Optimizing Time Series Discretization for Knowledge Discovery

Fabian Mörchen
Data Bionics Research Group
Philipps-University Marburg
35032 Marburg, Germany
fabian@informatik.uni-marburg.de

Alfred Ultsch
Data Bionics Research Group
Philipps-University Marburg
35032 Marburg, Germany
ultsch@informatik.uni-marburg.de

ABSTRACT

Knowledge Discovery in time series usually requires symbolic time series. Many discretization methods that convert numeric time series to symbolic time series ignore the temporal order of values. This often leads to symbols that do not correspond to states of the process generating the time series and cannot be interpreted meaningfully. We propose a new method for meaningful unsupervised discretization of numeric time series called Persist. The algorithm is based on the Kullback-Leibler divergence between the marginal and the self-transition probability distributions of the discretization symbols. Its performance is evaluated on both artificial and real life data in comparison to the most common discretization methods. Persist achieves significantly higher accuracy than existing static methods and is robust against noise. It also outperforms Hidden Markov Models for all but very simple cases.

Categories and Subject Descriptors: I.5 [Computing Methodologies]: Pattern Recognition

General Terms: Algorithms

Keywords: time series, discretization, persistence

1. INTRODUCTION

Many time series data mining algorithms work on symbolic time series. For numeric time series they usually perform unsupervised discretization as a preprocessing step, e.g. [6, 12, 5]. The choice of the method and the accompanying parameters are rarely justified, however. For the discovery of knowledge that is interpretable and useful to the expert, it is of great importance that the resulting interval boundaries are meaningful within the domain. If the time series is produced by an underlying process, that consists of recurring persisting states, it is desirable to obtain a discretization where the intervals in the value dimension describe these states.

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.

The most commonly used discretization methods are equal width and equal frequency histograms. The former method is vulnerable to outliers in the data. A single extreme value will affect all bin boundaries and their widths severely. The symbols obtained from two datasets cannot be meaningfully compared when outliers are present. This way any attempt of knowledge discovery, usually involving interpretation of the symbols as states like *high* or *low* will give spurious and misleading results. Both histogram methods potentially place cuts in high density regions of the observed marginal probability distribution of values. This is another disadvantage, if discretization is performed not merely for quantization and speedup of processing, but rather for gaining insight into the process generating the data. In general, it is hard to justify why a value slightly left of the cut should be different from a value slightly to the right. The higher the density at a cut point, the more of such problematic symbols we have. The assignment of values to a certain state is somewhat arbitrary near the decision boundaries. The same disadvantages also apply to other methods, e.g. setting cuts based on location and dispersion measures.

While static data offers no information other than the actual values themselves, time series contain valuable temporal structure that is not used by the methods described above. We propose a new method for meaningful unsupervised discretization of univariate time series by taking the temporal order of values into account. The discretization is performed optimizing the persistence of the resulting states.

In Section 2 we discuss related methods. The definition of the quality score in Section 3 is followed by the description of the new discretization algorithm in Section 4. The effectiveness of our approach is demonstrated with artificial and real life data in Section 5. Results, limitations, and extensions are discussed in Section 6. Section 7 summarizes the paper.

2. RELATED WORK AND MOTIVATION

A review of supervised discretization methods is given in [14], both supervised and unsupervised methods are covered in [3] and more recently in [17]. The only unsupervised methods for discretization listed in the above reviews are equal width and equal frequency histograms. With unsupervised discretization no class labels are available, thus there can be no optimization w.r.t. classification accuracy. But for time series data in particular there is rarely some sort of labeling for the time points available. Far more common is the classification of whole time series (e.g. [8, 22]) but

a single label is of little help for the discretization of the numerical values.

Recently, the Symbolic Approximation (SAX) has been proposed in [16]. Based on the Piecewise Aggregate Aggregation (PAA) [13] and the assumption of normality of the resulting aggregated values, a method similar to equal frequency histograms is obtained. SAX is the first symbolic representation of time series with an approximate distance function that lower bounds the Euclidean distance. While SAX is one of the first discretization methods designed especially for time series data, the temporal aspect of the data is only taken into account by the preprocessing step of performing the PAA. The window size and the alphabet size create a tradeoff between efficiency and approximation accuracy. The choice of these parameters has been analyzed in the context of temporal rule mining in [7]. Different methods for model selection are tried and judged by the support and confidence of the resulting rules. Rules are, however, typically created to gain a deeper understanding of the data and the patterns therein. Arguably, rules with high support and confidence are less likely to be spurious results. But they will not be useful if the interval boundaries of the discretization are not meaningful to the domain expert.

Several discretization methods for time series are discussed by Daw et al. [2]. The discretization is said to be often motivated by the desire to speed up processing and to remove noise. Many real life time series are smooth [11]. Using each time point or a small window for discretization will usually produce consecutive stretches of the same symbol. Daw et al. state that *"from the standpoint of observing meaningful patterns, high frequencies of symbol repetition are not very useful and usually indicate over-sampling of the original data"*. But interesting temporal phenomena do not necessarily occur at the same time scale. Trying to avoid this so called over-sampling would mean to enlarge the window size, possibly destroying short temporal phenomena in some places. We think that with smooth time series it will be better to keep this high level of temporal resolution and search for persisting states. This results in labeled interval sequences representing the concept of duration. More complex temporal patterns covering the concepts of coincidence and order can be searched with the Time Series Knowledge Mining (TSKM) framework [19, 20].

The related task of time series segmentation (e.g. [10]) is beyond the scope of this paper. Segmentation does not lead to recurring state labels per se. Instead of dividing the value dimension in intervals, the time dimension is segmented to produce line or curve segments homogeneous w.r.t. some quality measure. Feature extraction on the segments can lead to recurring labels like *increasing*.

3. PERSISTENCE SCORE

We propose a new quality score for meaningful unsupervised discretization of time series by taking the temporal information into account and searching for persistence. If discretization is not done purely for the sake of compression and noise filtering [2] the analyst is commonly interested in states of the underlying process generating the time series. We assume that the time series contain enduring states and that these states are of interest. Any long term trend that could cause such states not to be aligned on the value axis should be removed, e.g. by substracting moving averages. Time series with very fast changing behaviour are not well suited for the discovery of persisting states, but they can be after an appropriate feature extraction.

We argue, that one discretization is better than another if the resulting states show more persisting behavior. We expect many knowledge discovery approaches to profit from more meaningful symbols incorporating the temporal structure of the time series, e.g. rule discovery in univariate [6] and multivariate [5, 20] time series, or anomaly detection [12]. The quality measure used for persistence is based on the Kullback-Leibler divergence of the marginal and self-transition probability distributions of the symbols. Let $S = \{S_1, ..., S_k\}$ be the set of possible symbols and $s = \{s_i | s_i \in S \ i = 1..n\}$ be a symbolic time series of length n. Let $P(S_j)$ be the marginal probability of the symbol S_j. The $k \times k$ matrix of transition probabilities is given by $A(j,m) = P(s_i = S_j | s_{i-1} = S_m)$. The self-transition probabilities are the values on the main diagonal of A.

If there is no temporal structure in the time series, the symbols can be interpreted as independent observations of a random variable according to the marginal distribution of symbols. The probability of observing a each symbol is independent from the previous symbol, i.e. $P(s_i = S_j | s_{i-1}) = P(S_j)$. The transition probabilities are $A(j,m) = P(S_j)$. The most simple temporal structure is a first order Markov model (e.g. [21]). Each state depends only on the previous state, i.e. $P(s_i = S_j | s_{i-1}, ...s_{i-m}) = P(S_j | s_{i-1})$. Persistence can be measured by comparing these two models. If there is no temporal structure, the transition probabilities of the Markov model should be close to the marginal probabilities. If the states show persisting behavior, however, the self-transition probabilities will be higher than the marginal probabilities. If a process is less likely to stay in a certain state, the particular transition probability will be lower than the corresponding marginal value. Since it is the self-transitions in particular we are interested in, we will not compare the complete distributions of k symbols but only the the distributions of self vs. non-self per state. Note, that we are not assuming the symbolic time series to be produced by a first order Markov model. We merely use the self-transition probabilities as an indicator for persistence. The higher this probability, the longer the intervals of this particular state tend to be.

A well known measure for comparing two probability distributions is the Kullback-Leibler divergence [15]. For two discrete probability distributions $P = \{p_1, ..., p_k\}$ and $Q = \{q_1, ..., q_k\}$ of k symbols it is defined in Equation 1.

$$KL(P,Q) = \sum_{i=1}^{k} p_i log(\frac{p_i}{q_i}) \tag{1}$$

A symmetric version is obtained by taking the mean of both directions (Equation 2).

$$SKL(P,Q) = \frac{1}{2}\left(KL(P,Q) + KL(Q,P)\right) \tag{2}$$

Figure 1 shows a plot of the symmetric Kullback-Leibler divergence between two binary random variables for probability distributions from $\{0.01, 0.99\}$ to $\{0.99, 0.01\}$ for each. The value is 0 on the diagonal of the pq plane, where both distributions are equal. The values rise continuously toward the extreme differences in the left and right corners. For binary random variables we define the shortcut notation in

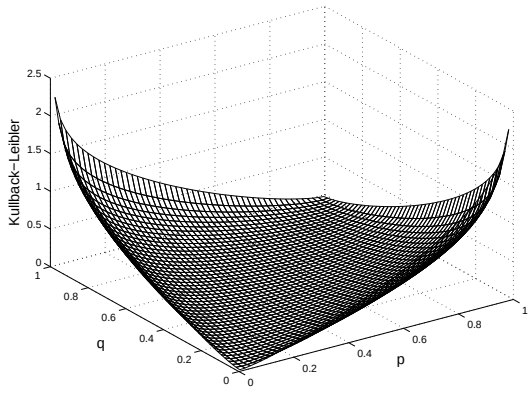


Figure 1: Symmetric Kullback-Leibler divergence of $P = \{p, 1-p\}$ and $Q = \{q, 1-q\}$

Equation 3.

$$SKL(p,q) := SKL(\{p,1-p\},\{q,1-q\})\ \forall p,q \in]0,1] \quad (3)$$

The *persistence* score of state j is defined in Equation 4 as the product of the symmetric Kullback-Leibler divergence of the transition and marginal probability distribution for self vs. non-self with an indicator variable. The indicator determines the sign of the score. States with self-transition probabilities higher than the marginal will obtain positive values and states with low self-transition probabilities inhibit negative values. The score is zero if and only if the probability distributions are equal.

$$Persistence(S_j) = sgn(A(j,j) - P(S_j))SKL(A(j,j), P(S_j)) \quad (4)$$

A summary score for all states is defined in Equation 5 as the mean of the values per state. This captures the notion of mean persistence, i.e. all or most states need to have high persistence for achieving high persistence scores. The higher the persistence score, the more likely it is to observe at any point in time the same state as for the previous time point and this implies longer persisting state intervals.

$$Persistence(S) = \frac{1}{k}\sum_{m=1}^{k} Persistence(S_m) \quad (5)$$

The calculation of the persistence scores is straight forward. Maximum likelihood estimates of all involved probabilities can easily be obtained by counting the number of symbols for each state for the $P(S_j)$ and the numbers of each possible state pair for A. A problem arises if there are self-transitions that are never observed. The Kullback-Leibler divergence is not defined if any involved probability is zero. This is a very rare case, however. The marginal probabilities are never zero, otherwise a state wouldn't exist. The non-self transition probability can only be zero if a state only occurs in a contiguous segment at the end of the time series. This can happen for at most one state. The self-transition can only be zero if a state never follows itself, unlikely for smooth time series. To avoid an invalid score with zero probabilities we set the probability to a small value (e.g. n^{-1}) that is subtracted from the complementary event. As long as the value is smaller than the smallest observed probability, the scores can still be used for comparison.

4. THE PERSIST ALGORITHM

The persistence score is used to guide the selection of bins in the *Persist* algorithm described in this chapter. The first step is to obtain a set of candidate bin boundaries from the data. We propose to use equal frequency binning with a large number of bins to obtain a coarse sampling of candidate cuts in sparse regions and a fine sampling in dense regions. As a default value for the number of bins we use 100 to get the percentiles of the data. If a higher accuracy is required more bins can be used. In each iteration of the algorithm all available candidate cuts are individually added to the current set of cuts and the persistence score is calculated. The cut achieving the highest persistence is chosen. This is repeated until the desired number of bins is obtained.

Let $X = \{x_i | x_i \in \mathbb{R}\ i = 1..n\}$ be the numerical values of a time series of length n and $C = \{c_j | c_j \in \mathbb{R}\ j = 1..m\}$ be the set of candidate cuts. Let D be a function performing the actual discretization given X and a set of bin boundaries B. The Persist algorithm for discovering k states is given in Figure 2. The time complexity is $O(n)$.

$B := \emptyset$ // bin boundaries
for $i = 1...k-1$
  $P := \emptyset$ // stores persistence scores
  foreach $c \in C$ // all candidates
    $P := P \cup \{Persistence(D(X, B \cup \{c\}))\}$
  end
  $b := max_i(P)$ $B := B \cup \{c_b\}$ // add best cut
end

Figure 2: Persist algorithm for unsupervised time series discretization

The symbols are not required to be equally frequent, thus even rare states can achieve high scores. In many cases it will be desirable to avoid the creation of very rare states, i.e. bin boundaries very close to cuts previously chosen. We excluded states that cover less than 5% of the time points. This is not shown in Figure 2 for brevity of presentation.

5. EXPERIMENTS

The evaluation of unsupervised algorithms is difficult due to the lack of the ground truth. We evaluated the performance of the Persist algorithm by extensive experiments using artificial data with known states and real life data where the true states were rather obvious. We compared the novel algorithm with the following eight methods.

EQF: For equal frequency histograms k bins containing approximately the same number of data points were created.

SAX: Inspired by the SAX method k bins were chosen such that values from a normal distribution with the same mean and standard deviation as the data would be equally likely in each bin. Note, that we are actually using a special case of SAX. A window size of one for the PAA is used, effectively skipping this step. We are also not using numerosity reduction or sliding windows to be comparable with the other methods. The results should therefore not be held against the full SAX method designed to preserve time series distances.

EQW: For equal width histograms the range of the data was divided into k segments of equal length.

M±S: The mean (μ) and standard deviation (σ) were used

for creating bins. For an even number of bins the boundaries were $\{\mu + i\sigma | i = -\frac{k-2}{2}, ..., \frac{k-2}{2}\}$, for odd values of k we used $\{\mu + \left(i - sgn(i)\frac{1}{2}\right)\sigma | i = -\frac{k-1}{2}, ..., \frac{k-1}{2}\}$ $(k > 1)$.

M±A: Same as M±S using median and adjusted median absolute deviation (AMAD) instead of mean and standard deviation, respectively.

KM: The k-Means algorithm[1] with uniform initialization and Manhattan distance was used to obtain k clusters. The bin boundaries were calculated as the midpoints of neighboring cluster centers.

GMM: A Gaussian mixture model [1] was initialized with the cluster centers from k-Means and the Expectation Maximization (EM) algorithm was used to estimate the means, variances, and mixing weights. The bin boundaries were obtained by maximum likelihood decision.

HMM: A Hidden Markov Model [21] with k states and a single Gaussian output distribution per state was initialized with the result of the GMM estimation. The training of the model and the calculation of the Viterbi state sequence were performed using the HMM Toolbox [2]. HMM is the only competing method using the temporal information of the time series. It is not quite comparable to the other methods, however, because it does not return a set of bins. The state sequence is directly created.

5.1 Artificial data

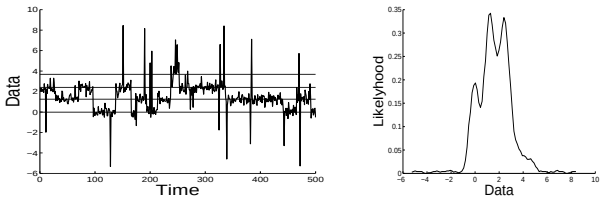
We generated artificial data using a specified number of states and Gaussian distributions per state. The standard deviation for each state was chosen with uniform random numbers between 0.1 and 1. The mean value of the first state was set to zero. The distance of the means from one state to the next was specified in terms of the number of standard deviations of both distributions. This distance is chosen with uniform random numbers between 1 and 2. E.g. if the standard deviation of the first and second states are 0.5 and 0.7, respectively, and the distance is 1.5, the second mean will be $0 + 1.5(0.5 + 0.7) = 1.8$. Each new state was chosen randomly with all states being equally likely. The duration of each state was chosen with uniform random numbers from the interval $[0.005n, 0.05n]$. We generated 1000 time series of length 1000 for $k = 2, ..., 7$ states. For each time series 10 additional noisy versions were created by adding 1% to 10% outliers uniformly drawn from the interval determined by the mean ± the range of the original time series.

An example for 4 states and 5% outliers is shown in Figure 3(a). The horizontal lines indicate the true means of the 4 states. The large spikes are caused by the outliers. Figure 3(b) shows the Pareto Density Estimation (PDE) [23] of the marginal empirical probability distribution. Three states are visible as peaks with significant overlap. The existence of the 4th state on the right is not at all obvious from the marginal probability distribution, because it is less frequent than the other states.

We applied all discretization methods using the known number of states. We measured the accuracy of the discretization by comparing the obtained state sequence with the true state sequence used for generating the data. The median accuracies and the deviations (AMAD) for $k = 5$ states and three levels of outlier contamination are listed in Table 1. We used the robust median and AMAD measures,

[1]the implementation in the Matlab Statistics Toolbox

[2]http://www.ai.mit.edu/~murphyk/Software/hmm.html



(a) Time series (b) PDE

Figure 3: Artificial data with 4 states and 5% outliers

because the empirical probability distributions of the accuracies were often right skewed and had some extreme values of low accuracy.

The Persist algorithm always has a higher median accuracy than any static method with large distances to the second best. The deviation is also much smaller than for the other methods, indicating high consistency. Even with 10% outliers, the performance of the new algorithm is still better than for any static method applied to the same data without outliers! Compared to the only other temporal method, HMM, the performance of Persist is slightly worse for 0% outliers. But with larger levels of outlier contamination, the HMM results degrade rapidly, even below the results from several static methods. The best competing methods for noisy data are EQF and SAX. The performance of both degrades rather slowly with increasing noise. Without outliers the GMM method shows the best performance among the static methods, but the accuracies are very bad with added noise. The methods using measures of location and dispersion (M±S,M±A) do not seem to be well suited for the generated datasets in general.

Table 1: Median accuracy for 5 states

Outliers	0%	5%	10%
EQF	0.74 ± 0.08	0.71 ± 0.08	0.69 ± 0.07
SAX	0.74 ± 0.09	0.74 ± 0.08	0.72 ± 0.08
EQW	0.67 ± 0.16	0.33 ± 0.09	0.32 ± 0.08
M±S	0.56 ± 0.11	0.48 ± 0.10	0.43 ± 0.09
M±A	0.51 ± 0.16	0.48 ± 0.15	0.45 ± 0.13
KM	0.71 ± 0.21	0.66 ± 0.22	0.61 ± 0.24
GMM	0.79 ± 0.18	0.27 ± 0.12	0.24 ± 0.11
HMM	0.94 ± 0.08	0.52 ± 0.34	0.44 ± 0.29
Persist	*0.90 ± 0.03*	*0.86 ± 0.03*	*0.83 ± 0.03*

The results for other values of k were similar. The absolute differences in accuracy were smaller for $k = 2, 3, 4$ and even larger for $k = 6, 7$. The performance of HMM degraded later w.r.t. outlier contamination for fewer states and earlier for more states. Figure 4 plots the median accuracies for 3 states, all methods, and all outlier levels. Again, the Persist method is always the best except for HMM at low outlier levels. EQW and GMM perform bad in general. All other methods degrade gently in the presence of noise, but are clearly inferior to Persist.

In order to check the results for statistical significance, we tested the hypothesis that the accuracy of the Persist is better than the accuracy of the competing algorithms.

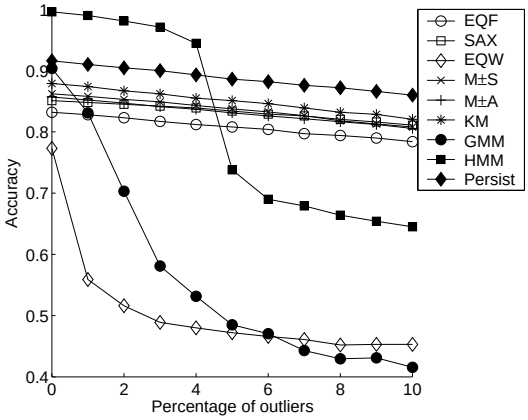


Figure 4: Median accuracy for 3 states

Since the accuracies do not follow a normal distribution, we used the robust (but weaker) rank sum test instead of the t-test. The test was performed for all k and all noise levels. For the competing static methods all p-values were smaller than 0.001, clearly indicating superior performance that can be attributed to the incorporation of temporal information. Compared to HMM, the results are significantly better for the larger amounts of outliers and worse for no of few outliers. The more states are present, the less robust HMM tends to be. Table 2 shows the result of the statistical tests between Persist and HMM. A plus indicates Persist to be better than HMM, for a minus the accuracy is significantly lower, circles are placed where the p-values were larger than 0.01.

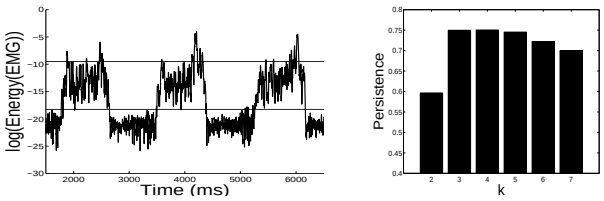
Table 2: Test decisions of Persist vs. HMM

Outliers	States					
	2	3	4	5	6	7
0%	−	−	−	○	+	+
1%	−	−	−	+	+	+
2%	−	−	○	+	+	+
3%	−	−	+	+	+	+
4%	−	○	+	+	+	+
5%	−	+	+	+	+	+
6%	−	+	+	+	+	+
7%	−	+	+	+	+	+
8%	−	+	+	+	+	+
9%	−	+	+	+	+	+
10%	○	+	+	+	+	+

In summary, the Persist algorithm was able to recover the original state sequence with significantly higher accuracy and more consistency than all competing static methods. The temporal HMM method is slightly better than Persist for no or few outliers, but much worse for more complicated and realistic settings with more states and outliers.

5.2 Real data

For real life data the states of the underlying process are typically unknown. Otherwise a discretization into recurring states wouldn't be necessary. We explored the behavior of the Persist algorithm in comparison with the other methods on datasets that clearly show several states. Due to lack of space the results of only one dataset are presented. Other datasets well suited for our method include e.g. the Power [9], the Auslan [8], and the Context Recognition data [18].



(a) Time series (b) Persistence

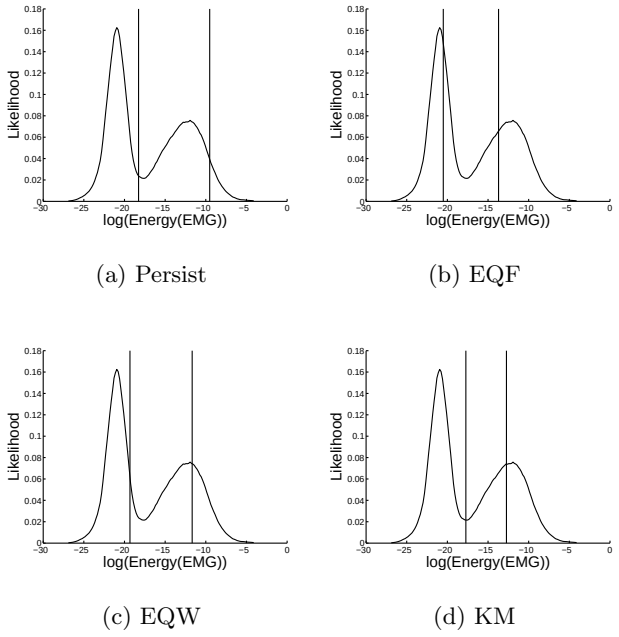
Figure 5: Muscle data

The Muscle dataset (donated to [9]) describes the muscle activation of a professional inline speed skater [20]. The muscle is expected to switch mainly between being active and relaxed states. A 5s excerpt of the data is shown in Figure 5(a). The muscle activation was calculated from the original EMG (Electromyography) measurements by taking the logarithm of the energy derived from a wavelet analysis and is displayed in arbitrary units. Figure 5(b) shows the persistence score for $k = 2, ..., 7$. The scores for $k = 3, ..5$ are almost the same. Consulting an expert we chose $k = 3$ states even though the score for 4 is slightly higher. The resulting bin boundaries of four selected methods are shown in Figure 6 as vertical lines on top of a probability density estimation plot. All methods except Persist place cuts in high density regions. EQF sets the first cut very close to the peak corresponding to the state of low muscle activation. This will result in a large amount of transitions between the first two states. EQW and KM do the same for the second peak in the density, corresponding to high muscle activation. Persist is the only method that places a cut to the right of the second peak. This results in a state for very high activation. The validity of this state can also be seen from Figure 5(a), where the horizontal lines correspond to the bins selected by Persist. The very high values are not randomly scattered during the interval of high activation but rather concentrate toward the end of each activation phase. This interesting temporal structure is not visible from the density plot and is not discovered by the other methods. The long interval of high activation is the gliding phase, where the muscle is needed to stabilize the body's center of gravity while gliding forward. The culmination of energy at the end is explained by the push off with the foot to generate forward propulsion.

6. DISCUSSION

The proposed quality score and algorithm for detecting persisting states has been shown to outperform existing methods on artificial data in the task of detecting the true underlying states. The results on the real dataset lead to more meaningful interval boundaries with a state of very high muscle activity, that was neglected by the other methods. Nevertheless the method is simple, exact, and easy to implement.

The only competing method using temporal structure was HMM. The EM estimation of HMM is more complex, needs



(a) Persist (b) EQF

(c) EQW (d) KM

Figure 6: Probability density estimate with bins for muscle data

a good initialization, is sensitive to noise, and only converges to a local maximum of the likelihood. The quality of HMM was better than that of Persist without noise, but degraded rapidly when outliers were added. HMM models are also harder to interpret than the result of binning methods like Persist.

In [4] EM like algorithms for the discovery of recurring states in time series are presented. An initial guess of the number of segments is required, however. This will be very difficult for long time series. We only require a rough guess for the number of states.

For the application of Persist, the time series need to be de-trended and should contain states defined by intervals on the value axis. This can be extended to handle states describing local trends e.g. with differentiation filters. Persisting states in multivariate time series can be discovered by discretizing each component and searching for coinciding states [19] or by incorporating the quality score in clustering methods. More research is needed on the suitability of the persistence score for selecting the number of states.

7. SUMMARY

Many time series discretization methods ignore the temporal order of values. The incorporation of temporal information leads to the unsupervised Persist algorithm. It achieves much higher accuracy than existing static methods and is robust against noise. It also outperforms HMM for all but very simple cases. The symbols created by the Persist method are likely to produce meaningful results in knowledge discovery tasks like rule generation or anomaly detection because they correspond to persisting states.

Acknowledgements: We thank Dr. Olaf Hoos, Department of Sports Medicine, Philipps-University Marburg, Germany, for the muscle data and interpretation of the results.

Reproducible Results Statement: All datasets and code used in this work are available by emailing the first author.

8. REFERENCES

[1] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical Report ICSI-TR-97-021, University of Berkeley, 1997.

[2] C. Daw, C. Finney, and E. Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74:916–930, 2003.

[3] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Int. Conf. on Machine Learning*, pages 194–202, 1995.

[4] A. Gionis and H. Mannila. Finding recurrent sources in sequences. In *Proc. 7th Int. Conf. on Computational Molecular Biology*, pages 123–130, 2003.

[5] S. K. Harms and J. Deogun. Sequential association rule mining with time lags. *Journal of Intelligent Information Systems (JIIS)*, 2003.

[6] M. L. Hetland and P. Saetrom. Temporal rule discovery using genetic programming and specialized hardware. In A. Lotfi, J. Garibaldi, and R. John, editors, *Proc. 4th Int. Conf. on Recent Advances in Soft Computing*, pages 182–188, 2002.

[7] M. L. Hetland and P. Saetrom. The role of discretization parameters in sequence rule evolution. In *Proc. 7th Int. Conf. on Knowledge-Based Intelligent Information & Engineering Systems, KES*. Springer, 2003.

[8] M. W. Kadous. Learning comprehensible descriptions of multivariate time series. In *Proc. 16th Int. Conf. on Machine Learning*, pages 454–463. Morgan Kaufmann, 1999.

[9] E. Keogh. UCR Time Series Data Mining Archive, *http://www.cs.ucr.edu/~eamonn/tsdma/index.html*, 2002.

[10] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In *Data Mining in Time Series Databases*. World Scientific Publishing Company, 2003.

[11] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *8th ACM SIGKDD, Edmonton, Canada*, pages 102–111, 2002.

[12] E. Keogh, S. Lonardi, and B. Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proc. 8th ACM SIGKDD*, Edmonton, Canada, July 2002.

[13] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[14] R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 114–119, 1996.

[15] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[16] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proc. 8th ACM SIGMOD, DMKD workshop*, pages 2–11, 2003.

[17] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, (6):393–423, 2002.

[18] J. Mäntyjärvi, J. Himberg, P. Kangas, U. Tuomela, and P. Huuskonen. Sensor signal data set for exploring context recognition of mobile devices. In *2nd Int. Conf. on Pervasive Computing*, Linz/Vienna, Austria, 2004.

[19] F. Mörchen and A. Ultsch. Discovering temporal knowlegde in multivariate time series. In *Proc. GfKl 2004, Dortmund, Germany*, 2004.

[20] F. Mörchen, A. Ultsch, and O. Hoos. Extracting interpretable muscle activation patterns with time series knowledge mining. *Int. Journal of Knowledge-Based & Intelligent Engineering Systems*, 2005.

[21] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of IEEE*, 77(2):257–286, 1989.

[22] J. J. Rodriguez, C. J. Alonso, and H. Boström. Learning first order logic time series classifiers. In *Proc. 10th Int. Conf. on Inductive Logic Programming*, pages 260–275, 2000.

[23] A. Ultsch. Pareto Density Estimation: Probability Density Estimation for Knowledge Discovery. In *Proc. GfKl 2003, Cottbus, Germany*, 2003.