A Non-Sequential Representation of Sequential Data for Churn Prediction

Mark Eastwood, Bogdan Gabrys

Computational Intelligence Research Group,
School of Design, Engineering and Computing,
Bournemouth University
{meastwood,bgabrys}@bournemouth.ac.uk

Abstract. We investigate the length of event sequence giving best predictions when using a continuous HMM approach to churn prediction from sequential data. Motivated by observations that predictions based on only the few most recent events seem to be the most accurate, a non-sequential dataset is constructed from customer event histories by averaging features of the last few events. A simple K-nearest neighbor algorithm on this dataset is found to give significantly improved performance. It is quite intuitive to think that most people will react only to events in the fairly recent past. Events related to telecommunications occurring months or years ago are unlikely to have a large impact on a customer's future behaviour, and these results bear this out. Methods that deal with sequential data also tend to be much more complex than those dealing with simple non-temporal data, giving an added benefit to expressing the recent information in a non-sequential manner.

1 Introduction

In the telecommunications industry, it has been estimated [13] that on average it can cost between 5-8 times more to gain a new customer than it would to keep an existing customer (for example by offering a small incentive). However this incentive is wasted if it is not offered to someone who, in the near future, is likely to churn (that is, to leave the company for a competitor). The high churn rate prevalent in this area means that fairly small improvements in the accuracy of churn prediction can mean significant cost savings. Thus the problem of predicting customer churn is an important one.

It is a very difficult problem. Though we have large quantities of data available, it is limited in that many possible reasons for churn will likely leave no imprint in this data, for example competitor's offers, or changes in personal circumstances.

We can expect, however, that in some cases the reason for the decision to churn will leave an imprint in the data prior to the event. This could be in the form of certain patterns of complaints, or repairs, or other warning signs in the pattern of customer behaviour. In these cases, which we focus on in this paper, we may be able to model and therefore detect situations which will likely result in churn.

The remainder of the paper will be structured as follows. The next section will present the related work which will be followed by the results from an HMM method using different length customer histories, as motivation for the non-sequential representation which will be presented in section 4. This section will also contain results using KNN for churn prediction. The final section will conclude.

2 Related work

At the base of all churn prediction methods is the data used, and here already there are many options. Demographical data (i.e data about the customer) can be used to predict churn, however this may be unsuitable for a number of reasons [12]. Alternatives are call pattern changes and contractual information [12] or customer repair/complaint/provision data [7]. This latter type of data is that used in the current paper.

Neural networks, regression trees and linear regression are compared with regards to their churn predicting potential on repair/complaint/provision data in [7]. The regression tree was found to be most accurate overall achieving 82% correct predictions. However linear regression was the most successful in predicting non-churners whereas the neural network was better in predicting churners. Similar data in a sequential representation encompassing months of a customers historical data is used in a k nearest sequence method in [11] to predict churn, with an improvement found over standard classification techniques which use only the last month of data.

In [12], contractual and call pattern data are used together with a decision tree (C4.5, see [10]) based combination method. The combination method is used to combat the skewed nature of the data; as there are many more non-churn than churn examples, trees are trained on subsets of the training data each of which contains all the churn examples but different samples of the non-churn examples. This gives a number of more balanced training sets on which the trees are trained. The individual predictions are combined via weighted voting. The popular combination methods of bagging [2] and boosting [6] are tested on a mixture of customer and contractual data in [8].

This paper will focus on churn prediction from repair/complaint/provision data. As customers interact with the service provider, certain details of these interactions are logged, and from these we can build customer histories by constructing a sequence of time-ordered events for each unique customer. For the purposes of this paper, each event is described by 5 features. The precise details of the features cannot be given for reasons of confidentiality, but can be described in general terms. The (anonymised) dataset is available on request. One of these features is more naturally categorical; it denotes the event as one of four different types one of which is churn. These categories were expressed numerically for use in a Mixture of Gaussians Hidden Markov Model, or MGHMM (see section 3), the other features are naturally real-valued. One takes positive integer values from zero to a few hundered, two are positive real valued from zero to a few tens, and the final one is real valued with range $\pm$ a few tens about zero.

Common sense suggests that more recent events should be given more weight when trying to predict future customer behaviour. This problem is quite common when dealing with prediction from sequential data; what is the relevance horizon of the data you have? In order to discover the timeframe over which it is best to take events when constructing a customer history, we constructed training sets in which only the most recent

N events are considered, for $N = 3 : 10$. When necessary, a subscript will denote the lengths of sequences allowed, so as an example TR_{any} or TR_N for $N = 3 : 10$.

It was found (see the next section) that models trained on the shortest histories performed best. This motivates the approach taken in section 4, as a short history can be expressed in a non-sequential representation quite easily. This could have applications in any domain where a short relevance horizon applies, especially in the services domain.

3 A sequential HMM approach

One class of method that has seen wide use and success on sequential data are Hidden Markov Models (HMMs) (see for example [5]). For a review of machine learning methods for sequential data see [4]. The simplest form of HMM assumes discrete outputs. For each event only certain discrete outputs can be produced, with the probability of each output depending only on the hidden state of the system. As the data we have consists of four continuous features and one categorical feature, it is more naturally represented in a continuous space so a more flexible model called Mixture of Gaussians HMM (or MGHMM) which allows for this is more useful. I will not describe the method further due to space constraints; the references above contain descriptions of the standard MGHMM we use in this paper.

HMM's will be generated from the customer data, trained iteratively via the usual EM (expectation maximisation) algorithm [1]. Separate models are trained on churn and non-churn sequences, denoted by M_c and M_n respectively, and classification is performed as follows. Given a trained model, the probability that it would generate a given test sequence can be calculated. The sequence can then be classified according to which model has the highest probability of generating it, taking into account the class priors.

These models are highly sensitive to the initialization of the model. One way of reducing this dependency on a specific initialization is to train a number of models using different initializations, and then combine their predictions. We have done this in a relatively simple, rank based manner. For a given individual pair of models M_c, M_n, after calculating for each sequence the probability of churn, the sequences are ranked in order of descending probability. For each sequence, then, we have the ranks $r = r_1, ..., r_N$ where N is the number of models to be combined. We define a function to map this vector of values onto the real numbers, and rank them again according to this new value. We tried a variety of simple functions, and settled on an inverse square function $s = \sum_i \frac{1}{r_i^2}$ though performance is not too sensitive to the form of this function so long as it increaces sufficiently quickly for small r_i.

We then take the top P sequences as our predictions. Here we have a trade-off to decide between. A larger P means we detect more of the actual churn events, but at a higher error rate. This trade-off is summed up in Fig. 1. For example, if we choose to take the top 0.4% as churn predictions (the percentage of sequences which are churn in the training set), we can expect a correct identification rate of just over 0.3. However if we choose to take the top 0.8% as predictions, we can expect to predict more churn events correctly (about 33% more) but at the lower recognition rate of 0.2. The experimental work will be covered in more detail in the next section.

This ability to specify trade-off easily is one advantage of a rank-based approach. Instead of choosing a fairly arbitrary threshold above which we will classify a sequence as in danger of churn, we can specify the level of trade-off we require and allow the data to set the threshold. We could then use this threshold for later classification of single sequences in for example an on-line scenario.

3.1 Results/Discussion

The data used in these experiments was constructed as described in section 2. There are 8080 customer history sequences from which to build the training data, but the final number of training/testing sequences will depend on the restrictions we place on their length. Sequences were split 60-40 into training and testing sets.

In Fig. 1, the performance measure is the fraction of churn predictions which are correct. The basic HMM architecture used was 12 hidden states, with 4 gaussians per state. Transitions depend only on the previous state. The HMM toolbox of Murphy [9] is used to build and train the HMM, and the default training parameters are used, with 12 training iterations. These values were chosen on the basis of preliminary tests. Varying these by a few either way has little effect, with the exception of reducing the number of gaussians below 4, which degrades performance quite markedly. A likely reason for this is that one feature (the event type) takes 4 discrete values, meaning at least 4 gaussians are needed to model the relative probabilities of these in general. Diagonal covariance matrices could have been used in order to reduce the number of parameters to be estimated, however this was not done as the data used is such that there is likely to be correlations between some features.

As can be seen in Fig. 1, the combination method improves performance quite significantly. This serves to illustrate that even quite simple combination methods can provide a large benefit in real world applications. The length of sequence used in the histories can also be seen to have a large impact on performance, with shorter sequences of only the most recent historical events resulting in much better performance. This illustrates a point that it is still extremely important to choose the data correctly and represent it in the most suitable way. It is in this spirit that we will represent the data in a non-sequential manner in the next section.

In order to compare results from the HMM approach with those from the KNN method that follows, we will introduce a new performance measure. Performance will be measured using the following value:

$$G = \frac{p_{churn}}{p_{prior}} = \frac{c_{1|1}(c_{1|1} + c_{1|0} + c_{0|1} + c_{0|0})}{(c_{1|1} + c_{1|0})(c_{0|1} + c_{1|1})}$$

Where $c_{*|*}$ denotes the confusion matrix element, the first subscript being the predicted value (1 for churn, 0 for non-churn) and the second the actual. p_{churn} is the fraction of churn predictions which are correct, and p_{prior} is the prior probability of churn. This is used because, unlike the HMM, there is no natural way of specifying a tradeoff between number of predictions, and accuracy - the KNN method will simply give a set number of predictions. The metric is appropriate to the problem, as it is

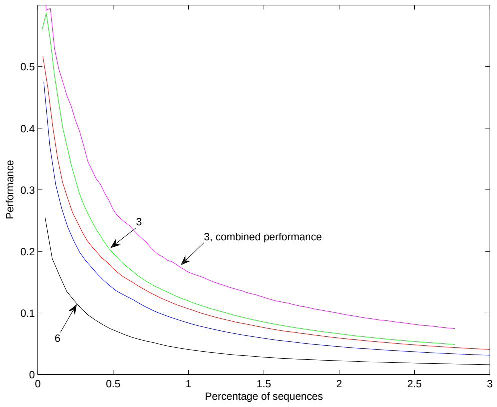

Fig. 1. The top line shows the combined performance using training sequences of length 3. Average performance of individual models plotted against percentage of sequences taken as predictions, for training sequences of length 3,4,5, and 6 are the lower plots (from top to bottom line).

the ratio of the fraction of the methods churn predictions which are truly churn, to the fraction of examples that are churn. Thus if $G = 5$ say, this indicates that using the prediction method we make 5x more correct churn predictions than if we simply made predictions based on randomly predicting an observation to be churn in proportion to the prior probability of churn. It is the proportion of correct CHURN predictions, not the number of correct predictions absolute, which is important.

A further set of experiments was run for the shortest sequences. $Q = 2 : 20$ hidden states in the HMM were used for histories of length 2, for histories of length 3 this was only taken up to $Q = 10$. The dataset was again split 60-40 into training/testing sets and runs over 20 different splits were performed for each Q. The results are shown in Fig. 2. The number of predictions taken to give the g value is the same proportion of the testing set as are churn in the training set. It is also illustrative to look at the confusion matrices corresponding to some specific g-values indicated by arrows in 2.

$$g_1 = \begin{pmatrix} 27 & 46 \\ 85 & 21026 \end{pmatrix} \qquad g_2 = \begin{pmatrix} 13 & 64 \\ 24 & 21128 \end{pmatrix}$$

in the top right

$$g_3 = \begin{pmatrix} 29 & 43 \\ 43 & 21067 \end{pmatrix} \qquad g_4 = \begin{pmatrix} 22 & 54 \\ 54 & 21098 \end{pmatrix}$$

and the bottom two are

$$g_5 = \begin{pmatrix} 19 & 55 \\ 78 & 17700 \end{pmatrix} \text{ and } g_6 = \begin{pmatrix} 24 & 49 \\ 49 & 17903 \end{pmatrix}$$

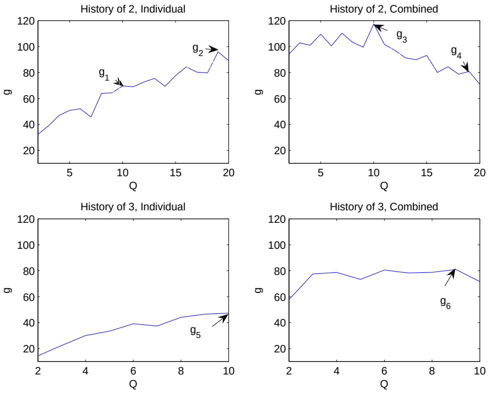

Fig. 2. G value vs Q for individual and combined HMM predictions, using histories as labelled

It can be seen that though the g-value is actually increacing for larger Q and sequences of length 2, few churn predictions are actually being made at higher values (see g_1 compared to g_2) making the model less useful. The drop in g-value for higher Q for the combined method is probably related to this low number of churn predictions made by the individuals, and even though g is lower the combination method keeps the number of correct churn predictions made to a more useful level.

As can be seen again, the combination method clearly outperforms the individual models. The even shorter sequences of length 2 also outperform those of length 3, giving further motivation to our attempt at a non-sequential representation. This is the subject of the next section.

4 A non-sequential KNN approach

Instead of representing the customer history as a sequence, and making the implicit assumption that each event is related to the one before as in a HMM based approach, we may try to represent the data non-sequentially. In this case we make a slightly different assumption, which is that while the decision to churn is based on previous events, the previous events are not necessarily related to each other. Which is truer is debatable, it is easy to imagine scenarios where either could be the case. However there is no doubt that non-sequential data is easier to deal with. All the classical techniques such as KNN, parzen, tree, and support vector classifiers can be used, we chose KNN as an illustration as it performed better in preliminary tests, due to its suitability for problems when the classes are highly imbalanced. This suitability stems from the fact that by choosing K appropriately the number of data points contributing to the classification can be limited so that points of the more prevalent class do not always swamp the minority class. The fact that the churn examples tend to be a little more clustered than the non-churn also contributes to making KNN an appropriate choice.

A non-sequential dataset could be made from the above event histories by either averaging over the events in a sequence for each feature, to give the non-sequential feature values, or by creating new features to represent the features for different events. This latter would give Nk features for sequences of length N and events with k features. It is the first method that we chose to use, though the second may be worth looking at in the future. The new features can be thought of as recording information answering questions like 'were things provided late during the last few events?', 'did the last few events take long?'. This is still highly useful information, what we lose is information on which event, for example, most of the delay/time was due to, or if it was spread over more than one. The second method of creating the new features would retain this information, but at the cost of creating many more features.

We show that when the relevance horizon for a sequential dataset is quite small, it is possible to get good results using classical techniques on a non-sequential representation. Guided by the results in the previous section which pointed to only the few most recent events being relevant, we chose to average features as little information is lost averaging a feature over just a few events.

Our features have some similarities with auto-regressive models. An AR model models a time series entry as being a linear combination of previous entries in the time series, possibly with a noise component:

$$X_t = c + \sum_{i=1}^{t-1} \alpha_i X_{t-i} + \epsilon_t$$

.

c is a constant and ϵ is a white noise component. We use a similar construction, but not in a predictive sense - we rather use it to construct a single feature which is a linear combination of feature entries in a time series:

$$X = c + \sum_{i=0}^{T-1} \alpha_i X_{T-i}$$

where T is the length of the series. Thus far we have used a very simple set of coefficients - the first τ α_i are $\frac{1}{\tau}$, the rest zero. An interesting extension would be to look at other sets of coefficients, perhaps exponentially decaying in timestep.

4.1 Method

The non-sequential dataset is constructed by averaging the features of the last τ events of the sequence, not including the last, label-defining event. The event type of this last event is used to label the data point as churn or non-churn. Only 3 features were included. These are event type (churn, complaint, repair and order are given the values 1,2,3 and 4 respectively), event duration (can be zero if not known or is not applicable), and promise (if something was promised, how early it was achieved; it is negative if that something was late. It can be zero if not relevant). These were chosen from a common sense view of what factors would be most likely to influence someone to churn, and from the results of preliminary experiments.

This dataset is split 60-40 into training and testing sets, and a simple K nearest neighbor algorithm is used to perform the classification. A nearest neighbor algorithm was chosen as it deals well with datasets such as this where the prior probabilities of the classes are highly imbalanced. The HMM is very computationally expensive to train, but it is cheap to calculate predictions when trained. In comparison, the KNN costs nothing to train, however it can be very expensive to calculate very large numbers of predictions. There are many methods available in the literature to increase the efficiency of such nearest neighbor searches though, for just one example see [3]. Results, and some discussion and interpretation, follow.

4.2 Results/Discussion

The first three subplots in Fig. 3 show the results on datasets averaging the features of 1,2 and 3 events respectively, using 1-7 nearest neighbors for classification. The next event in the sequence is predicted. Performance is measured using the g value presented in section 3.1.

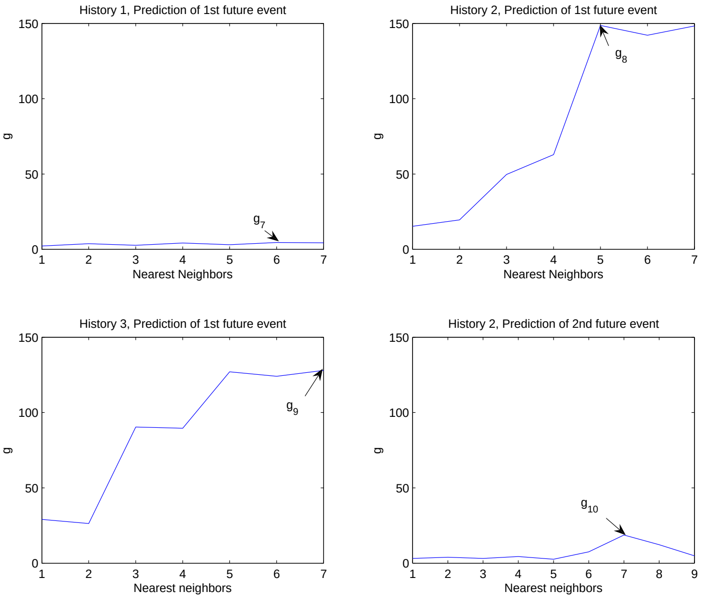

Fig. 3. Performance vs NN for histories and prediction time frame as labelled

The final subplot shows the performance when trying to predict two events into the future, using a history of 2 and 1-9 nearest neighbors. This can be seen to be much less successful, showing that knowledge of the most recent event is very important. Using histories of other lengths to predict two events into the future also results in bad performance, and so is not shown.

From the above figures, it can be seen that an event history of 2 gives optimal performance using this method, and that taking simply the last event is totally inadequate for prediction. This shows that it is necessary to take into account the sequential nature of the data, even if only over a short time. An event history of 3 performs well, but worse than the shorter time period. This shows that the most relevant events in a customers history are the last two or three, as intuition would support. Also the non-sequential representation is less suitable for longer sequences. Performance does not seem to be overly dependant on NN number, with a nearest neighbor count of over 5 performing well.

Again looking at the confusion matrices gives a little more insight. From top left to bottom right, the g values indicated are:

$$g_7 = \begin{pmatrix} 26 & 40 \\ 1278 & 13711 \end{pmatrix} \qquad g_8 = \begin{pmatrix} 28 & 36 \\ 16 & 14866 \end{pmatrix}$$

$$g_9 = \begin{pmatrix} 9 & 57 \\ 7 & 14948 \end{pmatrix} \qquad g_{10} = \begin{pmatrix} 3 & 51 \\ 34 & 12384 \end{pmatrix}$$

From g_7 we can see that although prediction from the last event detects churn quite well, there are very many false positives too resulting in a low g. From g_8 we see that including an extra event into the history has little effect on the number of correct churn predictions, but vastly reduces the number of false churn predictions, improving both specificity and sensitivity thus making this a much more useful tool for practical churn prediction. Looking at the confusion matrix g_3 for the HMM, we see that there is a decrease in false churn prediction compared to this too, while maintaining a very similar level of correct churn prediction. A third event in the history can be seen in g_9 to reduce churn predictions markedly, both correct and false. This lowers the sensitivity drastically, and in this case the number of churn predictions made is too low to be really useful, compared to using just 2 event histories.

Trying to predict more than one event into the future can be seen to result in very few churn predictions.

We can attempt to interpret what these results could mean in real terms by looking more closely at the nature of the data we have. Roughly half of all the churn examples correspond to sequences in which the last two events are complaints, which is revealing in itself, although it shouldn't really be surprising. Almost all the churn examples correspond to event sequences in which the last two events have been of the same type, and many of them where the last two events are quite similar. These observations could be interpreted as indicating that a customer does not like to have to do the same thing twice when dealing with the service provider, especially when that thing is a complaint. Churn examples are also quite closely clustered, indicating that complaints falling into a few distinct, well defined subclasses may be especially likely to provoke a churn response.

5 Conclusions

We have proposed, based on observations from a HMM method, that only the most recent events in a customers history have an effect on the future behaviour of that customer, and shown that a short sequence of events corresponding to a recent history can be represented easily in a non-sequential way. This allows the use of all the tools available for simple, non-sequential pattern recognition, and we show that a K-nearest neighbor algorithm performs well on this data. This provides much better performance and potentially reduced computational complexity over the HMM methods. We explain the success of this method by noting that many churn events when represented in this way lie in a few small, dense clusters, and observe that many churn events follow a history of two events of the same type, often with similar feature values. This indicates that perhaps having to do the same thing twice, especially with regards to a complaint, often leads to churn.

References

1. J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models, 1997.
2. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
3. Yong-Sheng Chen, Yi-Ping Hung, Ting-Fang Yen, and Chiou-Shann Fuh. Fast and versatile algorithm for nearest neighbor search based on a lower bound tree. *Pattern Recogn.*, 40(2):360–375, 2007.
4. Thomas G. Dietterich. Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30, London, UK, 2002. Springer-Verlag.
5. Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. John Wiley and Sons, 2001. 0-471-05669-3.
6. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.
7. John Haddon, Ashutosh Tiwari, Rajkumar Roy, and Dymitr Ruta. Churn prediction: Does technology matter? 2006.
8. Aurelie Lemmens and Christophe Croux. Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, XLIII:276–286, 2006.
9. Kevin Murphy. A hmm toolbox for matlab, available at http://www.cs.ubc.ca/ murphyk/software/hmm/hmm.html.
10. J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
11. Dymitr Ruta, Detlef Nauck, and Ben Azvine. K nearest sequence method and its application to churn prediction. In *IDEAL*, pages 207–215, 2006.
12. Chih-Ping Wei and I-Tang Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert Systems with Applications*, 23:103–112, 2002.
13. L. Yan, D. J. Miller, M. C. Mozer, and R. Wolniewicz. Improving prediction of customer behaviour in non-stationary environments. *Proc. of Int. Joint Conf. on Neural Networks*, pages 2258–2263, 2001.