Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 381–388

DOI: 10.15439/2015F420
ACSIS, Vol. 5

Robust histogram-based feature engineering of time series data

Eftim Zdravevski*,Petre Lameski†, Riste Mingov¶, Andrea Kulakov‡ and Dejan Gjorgjevikj§
Faculty of Computer Science and Engineering
Ss.Cyril and Methodius University, Skopje, Macedonia
Email: *eftim.zdravevski@finki.ukim.mk, †petre.lameski@finki.ukim.mk,
‡andrea.kulakov@finki.ukim.mk, §dejan.gjorgjevikj@finki.ukim.mk
NI TEKNA - Intelligent Technologies, Negotino, Macedonia
Email: ¶riste.mingov@ni-tekna.com

***Abstract*—Collecting data at regular time nowadays is ubiquitous. The most widely used type of data that is being collected and analyzed is financial data and sensor readings. Various businesses have realized that financial time series analysis is a powerful analytical tool that can lead to competitive advantages. Likewise, sensor networks generate time series and if they are properly analyzed can give a better understanding of the processes that are being monitored. In this paper we propose a novel generic histogram-based method for feature engineering of time series data. The preprocessing phase consists of several steps: deseasonalizing the time series data, modeling the speed of change with first derivatives, and finally calculating histograms. By doing all of those steps the goal is three-fold: achieve invariance to different factors, good modeling of the data and preform significant feature reduction. This method was applied to the AAIA Data Mining Competition 2015, which was concerned with recognition of activities carried out by firefighters by analyzing body sensor network readings. By doing that we were able to score the third place with predictive accuracy of about 83%, which was about 1% worse than the winning solution.**

***Keywords*—feature engineering, feature reduction, time series classification, temporal data mining**

I. Introduction

THE introduction of lightweight and low-cost sensors has increased the potential for real time measurements of different activities. The advancements in microelectronics, wireless communications and other scientific areas has introduced the possibility of placing tiny sensor nodes on specific places of the body in order to monitor the health of patients or human body activities in general [1]. These sensors generate large amounts of data that need to be processed often in real-time. Most of the data, like temperature, accelerometer readings, GPS locations, etc can be presented as a time series data and processed as such. Time series data analysis allows detection of patterns in the data and making assumptions about the current activities or even predict future activities based on the past data. The operations that can be performed based on the time series data are mainly directed towards pattern discovery, clustering, classification and rule discovery [2]. Due to the density of the available data that can be collected by the sensors and the nature of the time series data, there are three main tasks that need to be defined so that the previously mentioned operations can be performed [3]: Dimensionality reduction and Data representation, Distance measurement and Indexing.

Dimensionality reduction and Data representation is one of the most important tasks that need to be performed when analyzing the time series data. This process is taken for granted when we use our sensory organs to obtain the data and then our brain processes it so we can make conclusions. The time series data is usually noisy and too large to be processed by a computer in an acceptable time frame. The human brain processes have learned to ignore the noisy data when generating conclusions. This is why a good representation and dimensionality reduction is crucial before we can continue with the decision making based on the given or obtained data when using a machine learning method. The representation must consider the assumption that time series data is not always aligned properly [4], that it is noisy and that it should comply to the constraints of time and space for its processing. By choosing the right data representation we are able to engineer good features for any given dataset.

The distance measurement is one of the main things that need to be defined in order to make a successful distinction between different time series and be able to correctly classify or identify similar patterns in the data. There are several well known distance measurements that can be used to identify difference between time series. The distance measurement must be invariant to many transformations of the time series data such as amplitude or time shifting, uniform amplification, additive noise, time scaling, etc [3]. For this reason many types of distance measurements have been proposed in the literature and each have their advantages and drawbacks. They can be divided in several groups. There are distance measurements based on the time series shape that use the direct signal properties to give the distance between two series. Then, there are measurements based on the operations needed to make the signals similar with each-other. Also there are measurements based on features extracted from the signal and finally there are measurements based on finding some higher level structure so the series can be compared.

The Indexing problem is related to improving the retrieval speed for a given series when searching trough a database of time series data.

In this paper we propose a histogram-based method for feature engineering for time series data. We use Support Vector Machine to generate the classification model for the data and present the obtained results.

978-83-60810-66-8/$25.00

The paper is organized as follows. In section II we describe the problem that is addressed by this paper. Then, in section III we give overview of the process used to generate the features and the needed transformations of the data. Next, in section IV we address the machine learning method for generating classification models based on the obtained features. Thereupon, in section V we present the obtained results after applying the proposed methods on the competition dataset. Finally, in section VI we discuss the findings of this research and make some conclusions about the applicability of the proposed methods for feature engineering of time series in general.

II. Problem description

The topic of the AAIA'15 Data Mining Competition [5] was Tagging Firefighter Activities at a Fire Scene. In particular, the task is related to the problem of recognizing activities carried out by firefighters based on streams of information from body sensor networks. A fire ground is considered to be one of the most challenging decision taking environment. In dynamically changing situations, such as those occurring at a fire scene, all decisions need to be taken in a very short time [6]. Several initiatives and research projects analyze various aspect of this complex problem [6, 7, 8]. The lack of situational awareness is listed there as the main factor associated with major accidents among firefighters. The research presented in these papers aims to increase the firefighter safety by monitoring their kinematics and psychophysical condition during the course of fire and rescue actions. The following paragraph is extracted from the competition website [5] and describes the task in more detail.

During the course of the ICRA project [9], a so called "smart jacket" have been developed. This device is a wearable set of body sensors that allows to automatically track a firefighter at a fire scene. It also enables real-time screening of firefighter's vital functions and monitoring of ongoing activities at the scene. The later of those two tasks is the main scope of this AAIA'15 Data Mining Competition. The goal was participants to come up with efficient algorithms for labeling activities conducted by firefighters during their training exercises, based on provided data sets from a body sensor network. The data were obtained during training exercises conducted by a group of eight firefighters from The Main School of Fire Service. The sensors were registering firefighter's vital functions (i.e. ECG, heart rate, respiration rate, skin temperature) and movement (i.e. seven sets of accelerometers and gyroscopes placed on torso, hands, arms and legs). Each exercise session was also captured on video and the recordings were synchronized with time series representing the sensor readings. All this data were presented to experts who manually labeled it with activities. The objective in this competition is to devise efficient methods for automatic labeling of short series of the sensory data with basic activities of a firefighter. On the one hand, this task is very challenging due to a fact that different people tend to perform the same activities in different ways. On the other hand, however, automatically generated and accurate activity labels would facilitate monitoring of firefighter's safety and contribute to development of efficient command support systems.

The submitted solutions were evaluated on-line and the preliminary results were published on the competition leaderboard. The preliminary score was computed on a random subset of the test set, fixed for all participants. It corresponded to approximately 10% (about 2000 instances) of the test data. The final evaluation was performed after completion of the competition using the remaining part of the test data (about 18000 instances). Those results was also published on-line. The assessment of solutions was done using the balanced accuracy measure which is defined as an average accuracy within all decision classes. It was computed separately for the labels describing the posture and main activities of firefighters. The final score in the competition was a weighted average of balanced accuracies computed for those two sets of labels. Namely, for a vector of predictions *preds* and a vector of true labels *labels*, the balanced accuracy is defined with eq. (1) and (2). Here BAC_p is the balanced accuracy for labels describing the posture, and BAC_a is the balanced accuracy for labels describing the main activity.

$$ACC_i(preds, labels) = \frac{|j : preds_j = labels_j = i|}{|j : labels_j = i|} \quad (1)$$

$$BAC(preds, labels) = \left(\sum_{i=1}^{l} ACC_i(preds, labels) \right) / l \quad (2)$$

The final score in the competition for a solution s was be computed with eq. (3):

$$score(s) = \frac{BAC_p(s) + 2 \times BAC_a(s)}{3} \quad (3)$$

The instances of the available training and test datasets are comprised mostly of sensor readings as time series and 42 values representing some aggregations of data from sensors monitoring firefighter's vital functions. There are totally 7 sensor locations and 2 sensor types, and each sensor is providing readings for the 3 axes, so the total number of time series is $7 \times 2 \times 3 = 42$. Each of those 42 time series has 400 samples, and one additional series of the timestamps of the readings relative to the start of the series. Each instance is labeled with two labels: one representing the posture of the body, and one representing the main activity of the firefighter. After analyzing the datasets, it is evident that there are 4 classes of the first label and 16 classes of the second label, while there are totally 24 different class combinations of the first and second label. Table I displays the classes for each of the two labels and their distribution in the training set.

Obviously the first challenge in this task is the feature engineering of the time series, so that they can be powerful predictors in relation to the labels, but also to be invariant to several things:

- Invariant to the range of values of the sensor readings because different firefighters can perform the same actions differently.
- Invariant to the alignment of the interval represented by the time series. In the current case, let us consider an action that is being performed longer the duration of the intervals. For example, the firefighter might be running for 20 seconds. From those 20 seconds we

TABLE I. DISTRIBUTION OF CLASSES PER LABEL

Label 1	Label 2	Training Instances	Distribution (%)
crawling	searching	459	2.3
crouching	manipulating	1764	8.8
crouching	no_action	87	0.4
crouching	nozzle_usage	492	2.5
crouching	signal_water_main	46	0.2
moving	ladder_down	465	2.3
moving	ladder_up	476	2.4
moving	manipulating	331	1.7
moving	running	4324	21.6
moving	signal_water_first	41	0.2
moving	stairs_down	644	3.2
moving	stairs_up	1157	5.8
moving	walking	1064	5.3
standing	manipulating	2356	11.8
standing	no_action	491	2.5
standing	nozzle_usage	443	2.2
standing	signal_hose_pullback	98	0.5
standing	signal_water_first	496	2.5
standing	signal_water_main	405	2.0
standing	signal_water_stop	277	1.4
standing	striking	1022	5.1
standing	throwing_hose	234	1.2
stooping	manipulating	1898	9.5
stooping	throwing_hose	930	4.6

could extract thousands of different 1.8s subintervals that represent the action running. Ideally, the feature representation should describe all of those subintervals with almost the same feature vector.

- Invariant of the actions that precede the action that is currently being predicted. To put it differently, the feature descriptor should be invariant to Markov properties of the time series. Firefighter actions have these characteristics so they should be properly modeled. From the actions listed in table I, it seems highly unlikely that the firefighter can perform all pairs of actions (i.e. states) in sequence with the same probability. On the contrary, some state transitions seem very unlikely.

After the features are engineered and the dataset is processed, the next challenge is how to perform feature selection. This is essential because there is significant amount of features that can have negative impact on the used classification algorithms.

Finally, the last challenge is how to build classification models for the different labels given the training dataset. Table I reveals another significant challenge - the number of labels is large and their distribution is highly unbalanced. In the following sections we describe how we have coped with the above challenges and which results were obtained using different techniques. Even though at this point we have mentioned specific numbers like the number of time series or the number of samples in a time series, the approach is generic and in the remaining of the paper we use parameters for them.

III. FEATURE ENGINEERING

In order to address the feature engineering for any problem, first we need to understand the nature of the time series data. Time series data can have different time sampling intervals and different scales of the values, however, most of the data can be useful when building a classification model. In the following subsections we describe which methods were used to address different types of challenges in modeling the time series.

A. Capturing the sensitivity to change with first derivatives

By definition first derivatives are used to capture the speed of changes of some function. When instead of a continuous function we have a discrete sample, like a time series, finding an analytic solution is difficult. Nevertheless, we can estimate them. For a time series with K readings that are collected at times $t[i], 0 \leq i < K$, we can calculate $K-1$ first derivatives. In this case $K = 400$, but we also have the timestamps of the readings which show that usually the interval between readings is 4.5ms. Nevertheless, we decided to use the original time stamp in order to calculate the first derivatives more accurately. Eq. 4 shows how the first derivative fd of time series j and at time $t(i), 0 < i \leq N$ can be estimated.

$$fd_j(i) = \frac{reading_j(i) - reading_j(i-1)}{t(i) - t(i-1)} \tag{4}$$

B. Modeling seasonality

Often in time series there are seasonal components that can consist of periodic, repetitive, and generally regular and predictable patterns in the levels of its values. This is especially evident in business data where things like the holidays, days of week, months, quarters have impact on the values in a business time series (e.g. sales, profit, etc). Seasonal effects can conceal both the true underlying movement in the series, as well as certain non-seasonal characteristics which may be of interest to analysts. There are several main reasons for studying seasonal variation:

- Describing the seasonal effect can provide a better understanding of its impact on a time series.
- Eliminating the seasonal component from time series can aid studying other components such as cyclical and irregular variations.
- Use it to build better models for forecasting and prediction of future seasonal trends.

Keeping in mind that many body movements are also periodic, it occurred to us that maybe we should try to discover and model the seasonality in the current problem. We believe that seasonality in this domain should capture the individual characteristics and style of a particular firefighter and/or the context when some action is performed. By context we mean whether the firefighter is rested (e.g. at the beginning of an exercise), tired (after some time performing various actions), etc.

Before continuing to describing methods for modeling seasonality, some components need to be defined:

- The irregular component (sometimes also known as the residual) is what remains after the seasonal and trend components of a time series have been estimated and removed. It results from short term fluctuations in the series which are neither systematic nor predictable. In a highly irregular series, these fluctuations can dominate movements, which will mask the trend and seasonality.
- The trend is defined as the long term movement in a time series without irregular effects (like calendar related effects in business data), and is a reflection of

the underlying level. In financial data it is the result of influences such as population growth, price inflation and general economic changes.

To model a seasonal component, as described in [10], the following methods are usually used:

1) In an additive time-series model, the seasonal component is estimated as defined with eq. (5). There S stands for the seasonal values, Y is for actual data values of the time-series, T is for trend values, C is for cyclical values and I is for irregular values.

$$S = Y - (T + C + I) \tag{5}$$

2) In a multiplicative time-series model, the seasonal component is expressed in terms of ratio and optionally with percentage as in eq. (6):

$$S = \frac{T \times S \times C \times I}{T \times C \times I} \times 100 = \frac{Y}{T \times C \times I} \times 100 \tag{6}$$

3) The deseasonalized time-series data will have only trend (T) cyclical(C) and irregular (I) components and is expressed with eq. (7) and (8), respectively:

$$Y - S = (T + S + C + I) - S = T + C + I \tag{7}$$

$$\frac{Y}{S} \times 100 = \frac{T \times S \times C \times I}{S} 100 = (T \times C \times I) \times 100 \tag{8}$$

Additionally there is a pseudo-additive model that can be used, but as it was not explored in this research we do not provide more information about it.

The main challenge is discovering the seasonal index that describes the seasonality. Some of the methods for measuring it are: methods of simple averages, ratio to trend, and ratio to moving average. In order to apply them we would have needed experimentation about the length of the sliding window, for which we did not have time, so we decided to do something simpler. Namely, we have opted to use the additive model because we believed that it will correspond to the nature of the data. In particular, although there are difference between the same actions performed by different individuals, they are only linearly shifted values.

Next, we assumed that the seasonal component for each of the N time series ($N = 42$ in the case-study) in one sample can be modeled with the mean value of the K values ($K = 400$ in the case-study). In like manner, we calculated the deseasonalized values (refereed to as deltas in the remaining of the paper) in each training and test instance as defined with eq. (9), where $0 \leq i < 42$ and $0 \leq j < 400$.

$$Delta_i(j) = reading_i(j) - \frac{1}{K} \sum_{j=0}^{K-1} reading_i(j), \tag{9}$$

The motivation behind this approach came from the competition problem. Namely, the observation that some movements should have higher amplitudes than other (e.g. acceleration during running compared to acceleration during walking or standing). With this approach we hope to capture the characteristics of each movement in a more invariant way. Namely, the logic is that if some firefighter performs the same action with higher amplitudes than another, the actual sensor readings for the two series would differ more then the delta values (see eq. (9)).

After we have modeled the characteristics of each movement and posture with the first derivatives and the deltas, as explained in section III the set of available features in the datasets is comprised of:

- 42 values representing the vital functions (specific only to the current problem).
- $N \times K$ which corresponds to the number of series times the number of samples in the series. In this case, this is $42 \times 400 = 16800$ values for the deltas, which represent the original readings from the sensors. Although we are not including the original time series in the model, we calculate some statistics based on them, as described in the following paragraph.
- $N \times (K-1)$ which corresponds to the number of series times the number of first derivatives in the series. In this case this is $42 \times 399 = 16758$ values for the first derivatives.
- $N \times K$ which corresponds to the number of series times the number of samples in the series. In this case this is $42 \times 400 = 16800$ values for the deltas.

To summarize, after modeling the sensitivity to change with first derivatives as described in subsection III-A, and modeling the seasonality described in the current section, there are 3 time series: the original sensor readings, the series of first derivatives and the time series of deltas. For each of them we can calculate the minimum, maximum, mean and standard deviation, a total of 4 metrics, which results in $N \times 3 \times 4$ values (504 in this case). With this the total number of available features adds up to $N \times K + N \times (K-1) + N \times 3 \times 4$. In general, we could enrich the feature set by adding other statistics like first quartile, median, third quartile, interquartile range, skewness, kurtosis, etc, but in this research we have not explored this.

The first obvious problem is the number of features, which is way to high for most machine learning algorithms. More importantly, these features are dependent on the start and alignment of the time series. In section V we discuss how these considerations apply to the competition dataset. In the next subsection III-C we propose a robust method that can address these issues.

C. Histogram-based modeling of time series

In order to address the issues with the features that are available after modeling the sensitivity to change of the time series and modeling the seasonality, it is evident that we need to perform some transformation. Discrete Fourier Transformation (DFT) [11] converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids that has those same sample values. It is usually used to transform the sampled function from its time to the frequency domain. The obtained list of coefficients would be used as a descriptor for the time series, however, we needed

a simpler approach that would be more useful for real time applications.

Nevertheless, DFT lead us to the idea to discretize the values in the time series and then to compute histograms based on it. There are multiple ways in which we can discretize the data, but we decided to apply the simplest one, which is uniform discretization. In order to do that, we needed the minimum and maximum values of series and the number of discretization intervals (referred to as bins in the remaining of the paper). Namely, after calculating the minimum and maximum values of each of the N series of first derivatives and deltas, we only needed to decide how many discretization bins to use. In this case we have $N = 42$ series of first derivatives and $N = 42$ series of deltas. Using more bins means results in more values in the histograms and finer grained granularity, but yields more features. In our tests we have tried using 30, 50 and 100 bins. Somewhat surprisingly, the number of bins did not have a significant impact on the classification results. Our analysis showed that with proper classification model one can achieve good performance using a reasonably small number of bins. We provide more details regarding the influence of the number of discretization bins on the classification performance in section V.

To better explain how the histogram-based approach works let us consider only one time series instance with N values. The following steps should be performed prior applying the transformation:

- Determine the minimum and maximum values of each time series. We need them in order to find on which interval the discretization should be performed.
- Determine the step between discreet values ds based on the number of discretization bins B and the min and max values for the particular time series. This can be calculated with eq. (10)

$$ds = \frac{max - min}{B} \tag{10}$$

After the discretization step, ds, is calculated for a particular time series, all training and test instances can then be discretized. For each original value V we can calculate the discrete value DV with eq. (11).

$$DV = -|min| + round(\frac{|min| + V}{ds}, 0) \times ds \tag{11}$$

The next step is to calculate the histogram for the time series. If we use B bins, then the histogram for a particular training instance will have B values. By doing this, from a time series with N values we obtain a histogram of B values, where $N > B$. The B values represent the transformed features which are robust, but are also a significantly reduced representation of the original time series.

To illustrate the transformation let us consider the exemplary time series shown at Fig. 1. Let us assume that we want to discretize the values of this time series (red line) to $B = 7$ bins on the interval $[-3, 3]$. After we calculate the discretization step according to eq. (10) we determine the discrete values according to (11). Using the discrete values (blue line) we can calculate the histogram displayed on Fig. 2. Consequently

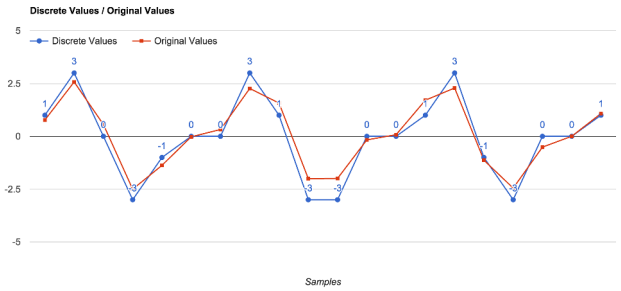

Fig. 1. Original and discrete values of an exemplary time series

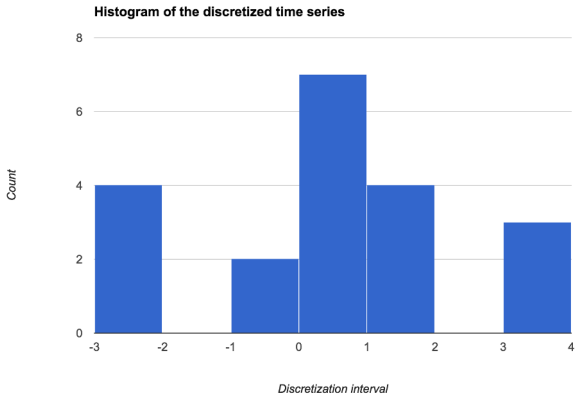

Fig. 2. Histogram of the discrete values of an exemplary time series

starting from a time series with 20 values, a histogram of 7 values is obtained.

After applying the histogram transformation of the time series described above, when using B discretization bins, the following dataset is obtained:

- 42 values representing the vital functions (specific to this case only).
- $2 \times N \times B$ for the first derivatives and the deltas. In this case $N = 42$, so $Hist = 2 \times 1260$ for $B = 30$, $Hist = 2 \times 2100$ for $B = 50$, and $Hist = 2 \times 4200$ for $B = 100$.
- $N \times 3 \times 4$ for the 4 aggregated values of the 3 types of series, which in this case is 504 additional features.

To summarize, after performing the histogram transformation, the total number of features in the dataset of the current problem was 3066, 4746 and 8946 when using 30, 50 and 100 bins, respectively.

D. Feature selection

After inspection of the transformed dataset it was evident that for some features almost all training instances had the same value. In order to address this, we calculated the variance of each feature in the training set. Using the variance for discarding non-informative features is a simple baseline approach to feature selection. It removes all features whose variance does not meet some threshold. If a feature has the same value in all training samples, then its variance is 0.

When discarding the features with different thresholds for the variance, we have obtained the results in the following table. It can be noted that using a threshold greater than 0.05 does not significantly reduce the number of features. On the contrary, the cross validation results showed decline in performance. Table II shows the number of retained features per discretization bins and variance threshold on the competition datasets.

TABLE II. RETAINED FEATURES PER DISCRETIZATION BINS AND VARIANCE THRESHOLD

Discretization bins	Features	Variance threshold	Retained features
30	3066	0.05	1780
50	4746	0.001	3196
50	4746	0.01	2601
50	4746	0.03	2272
50	4746	0.05	2137
50	4746	0.1	1927
100	8946	0.1	5569

In the case when we used 100 bins we applied 0.1 as a variance threshold aiming to discard more features, but still over 5000 features remained. That is why we applied PCA (Principal Component Analysis) [12] and limited the selected features to 2000 explicitly. We have chosen the value of 2000 because it was close to the number of retained features obtained by the variance filter when using smaller number of bins.

When doing feature selection with PCA the results were similar in terms of the actual selected features, but for that the computation time was greater. We acknowledge that with more sophisticated feature selection methods like wrappers we might further lower the dimensionality and improve the classification performance. Nevertheless, the nature of the features is such that the features are different from each other because they represent different things. With this in mind, the expectation is that there are no redundant features in the dataset. Additionally, some machine learning algorithms like Support Vector Machines (SVMs) are able to cope with small number of redundant features without significant degrading of performance.

E. Data normalization

Prior building any models we normalized the data using mean and standard deviation calculated from the training set. This process notably helps the next step while building prediction models. The process is recommended as part of the data preprocessing when using Support Vector Machines (SVM) [13] to make the classification model and is known to decrease the classification error [14] .

IV. MODEL BUILDING

Prior building any models for the competition challenge one needs to define how the two different labels will be handled. One idea is to build a separate model for each label and then to merge the predictions, but also making sure that there are not any contradictions, i.e. combinations that are not possible. Another idea is to perform hierarchical multi-label classification, so first we would classify the training data based on the first label, and then to perform classification based on the second label. When building the second-level models one can perform one-vs-all classification based on the possible outcomes of the second label assuming the first label was correctly predicted. This approach helps by not needing to explore all possible classes of the second label. Nevertheless, given that the combinations of the second label (16) were close to the number of total combinations (24) and due to the limited time for the competition we did not explore these approaches significantly. Instead we used a one-level classification. The label of the instances is the combination of the two original labels, meaning that our classification model tries to predict both labels at the same time.

The set of features after the feature selection (i.e. preprocessed descriptor) is used to generate a classification model using SVM. SVM generates a classification model by finding the optimal support vectors that divide the feature space with the highest margin between the vector and the nearest points, so that all instances on one side of the support vectors belong to one class, and all other instances belong to other classes. For the purpose of our task we are using SVM with Radial Based Function (Gaussian function) that uses two parameters in the optimization process, C and gamma. Since most of the classification problems are not linearly separable, we are using the Gaussian SVM so that the separation is done in a higher dimension vector space generated with the nonlinear Gaussian kernel. To obtain the best parameters on the given data, we are using grid search as suggested in [15]. During the grid search we use exponentially increasing parameters. We have searched both C and gamma parameters in the interval 10^{-5} to 10^{5} evaluating $11 \times 11 = 121$ combinations. Then in the intervals that were giving best results we conducted finer grained grid search with smaller steps for gamma and C. Depending on the different training datasets (varied by the number of discretization bins and the variance threshold for feature selection) we have obtained different optimal values. For C the optimal values ranged from 1 to 1000 and for gamma they were usually from 10^{-5} to 10^{-4}. Also important to realize is the unbalanced distribution on the classes. In order to address this problem, we used the class distribution to automatically adjust the weights inversely proportional to class frequencies. The estimated weights are then multiplied by the C parameter and those weighted C parameters are used when building the one-vs-all classification models for multi-class classification. This significantly improved the performance of the classifiers when using cross validation, but more importantly on the leaderboard set.

V. EVALUATION OF THE PROPOSED METHODS ON THE COMPETITION DATASET

For the current challenge, we began our analysis with plotting and visual inspection of some of the sensor readings for random training samples. Before inspecting the data the impression was that the interval of 4.5 milliseconds between different readings could be too short for the sensors to output different values. On the contrary, the analysis showed that there are evident differences between consecutive readings, meaning that indeed all values in the series are potentially useful for building prediction models.

After we have modeled the characteristics of each movement and posture with the first derivatives and the deltas, as explained in subsections III-A and III-B the set of available features in the datasets is comprised of:

TABLE III. SCORE ON THE LEARDERBOARD DATASET DEPENDING ON VARIOUS DISCRETIZATION BINS AND SVM CONFIGURATIONS

Id	Leaderbord Score	Bins	Variance	Retained features	SVM with RBF kernel
1	0.8502	N/A	N/A	N/A	Combination of configurations in rows 2 and 3 (below)
2	0.8381	50	0.05	2137	C=10 gamma=0.00002 weight=auto
3	0.8380	30	0.05	1780	C=10 gamma=0.000035 weight=auto
4	0.8379	50	0.05	2137	C=10 gamma=0.000015 weight=auto
5	0.8376	30	0.05	1780	C=10 gamma=0.00004 weight=auto
6	0.8371	50	0.05	2137	C=10 gamma=0.000025 weight=auto
7	0.8371	50	0.05	2137	C=20 gamma=0.00001 weight=auto
8	0.8349	50	0.05	2137	C=10 gamma=0.00004 weight=auto
9	0.8342	30	0.05	1780	C=10 gamma=0.00002 weight=auto
10	0.8341	50	0.05	2137	C=20 gamma=0.00002 weight=auto
11	0.8337	50	0.1	1927	C=10 gamma=0.00002 weight=auto
12	0.8313	50	0.05	2137	C=10 gamma=0.00001 weight=auto
13	0.8308	30	0.05	1780	C=10 gamma=0.00007 weight=auto
14	0.8302	30	0.05	1780	C=10 gamma=0.00008 weight=auto
15	0.8288	100	0.1	2000 (reduced from 5569 with PCA)	C=10 gamma=0.0001 weight=auto
16	0.8279	100	0.1	2000 (reduced from 5569 with PCA)	C=10 gamma=0.00004 weight=auto
17	0.8257	30	0.05	1780	C=10 gamma=0.0001 weight=auto
18	0.8251	50	0.1	1927	C=10 gamma=0.00004 weight=auto
19	0.8225	100	0.1	2000 (reduced from 5569 with PCA)	C=10 gamma=0.00002 weight=auto
20	0.8198	100	0.1	2000 (reduced from 5569 with PCA)	C=10 gamma=0.00001 weight=auto
21	0.8191	30	0.05	1780	C=10 gamma=0.0001 weight=auto
22	0.8167	50	0.05	2137	C=10 gamma=0.0001 weight=auto
23	0.8110	100	0.1	2000 (reduced from 5569 with PCA)	C=1.0 gamma=0.0001 weight=auto
24	0.8077	50	0.05	2137	C=1000 gamma=0.0001 weight=auto
25	0.8062	50	0.05	2137	C=10 gamma=0.0001

- 42 values representing the vital functions (specific to this case only).
- $2 \times N \times B$ for the first derivatives and the deltas. In this case $N = 42$, so $Hist = 2 \times 1260$ for $B = 30$, $Hist = 2 \times 2100$ for $B = 50$, and $Hist = 2 \times 4200$ for $B = 100$.
- $N \times 3 \times 4$ for the 4 aggregated values of the 3 series, which in this case is 504 additional features.

The first obvious problem is the number of features, which is way to high for most machine learning algorithms. More importantly, these features are dependent on the start of the time series. For instance, let us assume that we have an action (e.g. running) that is being performed with a total duration of 10000 milliseconds (10 seconds). If we extract two samples (training or test instances) from it, sample A spreading from $t_1 = 0ms$ to $t_2 = 1800ms$, and sample B spreading from $t_3 = 9ms$ to $t_4 = 1809ms$. The logic is that those two samples are nearly identical and should be treated accordingly in a good feature space. However, in the current representation they will be different because they are shifted by 2 periods of 4.5ms. Obviously this needs to be addressed, and this is performed by the histogram-based method described in subsection III-C. Then with the method described in subsection III-D we have reduced the number of features as shown with table II.

We have obtained many similar results based on the different alternatives we tried (different C and gamma parameters, number of discretization bins, etc.) on both the leaderboard dataset and with 5-fold cross validation with the training set. Table III shows some of the more significant configurations ordered by the score on the leaderboard dataset in descending order. Before explaining the best score, which is shown in the first line in this table, we first want to discuss the scores of the individual classifier configurations shown in all other rows.

The first obvious thing to notice is that when applying weights (denoted as "weight=auto" in table III to the C parameters improves the performance of the classifiers. Namely row 25 does not have weights applied to the C parameter, meaning the value of C=10 is used for all classes in the one-vs-all multi-class SVM. When applying weights proportional to the class frequencies in the training dataset the performance is improved as shown in row 22. We have noticed the same pattern in other configurations (i.e. different bin size, different C and gamma parameters) which are not shown in this table.

Next, we have noticed that when varying the values of the C and gamma parameters the classification score also varies when using the same feature set. This was expected and confirms the need for grid search, as described in section IV, to find the optimal values for those parameters.

Finally, the most important realization is that the greater number of discretization bins does not necessarily mean the score will be better. Most compelling evidence to this statement are the two best configurations shown in rows 2 and 3. Row 2 uses 50 bins, while row 3 uses 30 bins, but the difference between their score is 0.0001 in favour of the 50 bins feature set. On the other hand, the 30 bins feature set has 357 features less, which in turn leads to less training and test time. Another evidence that supports this claim is the case when we use 100 bins (rows 15, 16, 19, 20, 23). In those cases we obtain a much greater number of features than when using 50 or 30 bins, so we have to perform additional feature selection in order to reduce the training time. If we have retained more features and performed more detailed grid search for those feature sets, the results might have been better. Our initial tests showed that this in fact leads towards over-fitting and generating too many support vectors. For this reason this did not seem like worth doing, especially because we have managed to find better solutions with significantly less features.

The score difference between different configurations may seem negligible, but when looking at the actual predictions made by the 2 best models (rows 2 and 3) we came to an important realization. Namely, it was evident that both cross-validation and test predictions made by the 2 models are significantly different, yet the score of the 2 models was similar. This gave us an idea to train a second-level (ensemble) model. The features for this model were the predictions and probabilities

of the 2 individual classifiers made with cross validation on the training dataset and the class was combination of the two labels. By applying this we further improved the score on the leaderboard dataset, and our final solution had a score of 0.8502. It was the third-placed score on the leaderboard and was only 0.001 behind the second-placed solution, 0.008 behind the first place, and 0.025 better then the fourth-placed solution. After the final results were published our score was still third - 0.8261, while the first was 0.8391, the runner-up was 0.82985 and the fourth placed score was 0.80408. We acknowledge that combining more individual models should be explored with a more generic approach.

VI. Conclusion and future work

Based on the obtained results, we can conclude that the proposed descriptor gives a good model for the movements, while providing some invariance to deviation of the input values and also performing significant feature reduction. During the analysis of the predictions made by the different classification models we observed that most of the miss-classification errors were made when the classification model was trying to distinguish between very similar tasks for the given dataset, such as *moving + stairs-up* vs *moving + stairs-down* and *moving + manipulating* vs *crouching + manipulating*. These errors were inevitable since our approach for generating the descriptors is amplitude based and some specific movements have very similar amplitude characteristics. We believe that there is room for further improvement if we add additional descriptors to the feature space like: other statistics to describe the time series, amplitudes of the acceleration across all 3 axes, discovering and modeling the trend in the series, trying logarithmic transformations, adding second order derivatives, etc. Performing a better feature selection may also improve the performance. Finally, a better modeling that takes into account the hierarchical nature of the classification problems, as discussed previously, can further improve the results.

The main contribution of this paper is the proposed method for invariant modeling the time series by using first derivatives and deltas. Moreover, the novelty of our approach is in the proposed histogram-based method for feature reduction of the time series. Even though it was developed during the competition, it is applicable to time series regardless from their domain. In our future research we plan to affirm this method by analyzing time series from various domains and comparing it to other methods for modeling time series.

Acknowledgment

This work was partially financed by the Faculty of Computer Science and Engineering at the Ss.Cyril and Methodius University, Skopje, Macedonia and by NI TEKNA - Intelligent technologies, Negotino, Macedonia.

References

[1] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," pp. 1658–1686, Third 2014.

[2] T. chung Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164 – 181, 2011. doi: http://dx.doi.org/10.1016/j.engappai.2010.09.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197610001727

[3] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 12:1–12:34, Dec. 2012. doi: 10.1145/2379776.2379788. [Online]. Available: http://doi.acm.org/10.1145/2379776.2379788

[4] B. Hu, Y. Chen, and E. Keogh, "Classification of streaming time series under more realistic assumptions," *Data Mining and Knowledge Discovery*, pp. 1–35, 2015. doi: 10.1007/s10618-015-0415-0. [Online]. Available: http://dx.doi.org/10.1007/s10618-015-0415-0

[5] M. Meina, A. Janusz, K. Rykaczewski, D. Slezak, B. Celmer, and A. Krasuski, "Tagging firefighter activities at the emergency scene: Summary of AAIA'15 data mining competition at Knowledge Pit," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2015, in print September 2015.

[6] A. Krasuski, "A framework for dynamic analytical risk management at the emergency scene. from tribal to top down in the risk management maturity model," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. IEEE, 2014, pp. 323–330.

[7] A. Krasuski, A. Jankowski, A. Skowron, and D. Slezak, "From sensory data to decision making: A perspective on supporting a fire commander," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. IEEE, 2013, pp. 229–236.

[8] M. Meina, B. Celmer, and K. Rykaczewski, "Towards robust framework for on-line human activity reporting using accelerometer readings," in *Active Media Technology*. Springer, 2014, pp. 347–358.

[9] "ICRA' project," http://www.icra-project.org/, accessed: 2015-06-05.

[10] A. G. Barnett and A. J. Dobson, *Analysing Seasonal Health Data*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010. ISBN 9783642107481 3642107486

[11] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*. Springer-Verlag, 1993, pp. 69–84.

[12] I. Jolliffe, "Principal Component Analysis," in *Wiley StatsRef: Statistics Reference Online*, N. Balakrishnan, T. Colton, B. Everitt, W. Piegorsch, F. Ruggeri, and J. L. Teugels, Eds. Chichester, UK: John Wiley & Sons, Ltd, Sep. 2014. ISBN 9781118445112. [Online]. Available: http://doi.wiley.com/10.1002/9781118445112.stat06472

[13] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. doi: 10.1145/1961189.1961199. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[14] D. M. Tax and R. P. Duin, "Feature scaling in support vector data descriptions," Technical report, Technical report, American Association for Artificial Intelligence, Tech. Rep., 2000.

[15] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification."