Deliverable 3. Knowledge Discovery results.

Evaluation and interpretation.

Grupo de Sistemas Inteligentes

Departamento de Ingeniería de Sistemas Telemáticos

Universidad Politécnica de Madrid.

Project Report

Madrid, October 2012

Authors:

Adrián Pérez Orozco

Álvaro Carrera Barroso

Carlos A. Iglesias Fernández

Executive Summary

Contents

List of Figures

List of Tables

1 Data mining procedure

In previous stages of our project we have performed preliminary analysis on our data (mostly statistical) as well as the necessary preprocessing to apply different learning techniques in the following stages. Once we have completed these tasks, it is now time to perform the Data Mining techniques from which the actual knowledge will be obtained.

As we already mentioned, there are different groups in which we can categorize most of Data Mining procedures:

1. **Classification:** Learning a function that maps an item into predefined classes.
2. **Regression:** Learning a function that maps an item to a predicted variable.
3. **Segmentation:** Identifying a set of clusters to categorise the data.
4. **Summarization:** Finding a compact description for the data.
5. **Association:** Finding significant dependencies between different variables.[3].
6. **Sequence analysis:** Finding frequent sequences or episodes in data [4, 2].

Three of these categories can be useful for our learning objectives. *Sequence analysis* seems the most immediately appropriate category in which our objectives might fall – we have a large amount of sequential data from which we want to obtain patterns which might be useful to make predictions in the future. From these obtained patterns, we may be able to obtain *association* rules – obtain antecedent-consequent pairs from frequent sequences which would associate occurrence of alarms in a given period with the occurrence of other alarms in any other future period.

Alternatively, we can convert our problem into a *classification* task by modelling the alarms of the current period as variables, and the possible alarms for the prediction period as categories to classify our situation into.

The most immediate and convenient (in terms of later implementation and integration) for us is the first approach: sequential analysis from which we will obtain association rules.

1.1 Mining frequent sequences

The first step for our chosen approach is to find frequent sequences in our datasets. Frequent sequences will be good candidates from which we can be able to obtain association rules – if there is an unknown causal relation between two events, they will appear together considerably often. Several algorithms have been developed in the past in order to approach this task of finding frequent sequences. Some examples are the *GSB* algorithm and the *SPADE* algorithm, being the later an alternative to the first with better performance and results.

The procedure of finding frequent sequences in a dataset mainly consists on an iterative analysis of all the possible combinations of elements of the database in sequences. For example, the GSB algorithm can be roughly described as follows:

1. All the possible items (events) of the database are counted. These elements can be seen as sequences of length 1, which will be subsequences of any other larger sequence.

2. All the possible length 2 candidates are generated, as combination of length 1 sequences

3. The database is scanned to calculate occurrence of generated length 2 candidates

4.

2 Evaluation criteria

The predictive information obtained in the data mining process, will lead to the implementation of systems which will give us a prediction using current

events as its input. This prediction will be given in the form of an alarm or set of alarms, which are likely to be raised within a given prediction period. In this section we will approach the problem of *evaluation* of this predictive information.

As a first thought, it might seem appropriate to evaluate our predictions by how true they actually are. We can measure the *accuracy* of a prediction rule system easily by checking how often it becomes true and how often it does not. This is an important factor to take into account, but is however not completely significant of the overall quality of the system. In a limit case in which we only attained a trivial but highly accurate rule which gives valid but trivial predictions all the times, we would have an accuracy of 100%, while the overall quality of the system would be none. We must actually check not only the accuracy of our predictions, but also their relevance against the whole situation.

Therefore, we will need two different evaluation parameters: one related to the accuracy of our predictions, and other related to the fraction of events we are able to predict[1]. In first place, we will define *precision* as the fraction of our predictions which are accurate. In the case of evaluating a rule against a test set, $P_{accurate}$ would be the number of times when both the antecedent and consequent of the given rule have happened within the stipulated time window; while P_{total} would be the number of times when the antecedent of the given rule has happened, whether the consequent has or hasn't happened. Prediction can be as well calculated for a whole rule set, or for any kind of system which gives a predicted event based on other input events.

$$Prec_i = \frac{P_{i,accurate}}{P_{i,total}} \tag{1}$$

On the other hand, we will define *recall* as the relation between events which have successfully been predicted by our system ($E_{predicted}$) and the total number of events (E_{total}).

$$Rec_i = \frac{E_{i,predicted}}{E_{i,total}} \tag{2}$$

Notice that the number of events which have been predicted ($E_{predicted}$) is, in fact, the number of accurate predictions as calculated in the definition of *precision*, ($P_{accurate}$)

In other words, precision is the ratio between accurate predictions and the total number of predictions; while recall is the ratio between accurate predictions and the total number of events.

It is important to notice that in our context, an event can't be *wrongly* predicted. Our prediction can be either true or false, but if we make a prediction of the type $\{A, B\} \longrightarrow \{C\}$ and instead we observe that $\{A, B\} \longrightarrow \{D\}$; it does not mean in any way that we predicted C instead of D, but that our prediction of C was false and we did not predict D. As a result, some other tools generally used to complement values of precision and recall (such as *confusion matrices*) cannot be applied in our case.

Taking a further step, we can merge both indicators in a single one, obtaining a single indicator for a much easier evaluation. Precision and recall are often merged in the called *F-measure*, defined as:

$$F = \frac{(\beta^2 + 1) \cdot Prec \cdot Rec}{\beta^2 \cdot Prec + Rec} \tag{3}$$

where $\beta \in [0, 1]$ balances the importance between recall and precision.

3 Validation

Once the data mining process has been successfully performed, it is of essential importance to *validate* the obtained results. This is, once we have learnt to make predictions based on observation of events, we must actually test how good our predictions are. As we count on a vast amount of data logs, we can easily check our predictions against this historic data. However, if we do this on the same data we have used to obtain this knowledge (our *learning set*) we will obviously obtain extremely good results, as we have already learnt all the patterns happening on that exact data. If we had an ideal, infinite data set with *all* the possible situations that can ever happen in our system, we could have learnt absolutely every possible prediction to

be made on the system and no future event could be *unexpected* to our new prediction abilities. However, in real systems this is not the case, and it is very likely that patterns and characteristics of the systems vary along time.

Additionally, training our system over a single large set of data can lead to *overfitting*. This happens when our predictive knowledge becomes extremely accurate for the set we have been training on, but performs poorly on any other set of events not contained on our learning set. It is important to avoid overfitting by performing learning procedures in a way that not our whole amount of data available is used at the same time. In this direction, the usage of very large data sets for learning procedures can be very inconvenient. In one hand we might be learning patterns which are exclusive to the specific period we are studying (for instance, we may be trying to obtain knowledge from logs from a specific year which we intend to use for forthcoming years), and when we validate this information, we will obtain unrealistic good performance measures.

In order to make a proper validation of the obtained knowledge, we must separate our data in different sets. One of them will be the *learning set* – over which we will work to obtain our predictive knowledge – and the other will be used as a *testing set* – on which we will test our predictive abilities. This way we will obtain a better validation of our predictive knowledge, as the characteristics of the testing set were not taken into account on the learning process, as would happen for any future set of events.

In order to address this problem, one of the most used methods is the k-fold cross-validation (k-fold CV) method. This method consists on dividing the whole data set in k subsets of equal sizes, using *k-1* of them as the learning set and the kth one as the testing set. Performance results are stored for those specific learning and testing sets and the whole process is repeated a total of k times, until all the possible learning sets/testing sets combinations are obtained.

With this process, we obtain a total of k performance testing results for our model. The important point is that all of them have been tested on sets which were not used for their construction. The overall performance measure is obtained as the arithmetic mean of all the individual performance results.

In some cases, we can even randomize the division of the data into subsets, obtaining different subsets for each process of k-fold CV we perform. In our case, however, we are limited in this direction by the nature of our data, as it is very important to preserve sequential information of our data. Our subsets must therefore be conformed of contiguous observations, and cannot be randomized between different temporal subsamples.

A commonly used value for k is 10. As in our case we will generally work with data sets comprising about a year of historic data, this division will provide learning sets of about 9 months and testing sets of about 1 month, which reasonable when validating predictions in terms of days.

References

[1] L Torgo. *Data mining with R.* CRC Press, 2003.

[2] G M Weiss and Others. Predicting telecommunication equipment failures from sequences of network alarms. *Handbook of Knowledge Discovery and Data Mining*, pages 891–896, 2002.

[3] Q Zhao and S S Bhowmick. Association rule mining: A survey. *Available on WWW: http://www. cse. unsw. edu. au/˜ cs9318/readings/ARMining-Survey2. pdf (July 2007)*, 2003.

[4] Q Zhao and S S Bhowmick. Sequential pattern mining: A survey. *ITechnical Report CAIS Nayang Technological University Singapore*, pages 1–26, 2003.