In Workshop on Data Mining for Medicine and Healthcare at the 14th SIAM International Conference on Data Mining, pp. 82-87, 2014.

Learning Decision Lists with Lags for Physiological Time Series

Erik Hemberg* Kalyan Veeramachaneni† Prashan Wanigasekara‡ Hormoz Shahrzad§
Babak Hodjat¶ Una-May O'Reilly‖

Abstract

Increasingly large volumes of time series medical data need to be exploited to learn better models that forecast events such as acute hypotensive episodes (AHE). The models have to be transparent so clinicians can decipher and validate them with their expert knowledge. The nature of learning a forecasting model requires historical "lag" data but in most cases the extent of the relevant lag is open to question and the cost, with respect to model learning, increases with its duration. We present a novel decision list-based machine learning approach for forecasting physiological time series by classification. It is scalable, finds lag duration automatically and the rules it learns are interpretable and compact in terms of their representation of lagged variables.

1 Introduction

In the medical domain, data mining is more and more focused on longitudinal studies where the ability to track patient medical data over time is of primary interest. A large number of patient records are time series based. Some are at the granularity of high resolution physiological waveforms recorded in the ICU or via the remote monitoring systems. Others include lab work done at (*ir*)regular intervals time. In addition, data warehouses containing electronic health records or insurance claims store the data from multiple visits of patients at different points in time.

With the availability of these rich and heterogeneous time series data sources about patients' medical history, a number of studies have focused on building predictive models. Consider two recent examples. The first study focused on generating an alert for *cardiac arrest and resuscitation* using demographic information, signals from history, vitals and laboratory measurements [8]. The authors call the event *code blue* and aim to predict the possibility of its occurrence for a patient in the next 1h, 2h, 3h, 4h. The second study aims at early detection of diabetes from health claims [4]. From a patient's claim history they extract approximately 1054 features to predict onset of diabetes in a future time window.

Given a time-series of training exemplars each of length T (in samples), to build a discriminative model capable of predicting an event, features are extracted by splitting the time series into non-overlapping (or overlapping), segments of size k samples each, up to a certain point $h < T$ such that there are $m = \frac{h}{k}$ segments. A number of aggregating functions are then applied to each of these segments (a.k.a windows) to generate features for the problem. For example, consider the two papers we just mentioned. In the first paper, which predicts *code blue*, the time series is divided into three segments starting from the current time, t, corresponding to $t < -3hr$, $-3hr < t < -9hr$ and $-9hr < t < -18hr$. In each segment, the authors apply a number of aggregation functions over the samples from the time series to generate features. These are *minimum*, *maximum*, *average* and *standard deviation* for each vital and lab value. They also measure the slope of the signals in those segments by fitting a linear regression. This gives them 327 features. They then learn a classifier with these features and a label.

In the second paper, which detects diabetes, the feature vector for an event at time point t was created by considering the entire history of the patient up to t as a segment. First, a class of medications and ICD-9 codes were assembled as possible indicators of looming diabetes onset. A number of binary valued features were then extracted from the patients history where a value 1 indicated at least one instance (claim) of the patient being associated with the ICD-9 code or the class of medications. A discriminative model was built using regularized logistic regression.

In these approaches, there are several decisions one has to make: first, the size of the segments (a.k.a window size), k, the amount of history h to be used, and the aggregation functions need to be chosen. The best values for any of these parameters is not known a priori. Additionally, since dimensionality of the problem is $= \frac{h}{k} \times nf$, where h is the amount of history used, k the segment size and nf is the number of aggregation functions, it is often hard to justify use of large history

*CSAIL at MIT, Cambridge, MA 02139 USA
†CSAIL at MIT, Cambridge, MA 02139 USA
‡CSAIL at MIT, Cambridge, MA 02139 USA
§Genetic Finance, CA 94105 USA
¶Genetic Finance, CA 94105 USA
‖CSAIL at MIT, Cambridge, MA 02139 USA

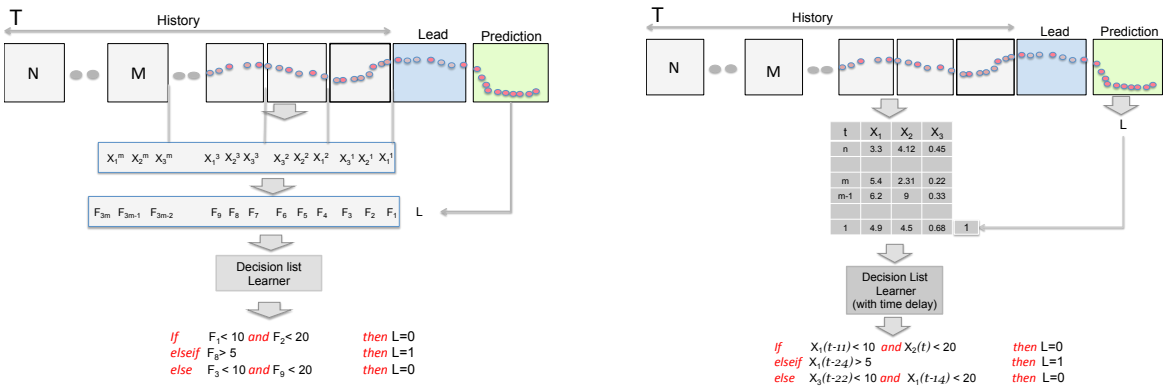

Figure 1: Two alternative approaches to learning interpretable models from time series. In the more conventional approach, left, a time series is divided into segments and three features are extracted from each segment resulting in a total of 9 features. The learner, a classification algorithm, learns a set of rules that considers these features. At right the time series itself is passed to the learning engine with a label attached to it. The learning engine produces a rule list with conditions applied to variables that reference values of the time series in the past.

or small segment size due to its effect in increase in the dimensionality. Increasing the dimensionality requires either regularization or feature selection to build robust discriminative models. Additionally, without regularization or feature selection the models could loose interpretability as well. In this paper we present an alternative representation for models and rely on the learning algorithm to identify the amount of history to be used automatically for every problem. The algorithm is given the aggregated time series.

The approach evolves interpretable decision lists. Figure 1 contrasts the two methods side by side, the entire time series up to T is split into N segments. 3 aggregate functions are applied to extract features from each segment. In the first approach, left, features are assembled by collecting the aggregate values for the m segments. This results in $3m$ features that are fed into a standard machine learning algorithm. In the second approach, right, multiple time series generated by applying aggregate functions to the entire N segments are fed into the machine learning algorithm we describe in this paper. The algorithm generates the decision list that makes use of the lagged values of the time series by generating conditions with variables of the form $X_i(t - \Delta)$, where $0 \leq \Delta \leq N$. The algorithm learns Δ which we call *lags* (a.k.a delays) as part of the model.

The learning methodology is based on a search and score methodology by referencing a large dataset through a large scale, distributed algorithm is employed, EC-Star [6], see Section 2. Important properties of this system are its ability to find lags automatically and its scalable distribution which breaks the data into small packages and scores a decision list on many asynchronous learners in parallel. We demonstrate our approach with time-series classification of arterial blood pressure (ABP), see Section 3. Our particular area of investigation is "acute hypotensive episodes" [5]. Finally, we conclude with a discussion regarding the conclusions and future work in Section 4.

2 Learning decision lists with time delays

In this section we present a brief overview of our learning algorithm called EC-Star, representation of the model it generates, and the representation of data fed to it.

2.1 Model and data representation A decision list in EC-Star is similar to a decision list [7], each rule is a variable length conjunction of conditions with an associated class prediction, see at the bottom of the right part of Figure 1. In the evaluation each condition compares a lagged value or the current value of the time series to a threshold (decision boundary). The decision lists in EC-Star have a variable number of rules and conjunctive clauses in each rule, but are limited by *max decision list size*. This representation is different from many other classifiers e.g. DecisionTrees, simple Decision Lists, Support Vector Machines and Logistic Regression, which requires every time lagged value or an aggregate to be set as a different feature.

Furthermore, the EC-Star algorithm requires a specific layout of the data. The data is assembled as data packages, where each data package is a classification example. Consider two time series $x_1(t)$ and $x_2(t)$. Within each data package for each time interval $t = a$ the values of $x_1(a)$ and $x_2(a)$ are stored as columns. This is

shown below in Table 1. If the problem has more time series additional columns can be incorporated into the data package. Each data package is associated with a label l.

The rule is evaluated for each data package and its error rates, false positive and false negatives are calculated by accumulating the discrepancy between its predicted label and the true label for the data package. Table 1 presents a rule and its prediction for a data package.

2.2 Learning algorithm To learn a best fit decision list EC-Star [6] performs an iterative parallel stochastic search over decision lists, sometimes referred to as "evolutionary search" [1]. The algorithm utilizes a population of decision lists and slightly alters a number of factors in the decision lists. These are the conditions used in a rule, the number of conditions in a rule, the lags in the rule, and number of rules in each of the decision lists. These changes are made via commonly known computational procedures *crossover* and *mutation* (a.k.a evolutionary operators). In our current version of the algorithm, the condition thresholds are preset prior to the search itself. In each iteration the decision lists with the top scores are selected and are used as the seed for next iteration for alteration. This approach relies heavily on being able to evaluate each decision list in the population on the data set, thus requiring multiple passes through the data within each iteration. For large datasets, as in this paper, this is prohibitive to carry out on a single node. Next we give a brief overview of our distributed architecture.

2.3 Distributed architecture In EC-Star everytime a decision list is changed it needs to be re-evaluated on all the data packages. The EC-Star system uses a hub-and-spoke architecture for a distributed and asynchronous evaluation of decision lists. The learner coordinator server is the hub, which maintains an archive of decision lists with different scores and passes the high ranking solutions thus far to the local learners.

The local learners have two functions. First, they receive the best ranking decision lists from the learner coordinator server and iteratively alter them using the evolutionary operators. During each iteration they evaluate the decision lists by uniformly sampling a subset of EC-Star data packages with replacement, from a separate data package server. After a certain number of iterations they pass their best decision lists thus far to the learner coordinator server.

Second, the learners evaluate the decision lists they received from the learner coordinator server on additional EC-Star data packages and update the scores

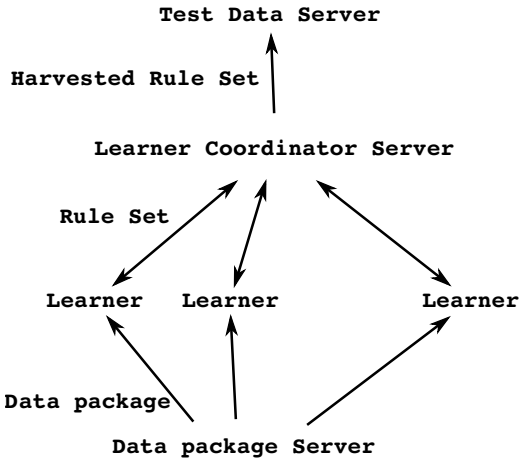

Figure 2: EC-Star architecture. learner coordinator server distributes decision lists to the learners on the network. The learners request EC-Star data packages from the data package server. decision lists are harvested from the learner coordinator server and evaluated on out-of-sample test data.

for these decision lists and report it back to the learner coordinator server.

The learner coordinator server from time to time updates the ranking of the decision lists based on the updated scores. Although the comparisons at a learner during iterative search are noisy due to evaluation on partial data, it enables EC-Star to handle large volumes of data by using fewer exemplars for training and removing poor decision lists early. The learner coordinator server emerges with solutions that perform well across the data as time progresses. This distribution of evaluation, model generation and comparisons allows EC-Star to operate on large amounts of data with a pool of large number of learners.

3 Demonstration- Arterial Blood Pressure Event Prediction

In this section, we employ the algorithm and its ability to learn transparent decision lists from the time-series to acute hypotensive event prediction in the ICU. We define a time series prediction problem based on *lead*, and *prediction duration*. Consider the mean arterial pressure signal at time t as $x(t) \in \mathbb{R}$. The goal is to predict if the value of the statistic $\overline{m} = mean([x(t+\alpha), \ldots, x(t+\alpha+\beta)])$ falls into one of the three intervals:

$$label = \begin{cases} \text{Low} & \text{if } \overline{m} \leq 55mmHg \\ \text{Normal} & \text{if } 55mmHg < \overline{m} \leq 85mmhg \\ \text{High} & \text{if } \overline{m} > 85mmhg \end{cases}$$

The parameter α is the *lead* time and β is the prediction window duration. The history, γ is the period

Nr		Condition 1		Condition 2	Action
1	if	$x_1(T) < 10$	and	$x_2(T-1) > 20$	then $l = 0$
2	if	$x_2(T) < 10$	and	$x_2(T-3) > 20$	then $l = 1$
3	if	$x_1(T-1) < 10$			then $l = 0$

T	$x_1(T)$	$x_2(T)$	l
0	10	20	
1	10	32	
2	9	30	
$\underline{3}$	8	20	0

Table 1: Example of a decision list (top) and a data package (bottom). Demonstration of evaluation of a rule on a data package. The example shows how the current Time, $T = \underline{3}$ is applied to each rule. Rule nr 1 evaluates the first clause with x_1 at the current time to true (green). The second clause of rule 1, x_2 at current time -1, $T-1 = 2$ is also true. Rule nr 1 applies label $l = 0$ (grey) as the action, which matches the label for $T = 3$. The first clause in rule nr 2 compares x_2 at $T = 3$, which is false (red). The second clause compares x_2 at $T-3 = 0$, which is again false. Thus, the rule takes no action. The third rule compares x_1 at time $T-1 = 2$ and is true and takes the action $l = 0$, which is correct. When there is more than one prediction the current heuristic for choosing the action is to take the first prediction, in the same manner as a decision list. Thus, the action from rule nr 1, $l-0$, is predicted, which is correct. If no rules are true then the action will be "Null", which is always incorrect.

$t-\gamma$ that decision list can access when defining the lag for a variable.

We demonstrate the efficacy of the approach on roughly 4000 patients ABP waveforms from MIMIC II v3. In MIMIC Waveform records available are sampled at 125Hz (125 samples/second) [2] and ABP is recorded invasively from one of the radial arteries. The raw data size was roughly 1 Terrabyte. We employ a number of preprocessing steps before employing our learning engine.

Step 1: Conditioning For each patient's waveforms, we extracted beat onsets and extract data per beat. We then checked whether the beat is valid or not. There were roughly 1 billion beats.

Step 2: Extracting MAP time series From the periodic waveform we then generate an aperiodic waveform of mean arterial pressure (MAP). Each sample in this waveform is generated by taking the mean arterial pressure per beat.

Step 3: Creating aggregated feature time series We then generate four time series by applying four aggregate functions to the MAP time series and choosing an aggregation window of 1 min. The four aggregate functions used are: (1) Mean MAP, (2) Std of MAP, (3) Kurtosis of MAP (4) Skew of MAP. Applying these four aggregate functions generates four aperiodic time series for our problem numbered $x_i(t)$ where $i \in \{1 \dots 4\}$.

Step 4: Forming the training exemplars We then form a repository of time series segments that are at least 120 *minutes* in duration. We set the event definition time of 1 *minutes* i.e. $\beta - \alpha = 1$ the lead time of $\alpha = 20$. The segments are then split into data packages with 100 lines each.

The labels in the data are imbalanced, the total number of Low event are just 1.9% of the total number of events. In total we had 64,123 EC-Star data packages from 4,414 patient records. 10% of the data is withheld, the rest is used for training with 10 fold cross validation. The largest number learners used was 3,000.

For the results presented in this paper the decision list after 2h of computation using 4 learners is tested on out-of-sample data. We perform 6 independent runs for each fold. The quality of the decision list is determined by the weighted error (WE), L is the set of labels, $C_{i|j}$ is cost of predicting label $i, i \in L$ as $j, j \in L$, $p_{i|j}$ is probability of predicting label i when it is j.

$$WE = \sum_{j]in L} \sum_{i \in L} C_{i|j} p_{i|j} \tag{3.1}$$

The cost is

$$C = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ Null \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 500 & 0 & 1 \\ 600 & 1 & 0 \\ 600 & 1 & 1 \end{pmatrix} \end{array}$$

Table 2: Results from out-of-sample test of the best decision lists for Low against all other labels. Test data is the best solution from the withheld data. Fold average is the mean±std over the folds. WE is the Weighted Error (Eq. 3.1), TPR is the True Positive Rate (TP/(TP+FN) and FPR is the False Positive Rate (FP/(FP+TN). TP is True Positive, TN is true negative, FP is false positive and FN is false negative.

Name	WE	TPR	FPR
Fold average	1.251±0.250	0.906±0.035	0.208±0.091
Withheld	0.895	0.964	0.364

3.1 Results The data is unbalanced with a majority of events labeled as Normal or High blood pressure and very, very few as Low blood pressure. We are foremost interested in for Low (hypotension) against all other labels.

The results are shown in Table 2. There is a high variance on the false positive rate among the 10 folds. Our performance on 10 fold cross validation is similar to the best decision lists performance on the test data.

While the false positive rate seems high, we note that four time series used in our experiments were derived from only one attribute per beat - mean arterial pressure. In our current work we extract multiple attributes per beat like *systolic* and *diastolic* pressures, pulse duration, area under systole. With this additional information about the patient state, our preliminary studies show that our false positive rates are significantly lower.

An advantage of the representation used in EC-Star is the search over the lags and data size. Let $L \in \mathbb{N}$ be the max lag, $n \in \mathbb{N}$ be the number of features and $m \in \mathbb{N}$ be the number of exemplars.

Number of features EC-Star searches among n features for each lag, which in total is nL.

Data size The size of the data is constant for EC-Star, $n \times m$. For the conventional fixed lag approaches the smallest data size is $n \times m$ when no lag is used and the data size grows linearly with the lag to $nL \times m$. In adition, the the conventional fixed lag approach creates $n(L-1) \times m$ redundant data entries.

Data packaging The data ordering of the data in EC-Star makes it non-redundant. The minimum data size of a data package, i.e. input data to the algorithm, in EC-Star is $n \times L$ and the conventional fixed lag has a minimum size of $nL \times 1$.

3.2 Rule set example In Table 3 a decision list is shown. In summary when reading the rule in Table 3 there are three simple ordered rules: *1*) if the current $Mean$ is below $72.75 mmHg$ (Rule 3) the classification is Low *2*) if $Mean$ is between $73.1 - 121.96 mmHg$ 1 minute back and above $88.94 mmHg$ 4 minutes back the classification is High. *3*) if it is between $72.75 - 97.53 mmHg$ the classification is Normal. Other rules are based on the less intuitive Std, $Kurtosis$ and $Skew$.

The decision list has 16 rules and 28 clauses. On closer inspection we see that there are 7 rules with 1 clause, 6 with 2 clauses and 3 with 3 clauses. The most used feature in the clauses is $Mean$, there are 14 clauses with $Mean$, 2 with Std, 7 with $Kurtosis$ and 5 with $Skew$. This shows that when predicting the future of MAP the past values of MAP are informative. Similarly, lag of 0 is the most frequently used: 10 conditions have delay 0, 2 have delay 1, 3 have delay 3, 5 have delay 4, 2 have delay 5, 1 has lag 7, 8 and 9, and 2 have lag 10. In addition, the labels for the actions are roughly equal, there are 6 label 0, 5 Normal and 5 Normal. The importance is the ordering of the rules, and there are more Low in the earlier rules. There are some rules which can be pruned by the learning algorithm.

4 Discussion

We have begun to design a methodology to derive interpretable models based on decision lists with variable *lags* for time-series classification problems. Furthermore, our approach is based on a large scale iterative search that can scalably learn from the increasing volumes of physiological (or other) time-series data and find variable *lags* for the transparent models.

We demonstrated the EC-Star approach and its ability to find variable "lags" in a challenging acute hypotensive prediction problem. The approach does not need transformation of the data. In addition, a data-driven approach can be useful in interpreting and understanding the progression of patients health and early indicators of problems. However, our demonstration is currently for only physiological time series. There exist a plethora of time series problems and we are currently pursuing a number of different problems to validate our approach and aid in knowledge discovery from time series data. These include time series repositories in electronic health records and insurance claims.

References

[1] Thomas Back, David B Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[2] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G

Table 3: decision list from out-of-sample test. E.g. Std(T-4) means that the standard deviation of the aggregated MAP over 4 minutes earlier was used for the classification. All the features are extracted from aggregations of MAP

Nr	*Clauses*						*Action*
1	if	$Mean$(T-4) $< 72.75mmHg$	and	$Kurtosis$(T-3) < 4.09			then Low
2	elseif	$Skew$(T-10) > 2.01	and	$Mean$(T-8) $< 88.92mmHg$	and	$Skew$(T-4) < 0.15	then Normal
3	elseif	$Mean$(T) $< 72.75mmHg$					then Low
4	elseif	$Mean$(T-10) $< 73.10mmHg$					then Low
5	elseif	$Mean$(T-1) $< 121.96mmHg$	and	$Mean$(T-4) $> 88.92mmHg$	and	$Mean$(T-1) $> 73.10mmHg$	then High
6	elseif	$Mean$(T) $< 97.53mmHg$					then Normal
7	elseif	$Mean$(T) $< 97.53mmHg$	and	$Kurtosis$(T) > 12.71			then Normal
8	elseif	$Mean$(T-4) $< 72.75mmHg$	and	$Kurtosis$(T-7) > 4.03			then Low
9	elseif	$Mean$(T-4) $> 121.96mmHg$	and	$Kurtosis$(T-5) > 12.71	and	$Kurtosis$(T-3) > 1.00	then Normal
10	elseif	Std(T) < 10.76					then High
11	elseif	$Kurtosis$(T) > 1.00					then High
12	elseif	$Mean$(T) $< 72.75mmHg$	and	Std(T-4) > 0.01			then Low
13	elseif	$Kurtosis$(T) < 4.09	and	$Skew$(T-3) > 2.01			then Normal
14	elseif	$Skew$(T-9) > 0.06					then High
15	elseif	$Skew$(T) < 1.95					then High
16	elseif	$Mean$(T) $< 72.75mmHg$	and	$Mean$(T-5) $< 52.12mmHg$			then Low
17	else						then None

Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[3] Babak Hodjat, Erik Hemberg, Hormoz Shahrzad, and Una-May OReilly. Maintenance of a long running distributed genetic programming system for solving problems requiring big data. In *Genetic Programming Theory and Practice XI*. Springer, 2014.

[4] Rahul Krishnan, Narges Razavian, Youngduck Choi, Somesh Nigam, Saul Blecker, Ann-Marie Schmidt, and David Sontag. Early detection of diabetes from health claims. In *Machine Learning for Clinical Data Analysis and Healthcare NIPS Workshop 2013*, 2013.

[5] George B Moody and LH Lehman. Predicting acute hypotensive episodes: The 10th annual physionet/computers in cardiology challenge. In *Computers in Cardiology, 2009*, pages 541–544. IEEE, 2009.

[6] Una-May OReilly, Mark Wagy, and Babak Hodjat. Ec-star: A massive-scale, hub and spoke, distributed genetic programming system. In *Genetic Programming Theory and Practice X*, pages 73–85. Springer, 2013.

[7] R.L. Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

[8] Sriram Somanchi, Samrachana Adhikari, Allen Lin, Elena Eneva, and Rayid Ghani. Early code blue prediction using patient medical records. In *Machine Learning for Clinical Data Analysis and Healthcare NIPS Workshop 2013*, 2013.