DEEP LEARNING ON SYMBOLIC REPRESENTATIONS FOR LARGE-SCALE HETEROGENEOUS TIME-SERIES EVENT PREDICTION

Shengdong Zhang, Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, Mohak Shah

Research and Technology Center, Robert Bosch LLC, Palo Alto, CA

ABSTRACT

In this paper, we consider the problem of event prediction with multi-variate time series data consisting of heterogeneous (continuous and categorical) variables. The complex dependencies between the variables combined with asynchronicity and sparsity of the data makes the event prediction problem particularly challenging. Most state-of-art approaches address this either by designing hand-engineered features or breaking up the problem over homogeneous variates. In this work, we formulate the (rare) event prediction task as a classification problem with a novel asymmetric loss function and propose an end-to-end deep learning algorithm over symbolic representations of time-series. Symbolic representations are fed into an embedding layer and a Long Short Term Memory Neural Network (LSTM) layer which are trained to learn discriminative features. We also propose a simple sequence chopping technique to speed-up the training of LSTM for long temporal sequences. Experiments on real-world industrial datasets demonstrate the effectiveness of the proposed approach.

***Index Terms*—** Event Prediction, Feature Discovery, Recurrent Networks

1. INTRODUCTION

Collection of raw time series data and system logs has become quite prevalent recently owing to increased connectivity of physical systems to the Internet. Such datasets enable applications such as predictive maintenance, service optimizations and efficiency improvements for physical assets. At the same time, these datasets also pose interesting research challenges such as complex dependencies and heterogeneous nature of variables, non-uniform sampling of variables, sparsity, etc which further complicates the process of feature extraction for data mining tasks.

Feature extraction from time-series data for classification has been long studied [1]. For example, well-known Crest factor [2] and Kurtosis method [3] extract statistical measures of the amplitude of time-series sensory data. Other popular algorithms include feature extraction using frequency domain methods, such as power spectral density [4], or time-frequency domain such as wavelet coefficients [5]. More recent methods include wavelet synchrony [6], symbolic dynamic filtering [7, 8] and sparse coding [9, 8]. On the other hand, summary statistics such as count, occurrence rate, and duration have been used as features for event data [10].

These feature extraction algorithms are usually performed as a pre-processing step before a classifier is trained on the extracted features and thus are not guaranteed to be optimally discriminative for the given classification task. Several recent works have shown that better performance can be achieved when a feature extraction algorithm is jointly trained along with a classifier in an end-to-end fashion. For example, in [11, 12], dictionaries are trained jointly with classifiers to extract discriminative sparse codes as feature. Recent successes of deep learning methods [13] on extracting discriminative features from raw data and achieving state-of-the-art performance have boosted the effort for automatic feature discovery in several domains including speech [14], image [15], and text [16] data. In particular, it has been shown that recurrent neural networks [17] and its variants such as LSTMs [18, 19] are capable to capture long-term time-dependency between input features and thus are well suited for feature discovery from time-series data.

While neural networks have also been used for event classification, these efforts have been mostly focused on either univariate signal [20] or uniformly sampled multi-variate time-series data [6]. In this paper, the focus is on event prediction, which is a more challenging task than event classification, where the application data consists of multi-variate and heterogeneous time-series data. Furthermore, the time-series data used here are not uniformly sampled. Followings are the main contributions of the paper:

- We propose a deep learning based architecture for fault prediction on multi-variate heterogeneous time-series data. The proposed algorithm first symbolizes the continuous time-series and combine them with categorical inputs to form words. The vocabulary representation and recurrent network is then trained jointly to learn discriminative features for the given prediction task.
- We propose to use a cost-sensitive formulation to transform the prediction task into a classification task. In this setting, false negatives on near future events are further penalized than false negatives on remote events. The proposed formulation also handles rare event prediction

where distribution of positive and negative samples are highly imbalanced.

- We show that the proposed algorithm achieves state-of-the-art performance on two problems of hard disk failure prediction and heating system fault prediction.

2. SYMBOLIZATION

Symbolization has been widely used as first step for feature extraction providing a more compact representation of time-series data. For example, symbolization is used in [8] to construct probabilistic finite state automata and a measure on the corresponding state transition matrix is used as final feature which is fed into a classifier. However, this kind of designed measures are extracted without explicitly optimizing for the given discriminative task, and thus are only suboptimal. In this work, we directly use the symbolized sequence as the input to a recurrent model to learn discriminative features. Each word of the symbolized sequence is a descriptor of a "pattern" at a time step. Even though the process of generating symbols ignores dependency among variables, we hypothesize that as long as a dataset is large enough and representative patterns occur frequently, a recurrent neural network model should be able to capture the dependencies among the variables. Symbolization for a discrete variable is trivial as the number of symbols is equal to the number of available categories. For continuous variables, this requires partitioning of data. The signal space for each continuous variable, approximated by training set, is partitioned into a finite number of cells that are labeled as symbols. The number of splits for each variable are determined by observing the corresponding marginal data distribution. The locations of splits can be determined either by observation or a clustering algorithm, or uniform partitioning. In this paper, we use Jenks natural breaks algorithm for partitioning [21].

Figures 1 and 2 illustrate the symbolization procedure in a simple example converting a synthetic 2 dimensional time series $\{Z_1, Z_2\}$ into a sequence of words. The histogram of continuous variable Z_1 contains two Gaussians and thus it is partitioned into 2 splits, i.e. for any realization of this variable in the time series, the value is replaced with symbol "a" if it is less than 7, and with "b" otherwise. For discrete variable Z_2, assuming it has 5 categories $Z_2 \in \{C1, C2, C3, C4, C5\}$, we assign symbol "a" to "C1", symbol "b" to "C2", and so on. Following this procedure, each time step is represented by a word which is formed by orderly connecting the symbol realizations of the variables. Thus, each time-series is represented by a sequence of words.

3. PREDICTION ARCHITECTURE

3.1. Formulation of Prediction Problem

In this section, we formulate the event prediction problem. Let $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m\}$ be a sequence of observation sets collected in m time steps, where $\boldsymbol{X}_i = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iN_i}\}$ represents ith input sequence consisting of N_i consecutive measurements. As notation indicates, it is *not* assumed that the number of observations within each time step is constant. Let $\{e_1, e_2, \ldots, e_m\}$ be the corresponding sequence labels for $\boldsymbol{X}$, where $e_i \in \{0, 1\}$ encodes presence of an event within the ith time step, i.e. $e_i = 1$ indicates that a fault event is observed within the period of time that input sequence $\boldsymbol{X}_i$ is collected. We define target labels $y = \{y_1, y_2, \ldots, y_m\}$ where $y_i = 1$ if an event is observed in the next K time-steps, i.e. $\sum_{j=i+1}^{j+K} e_j > 0$, and $y_i = 0$ otherwise. In this formulation, K indicates the prediction horizon and $y_i = 0$ indicates that no event is observed in the next K time-steps, refereed to as monitor window in this paper. The prediction task is then defined as predicting y_i given input $\boldsymbol{X}_t$ and its corresponding past measurements $\{\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-2}, \ldots, \boldsymbol{X}_1\}$. Using the prediction labels y, the event prediction problem on time series data is converted into a classic binary classification problem. Note that although the proposed formulation can in theory utilize all the past measurements for classification, we usually fix a window size of m past measurements to limit computational complexity. For instance, suppose that X_i's are sensory data measurements of a physical system collected at the ith day of its operation and let $K = 7$ and $m = 3$. Then the classification problem for X_i is to predict y_i, i,e, whether an event is going to be observed in the next coming week of the physical system operation, given current and past three days of measurements.

3.2. Temporal Weighting Function

In rave event prediction tasks, the number of positive data samples, i.e. data corresponding to occurrences of a target event, is much fewer than the one of negatives. If not taken care of, this class imbalance problem causes that the decision boundary of a classifier to be dragged toward the data space where negative samples are distributed, artificially increasing the overall accuracy while resulting in low detection rate. This is a classic problem in binary classification and it is a common practice that larger misclassification cost are associated to positive samples to address this issue [22]. However, simply assigning identical larger weights to positive samples for our prediction formulation cannot emphasize the importance of temporal data close to a target event occurrence. We hypothesis that the data collected closer to an event occurrence should be more indicative of the upcoming error than data collected much earlier. Therefore, we design the following weighting function to deal with the temporal importance:

$$w_t = \begin{cases} \sum_{j=1}^{K} (K - j + 1) e_{t+j} & \text{if } y_t = 1 \\ 1 & \text{if } y_t = 0 \end{cases} \quad (1)$$

This weighting function gives relatively smaller weights to data far from event occurrences compared to those which are closer. In addition to temporal importance emphasis, it also deals with overlapping events. For example, suppose that two errors are observed at time samples $t + 1$ and $t + 3$ and

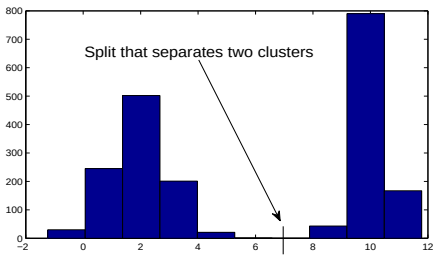

Fig. 1: Partitiong of continous variable Z_1 based on its histogram.

$$\begin{array}{c} \\ t=1 \\ t=2 \\ t=3 \\ t=4 \\ t=5 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cc} Z_1 & Z_2 \end{array} \\ \left[\begin{array}{cc} 2.1 & C5 \\ -0.2 & C1 \\ 0 & C3 \\ 11 & C2 \\ 5 & C1 \\ \vdots & \vdots \end{array}\right] \end{array} \Rightarrow \begin{array}{c} \begin{array}{cc} Z_1 & Z_2 \end{array} \\ \left[\begin{array}{cc} a & e \\ a & a \\ a & c \\ b & b \\ a & a \\ \vdots & \vdots \end{array}\right] \end{array} \Rightarrow \begin{array}{c} \\ ae \\ aa \\ ac \\ bb \\ aa \\ \vdots \end{array}$$

Fig. 2: Time Series symbolization.

$$\left.\begin{array}{ll} ae & \boldsymbol{w}_{ae} \\ aa \Rightarrow & \boldsymbol{w}_{aa} \\ \vdots & \vdots \end{array}\right\} \text{1st Subseq.} \Rightarrow \textbf{SharedLSTM Layer} \Rightarrow \boldsymbol{f}_1$$

$$\left.\begin{array}{ll} ad & \boldsymbol{w}_{ad} \\ bd \Rightarrow & \boldsymbol{w}_{bd} \\ \vdots & \vdots \end{array}\right\} \text{2nd Subseq.} \Rightarrow \textbf{SharedLSTM Layer} \Rightarrow \boldsymbol{f}_2$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$\Bigg\} \Rightarrow \textbf{Max Pooling Layer} \Rightarrow \textbf{Fully Connected Layer} \Rightarrow \hat{\mathbf{y}}$$

Fig. 3: Time-series prediction model. $\boldsymbol{w}_i$ indicates the embedding vector for word i, and $\boldsymbol{f}_i$ indicates the output of LSTM after reading the i-th chunk of embedding vectors of words.

prediction horizon K is set to 5. Then input sample X_t is within the monitor windows of both events and thus its weight is set to higher value of $w_t = (5-1+1)+(5-3+1) = 8$ as misclassification in this day may result in missing to predict two events. By weighting data samples in this way, a classifier is trained to adjust its decision boundary based on the importance information.

The above weight definition deals with temporal importance information for event prediction. We also need to re-adjust weights to address the discussed class imbalance issue. After determining the weight using Eq. 1 for each training sample, we re-normalize all weights such that the total sum of weights of positive samples becomes equal to the total sum of weights of negative samples.

3.3. Loss Function

The weighted cross entropy loss function is used as the optimization criterion to find parameters for our model. For the given input $\boldsymbol{X}_i$ with weight w_i and target label y_i, the loss function is defined as :

$$l(y_i, \hat{y}_i) = w_i(y_i log\hat{y}_i + (1-y_i)log(1-\hat{y}_i)) \qquad (2)$$

where $\hat{y}_i$ is the predicted label.

3.4. Proposed network for event prediction

Figure 3 summarizes the model used for prediction. Similar to use of recurrent models on a natural language processing task, an embedding matrix is learned to map each word in the vocabulary to a vector [23]. The embedding layer generates a sequence of embedding vectors for the given sequence of words. The generated vectors are then fed into an LSTM layer. LSTM [18] has become a popular model for sequential data with variable-length inputs. It is capable of capturing long term dependency in data.

A simple trick is used to increase the use of GPU parallel computing power during the training phase. For a given training time-series with T words, instead of sequentially feeding entire samples to the network, time-series is first divided into M sub-sequences of maximum length $\frac{T}{M}$ where each of these sub-sequences are processed independently and in-parallel. A max-pooling layer is then used on these feature vectors to get the final feature vector which represents the entire time-series. The generated feature is fed into a feed-forward neural network with sigmoid activation function to generate the predictions for the binary classification task. We call the sequence division technique as sequence chopping. Even though training an LSTM with this technique sacrifices temporal dependency longer than $\frac{T}{M}$ time steps, we have observed that by selecting a suitable chopping size, we can achieve competitive results and at the same time significant speed-up of the training procedure.

In our model, the LSTM serves not only as a symbol representation learner, but also a temporal feature extractor. On one hand, it finds the vector representation in continuous vector space for each word, which is a summary descriptor of heterogeneous data at a time step. On the other hand, given a sequence of words, the output of LSTM is a temporal feature

vector that describes the sequence and is used for classification. The main idea of using a recurrent neural network's is its universal approximation ability for open dynamical system [24] to perform a joint learning of dynamics of the system and representations which generate it at the same time.

4. RESULTS

4.1. Backblaze Reliability Dataset

Backblaze data center has released its hard drive datasets containing daily snapshot S.M.A.R.T (Self-Monitoring, Analysis and Reporting Technology) statistics for each operational hard drive from 2013 to June 2016. The data of each hard drive are recorded until it fails. In this paper, the 2015 subset of the data on drive model "ST3000DM001" are used. As far as we know, no other prediction algorithm has been published on this data set and thus we have generated our own training and test split. The data consists of several models of hard drives. There are $59,340$ hard drives out of which 586 of them, less than 1%, had failed. The data of the following 7 columns of S.M.A.R.T raw statistics are used: $5, 183, 184, 187, 188, 193, 197$. These columns corresponds to accumulative count values of different monitored features. We also added absolute difference between count values of consecutive days for each input column resulting in overall 14 columns. The data has missing values which are imputed using linear interpolation. The task is formulated to predict whether there is a failure in the next $K = 3$ days given current and past 4 days data.

The dataset is randomly split into a training set (containing 486 positives) and a test set (containing 100 positives) using hard disk serial number and without loosing the temporal information. Thus training and test set do not have any common hard disk. Then the data are symbolized and converted into sequences of words. The vocabulary is constructed using all words that have been repeated more than once. The training words with frequency of one are all mapped to a single out-of-vocabulary (OOV) word. The resulted vocabulary consists of 353 words, including the OOV. The size of the embedding layer and LSTM layer are found using cross validation on a small validation set and are set to 16 and 8, respectively. For comparison purpose, we also provided the results using logistic regression classification (LR) and LSTM classifiers (without symbolization) on normalized raw data. For LR, the past from five days are concatenated to form the feature vector. All reported models are evaluated using both with and without proposed temporal weighting. The models are trained using ADAM algorithm [25] with default learning rate of 0.001. We reported the balanced accuracy, arithmetic mean of the true positive and true negative rates, the area under curve (AUC) of ROC as performance metrics. The balanced accuracy numbers are generated by picking a threshold on ROC that maximizes true prediction while maintaining a false positive rate of maximum 0.05. Tabel 1 summarizes the performance on test data set. It is seen that the proposed cost-sensitive LSTM with symbolization achieves the best performance.

Table 1: Performance comparison of the studied event prediction models on Backblaze Reliability dataset. The performance is reported for the logistic regression classifier (LR), LSTM without symbolization (LSTM), and the proposed LSTM on symbolized inputs (Sym. + LSTM). For all the methods, performance is reported with (indicated by W.) or without the proposed temporal weighting.

Models	Balanced Accuracy	AUC of ROC
LR	0.799	0.679
LSTM	0.802	**0.835**
Sym. + LSTM	**0.834**	0.812
LR + W.	0.85	0.859
LSTM + W.	0.83	0.864
Sym. + LSTM + W.	**0.852**	**0.874**

Table 2: Performance comparison of our developed model and the model based on hand-designed features on Thermo-technology dataset.

Models	Balance Accuracy	AUC of ROC
LSTM + W	0.711	0.801
LSTM + Sym + W	**0.775**	**0.812**

4.2. Thermo-technology Dataset

We also applied our method on an internal large dataset containing sensor information from thermo-technology heating systems. This dataset contains $132,755$ time-series of 20 variables where each time-series is data collected within one day. Nine of the variables are continuous and the remaining 11 variables are categorical. The task is to predict whether a heating system will have a failure in coming week. The dataset is highly imbalanced, where more than 99% of the data have no fault in the next seven days. After symbolizing the training data, those words that have relative frequency of less than 1% are considered as OOV words. The embedding dimension and size of hidden variables are chosen as 20 and 15, respectively using cross-validation on a validation set. A fully connected layer of dimension 50 after LSTM layer which is followed by the final classification layer. The performance of our model trained on raw data with a similar LSTM architecture which is trained on 119 hand-engineered features are compared in Table 2. The results are reported on test data. As seen, our methods results in competitive performance without the need for the costly process of hand-designing features.

5. CONCLUSIONS

We proposed a recurrent architecture on symbolized input sequence generated from heterogeneous multi-variate time-series data for event prediction. We further proposed a weighting formulation to model temporal importance of the samples and showed that the proposed algorithm achieves the state-of-the-art performance on two real-world datasets.

6. REFERENCES

[1] Ingo Mierswa and Katharina Morik, "Automatic feature extraction for classifying audio data," *Machine learning*, vol. 58, no. 2-3, pp. 127–149, 2005.

[2] N. S. Jayant and Peter Noll, *Digital Coding of Waveforms, Principles and Applications to Speech and Video*, p. 688, Prentice-Hall, 1984.

[3] Jürgen Altmann, "Acoustic and seismic signals of heavy military vehicles for co-operative verification," *Journal of Sound and Vibration*, vol. 273, no. 4, pp. 713–740, 2004.

[4] Dan Li, Kerry D Wong, Yu Hen Hu, and Akbar M Sayeed, "Detection, classification, and tracking of targets," *IEEE signal processing magazine*, vol. 19, no. 2, pp. 17–29, 2002.

[5] Zhiyuan Lu, Xiang Chen, Qiang Li, Xu Zhang, and Ping Zhou, "A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 2, pp. 293–299, 2014.

[6] Piotr Mirowski, Deepak Madhavan, Yann LeCun, and Ruben Kuzniecky, "Classification of patterns of eeg synchronization for seizure prediction," *Clinical neurophysiology*, vol. 120, no. 11, pp. 1927–1940, 2009.

[7] Shalabh Gupta and Asok Ray, "Symbolic dynamic filtering for data-driven pattern recognition," *Pattern recognition: theory and application*, pp. 17–71, 2007.

[8] Soheil Bahrampour, Asok Ray, Soumalya Sarkar, Thyagaraju Damarla, and Nasser M. Nasrabadi, "Performance comparison of feature extraction algorithms for target detection and classification," *Pattern Recognition Letters*, vol. 34, no. 16, pp. 2126 – 2134, 2013.

[9] Ke Huang and Selin Aviyente, "Sparse representation for signal classification," in *Advances in neural information processing systems*, 2006, pp. 609–616.

[10] Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application," *Journal of Machine Learning Research*, vol. 6, no. May, pp. 783–816, 2005.

[11] Julien Mairal, Francis Bach, and Jean Ponce, "Task-driven dictionary learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[12] Soheil Bahrampour, Nasser M Nasrabadi, Asok Ray, and William Kenneth Jenkins, "Multimodal task-driven dictionary learning for image classification," *IEEE Trans. on Image Processing*, vol. 25, no. 1, pp. 24–38, 2016.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, "Deep learning," Book in preparation for MIT Press, 2016.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[17] Jeffrey L Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[18] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[20] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al., "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[21] George F. Jenks, "The data model concept in statistical mapping," *International yearbook of cartography*, vol. 7, no. 1, 1967.

[22] CM Bishop, "Bishop pattern recognition and machine learning," 2001.

[23] Y. Gal, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," *ArXiv*, 2015.

[24] Anton Maximilian Schäfer and Hans Georg Zimmermann, "Recurrent neural networks are universal approximators," in *International Conference on Artificial Neural Networks*. Springer, 2006, pp. 632–640.

[25] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.