activities.

In the simulated data all larger amounts of downloaded of data were due to movies, and all uploaded data due to illegal redistribution of movies. In the real data many users were constantly both uploading and downloading large amounts of data, even without ever ordering any movies. This simplification of course made it easier for the learning systems to spot illegal redistribution, but those same trained systems would not work well if given the real data.

- Differing characteristics of subscription fraud.

In the real data there were a few examples of subscription frauds, seen as a large amount of ordered movies of some normal users, those being subjected to the fraud. In the simulated data on the other hand, this fraud was characterized by a large number of failed order requests, presumably trying to model the impostors failing attempts to log in as something else.

- Unphysical cases.

At least one unrealistic sequence of events was found in the simulated data: According to the data, a user once tried to order a film, but this failed and no delivery notification was sent. Nevertheless, a billing notification was sent and the movie downloaded. This should not happen in reality if the order failed. This was however only seen once, and therefore it could be some error that has occurred somewhere else in the preprocessing, and not in the simulator. But something to watch out for, is that too imprecise statistical models in the simulator may with a small probability produce unphysical or unrealistic data.

More details about the simulated data and the hard problems involved can be found in [Lundin *et al.*, 2001].

8.7.3 Summary

In summary, all results of the Telia project are promising. The systematic misclassifications are due to mistakes in the generation and preprocessing of data rather than due to the models themselves. The fact that the data was simulated does cast a shadow of doubt as to the value of the results, but it is important to remember that real world data would be on a similar format and contain similar characteristics. This is very promising for the future, as the training of the models was not the main problem.

8.8 References

Lundin E., Kvarnström H., and Jonsson E. (2001). Generation of high quality test data for evaluation of fraud detection systems. In *Proceedings of the Sixth Nordic Workshop on Secure IT Systems (NordSec 2001)*. Copenhagen, Denmark, November 1–2 2001.

