Model-based Classification of Data with Time Series-valued Attributes

Christian Böhm
University of Munich
boehm@dbs.ifi.lmu.de

Leonhard Läer
Technische Universität München
leonhard.laeer@lrz.tum.de

Claudia Plant
Technische Universität München
plant@lrz.tum.de

Andrew Zherdin
Technische Universität München
andrew.zherdin@lrz.tum.de

Abstract: Similarity search and data mining on time series databases has recently attracted much attention. In this paper, we represent a data object by several time series-valued attributes. Although this kind of object representation is very natural and straightforward in many applications, there has not been much research on data mining methods for objects of this special type. In this paper, we propose a novel model-based classifier exploiting the benefits of representing objects by several time series. Classification decisions are based on class-specific interaction patterns among the time series of an object. Experimental results on benchmark data and real-world medical data demonstrate the performance of our classifier.

1 Introduction

Time series data are collected in many applications including finance, science, natural language processing, medicine and multimedia. The content of large time series databases can not be analyzed manually. To support the knowledge discovery process from time series databases, effective and efficient data mining methods are required. Often, the primary goal of the knowledge discovery process is classification, which is the task to automatically assign class labels to data objects. To learn rules, strategies or patterns for automatic classification, the classifier needs to be trained on a set of data objects for which the class labels are known. Typically, this so-called training data set has been labelled by human experts. Based on the patterns learned form the training data set, the classifier automatically assigns labels to new, unseen objects.

There is a huge volume of papers on classification of time series, e.g. [Kad99, KP97, Man95]. However, most existing approaches consider a single time series as the data object to be classified. In this paper, we focus on the representation of a data object by several time series-valued attributes and propose a novel algorithm for classification of such objects. Representing an object by a set of time series appears very natural and straightforward in many applications.

Consider for example a meteorological database storing measured data from different meteorological stations. Each station can be regarded as an object which is represented

by several time series-valued attributes, including temperature, rainfall and air pressure. Based on the measured data over a certain period, stations can be automatically classified e.g. into stations with normal weather conditions and stations with exceptional conditions which are presented to a human expert.

In many medical applications, it is natural to represent a data object by a system of time series. Consider for example EEG data: Electroencephalography measures the electric activity in the brain by an array of usually 64 electrodes which are distributed over the scalp [PM07]. It is very natural to consider the EEG of one person as a data object, where the 64 time series-valued attributes represent the 64 electrodes. Another example is functional magnetic resonance imaging (fMRI) [GVS+07]. This imaging modality yields time series of 3-dimensional images of the brain. Also in this case, it is reasonable to regard each subject as a data object which is represented by time series-valued attributes. Each 3-d pixel called voxel of the fMRI image is an attribute of time series type.

Increasing amounts of captured motion stream data are collected in multimedia applications [CLP07]. Gesture sensing devices, such as a CyberGlove usually contain multiple sensors to capture human movements. Motion classification is the task to automatically assign movements, usually performed by different people, into predefined motion classes and has important applications in gait analysis and virtual reality. Again, it is reasonable to regard each movement as a data object which is represented by the time series obtained from the different sensors of the gesture sensing device.

Although representing an object by a system of time series is very natural in many applications, there has not been much research on data mining methods for objects of this special type. We propose a classifier which exploits the benefits of this kind of object representation by focussing on interaction patterns between the time series within a data object. We derive a description of each class in terms of a system of linear models characterizing class-specific interaction patterns. Our experiments demonstrate that the information on interactions substantially improves the classification accuracy. In addition, the models provide interesting insights for understanding class-specific differences.

The remainder of this paper is organized as follows. After a brief survey of related work in the next section, we introduce our classifier in Section 3. Section 4 presents an experimental evaluation and Section 5 concludes the paper.

2 Related Work

To describe the interaction patterns within the time series of an object, we follow a model-based approach first introduced in [KKP+08] for the purpose of efficient compression of time series. To derive an approximation to speed up similarity search, in this approach each time series is represented by a combination of a set of specific reference time series. We use equivalent linear models to represent interactions of time series and a similar algorithm for model finding but with a very different objective.

In comparison to general time series classification, only relatively few papers focus on classification of multivariate time series. In [YSYT03] a decision tree for data with time-

series attributes is proposed. At the internal nodes of this tree, time series are stored and splits are performed based on dissimilarity between pairs of time series. The method selects upon split a time series which exists in data by exhaustive search based on class and shape information.

In [KS05] the authors present tClass, an algorithm for classifying multivariate time series data based on so-called meta-features. These meta-features model some kind of recurring substructure in the instances, such as strokes in handwriting recognition, or local maxima in time series data. The types of substructures are defined by the user, but the substructures are extracted automatically and are used to construct attributes. The parameter space defined by the meta-features is segmented into regions beneficial for classification using a heuristic technique. After feature construction, classification is performed with conventional propositional learning algorithms, such as decision trees.

For EEG data, some specialized classifiers have been proposed, e.g. in [PM07] to use EEG as a biometric for person identification. To identify persons with high accuracy based on EEG data with visual stimuli, after selection of the best discriminative channels classification is performed by a neural network.

Also in [PR01] a classification method especially designed for EEG with visual stimuli is proposed. This method extracts the visual evoked potential (VEP) which is embedded in the ongoing EEG and blurred by noise. The VEP contained in the gamma band is extracted by digital filters. Similar to [PM07], classification is performed by a neural network.

In [ZG02] a distance-coupled Hidden Markov model (HMM) is introduced for classification of EEG data. The distance-coupled HMM applies several HMMs to model multivariate time series. The link between the different HMMs is established by the fact that the state of one model at some particular time stamp t depends on the states of the other models at time stamp $t-1$. In their experiments the authors demonstrate the usability of coupled HMMs for EEG classification. In contrast to [ZG02], our models capture dependencies not between time points but between the dimensions. Our models are applicable to classes and not to single objects. So our models have a very different structure compared to HMMs.

3 Model-based Classifier

Our method for model-based classification involves 3 steps:

1. model finding,
2. model refinement,
3. classification.

In the training phase of the classifier, in the first step the interaction patterns of the time series within each class are characterized by a system of linear models. In the second step,

these models need to be refined to be suitable for classification. The final step is the classification of test objects.

Notations and Definitions. We represent each class as a system of linear models. Following [KKP+08], a single model can be defined as follows.

Definition 1 (Model) *A mathematical model $M = (\vec{X}, \vec{\alpha}, f)$ for a dependant variable Y (output) consists of the following parts:*

- *a set of* exploratory variables $X_1, \ldots, X_k$ *(inputs),*
- *a* mathematical function
$$f(\vec{X}, \vec{\alpha})$$
that is used to describe the dependency between Y and $X_1, \ldots, X_k$. $\vec{\alpha}$ denotes the model parameters also called coefficients of the model.

The general form of the model is given by $Y = f(\vec{X}, \vec{\alpha}) + \varepsilon$, where ε denotes the random error.

In this paper, we focus on linear models, i.e. f is a linear combination of the inputs.

Further, we consider a data set DS with n objects, i.e. $DS = \{O_1, ..., O_n\}$. Each object O_i is represented by d time-series valued attributes X_j, each consisting of a real-valued time series with m time points, i.e. $O_i = \{X_1, ...X_d\}$ and $X_j = \{t_1, ..., t_m\}$. For the training data, a categorical class label is known. To facilitate our argument, we focus on a 2-class problem with class labels *class1* and *class2*. However, our method can be straightforwardly applied in a multi-class setting.

Model Finding. To capture the interaction patterns of time series within a class, e.g. *class1* we first have to generate a summarization of the time series attributes of the objects of the class. For each of the d dimensions, we concatenate the time series of all objects of the class. As a result we have a novel object O_s with d time series-valued attributes with $|class1| \cdot m$ time points. Now, we can derive a system of linear models as specified in Definition 1 for O_s. All d attributes of O_s are applied as output Y once. The idea is to try to explain each individual attribute by a linear combination of (potentially all) other attributes. To avoid over fitting, models with few inputs are preferable. In addition, an exhaustive search for all combinations of inputs is not feasible for large d. To cope with these problems, stepwise algorithms are often a good choice [Lar06, Geo00].

We propose an iterative stepwise algorithm for model finding which is displayed in pseudocode in Figure 1. We start with the output Y and the set of inputs X. Our purpose is to find a subset of X containing only that inputs which are most relevant to explain Y. To achieve a balance between model complexity and goodness of fit, we apply the Bayesian Information Criterion (BIC) [Geo00]. Theoretically well founded in Bayesian statistics and information theory, this criterion penalizes overly complex models and allows us to select the most relevant inputs in a parameter-free way. Our algorithm starts with an empty set $relevantInputs$. It then iteratively either adds a new input to $relevantInputs$ or

```
algorithm findModel(Output Y, Inputs X): Model M
  relevantInputs := ∅;
  oldMIN := +∞;
  do {
//(1) find a relevant input
      [minPlus, x+] := min_{x∈(X\relevantInputs)} (BIC(relevantInputs ∪ {x}, Y));
      [minMinus, x−] := min_{x∈relevantInputs} (BIC(relevantInputs \ {x}, Y));
      if(min(minPlus, minMinus) > oldMIN) then
// (2) no more relevant inputs
        break;
      end if
      if(minPlus < minMinus) then
        relevantInputs.add(x+);
        oldMIN := minPlus;
      else
        relevantInputs.remove(x−);
        oldMIN := minMinus;
      end if
  } while()
// (3) build the Model with inputs
  M := new Model(relevantInputs, Y);
return M;
```

Figure 1: Algorithm for Model Finding.

removes an already existing element depending on which of these two actions leads to a greater improvement of BIC (1). Based on these $relevantInputs$ the model is constructed by least-squares fitting. The algorithm terminates if no further improvement of BIC can be achieved (2). The algorithm finally returns the model based on $relevantInputs$ (3).

Model Refinement. After application of the model finding algorithm using each attribute of O_s as an output, we obtain a set of models SM consisting of d models per class. However, these models can not be directly applied for classification. SM may contain models with large error and models representing general trends in the data which are not class-specific. Therefore the set of models needs to be refined to build a classifier with good discriminatory power. As depicted in pseudocode in Figure 2, the algorithm for model refinement involves two steps: First, we exclude models with large error since they do not capture any distinct interaction information. This is implemented by removing all models having an error larger than three times the median of the error on training data (1). From the remaining models we select those which discriminate between classes. For all classes and models, we compute the cumulative error on the training data. Models having a smaller cumulative error for the correct class than for the wrong class are included in the classifier (2).

Classification. The classification of the test objects is now simple. To classify an object, we sum up the mean square error for all relevant models for all classes. We assign the object to the class with the smallest mean square error.

```
algorithm refineModels(set of models SM, training objects O): refined set of models RSM
// (1) remove corrupt models
   median := median error of SM;
   for each i ∈ {1 ... SM.size()} do
     if(SM[i].error > 3 · median) then
       SM.remove(i);
     end if
   end for

// (2) find class separating models

   for each m ∈ {1 ... SM_class1.size} do
     for each o ∈ {1 ... obj_class1.size} do
       difference[i]+ = mean_square_error(m, o);
     end for
     for each o ∈ {1 ... obj_class2.size} do
       difference[i]− = mean_square_error(m, o);
     end for
   end for
   for each m ∈ {1 ... SM_class2.size} do
     for each o ∈ {1 ... obj_class1.size} do
       difference[i]− = mean_square_error(m, o);
     end for
     for each o ∈ {1 ... obj_class2.size} do
       difference[i]+ = mean_square_error(m, o);
     end for
   end for

// result

   for each i ∈ {1 ... difference.size} do
     if(difference[i] < 0) then
       RSM.add(i);
     end if
   end for
return RSM;
```

Figure 2: Algorithm for Model Refinement.

Runtime Complexity. The worst-case runtime complexity of the stepwise model finding algorithm is quadratic in d. Usually, the number of time points m is much larger than the number of dimensions d. Due to matrix inversion required for least square model fitting, we then have cubic complexity in m. The model refinement step, as well as the classification of an object are quadratic in d and linear in m.

4 Evaluation

Data Sets. Table 1 provides a summary on the data sets used for evaluation. Data sets 1 to 8 are different EEG data sets available at the UCI machine learning repository[1]. These data sets are derived from a study on alternations of brain activity in alcoholic subjects in comparison to a control group. Each subject was exposed to either a single visual stimulus (S1) or to two visual stimuli (S1 and S2). When two stimuli were shown, they were presented in either a matched condition where S1 was identical to S2 or in a non-matched condition where S1 differed from S2. Data sets 1 to 4 correspond to the so-called

[1] http://archive.ics.uci.edu/ml/databases/eeg/eeg.data.html

Name	Description	Data Objects	#Objects	#Classes	#Dimensions	Length
DS1	EEG single stimulus	subjects	40	2	64	approx. 2560
DS2	EEG matched stimulus	subjects	40	2	64	approx. 2560
DS3	EEG non-matched stimulus	subjects	40	2	64	approx. 2560
DS4	EEG combined	subjects	40	2	64	approx. 7680
DS5	EEG single stimulus	runs	20	2	64	256
DS6	EEG matched stimulus	runs	20	2	64	256
DS7	EEG non-matched stimulus	runs	20	2	64	256
DS8	EEG combined	runs	20	2	64	768
DS9	fMRI	subjects	26	2	90	325
DS10	motion stream	signs	2565	95	22	approx. 57

Table 1: Data Sets.

Large Data Set at UCI where training and test data are available. The task is to classify the subjects into the classes alcoholic and control based on 10 time series-valued attributes. To generate a representation of the objects of a class, we concatenated the time series of all objects and all runs resulting in a time series of 2,560 time points. Data sets 5 to 8 are are derived from the so-called Small Data Set which contains data of 2 subjects, one alcoholic and one of class control. For each of the 3 experimental paradigms, 10 runs of EEG have been recorded. The task here is to classify the EEG runs to the classes alcoholic and control.

DS9 [GVS+07] consists of 26 functional MRI images. This data set has been obtained from a study on somatoform pain disorder and consists of images of 13 diseased subjects and 13 healthy controls. For classification we selected 90 regions of interest as proposed in [TMLP+02].

DS10, also from UCI, [2] contains motion stream data of Australian sign language. For each of the 95 Australian language signs, 27 examples were captured from a native signer using high-quality position trackers and instrumented gloves.

Comparison Methods and Validation. For comparison, we implemented two basic classifiers: Nearest neighbor (1NN) and classification to the nearest class centroid (centroid). Both basic classifiers do not consider interaction patterns among time series. To classify a test object, for all training objects and all attributes the nearest neighbor classifier sums up the Euclidean distance and assigns the object to the class of that training object with the smallest overall distance. The centroid classifier computes for each class the centroid by averaging all time series-valued attributes of all training instances. A test object is assigned to the label of the closest centroid.

We obtained the code of TClass from the authors [KS05]. The parametrization of this method is very difficult, since the type of meta-features as well as the base learner need to be selected by the user. As recommended in [KS05], we use a universal set of meta-features for temporal domains. In addition, we use following aggregate global attributes: mean, min, max and mode. Mean, max and min were recommended by authors for DS10.

[2] http://kdd.ics.uci.edu/databases/auslan2/auslan.data.html

Data Set	Centroid	1-NN	Models	TClass	Others
DS1	50%	50%	**100%**	70%	n.a.
DS2	55%	55%	**90%**	74%	n.a.
DS3	80%	55%	**100%**	72%	n.a.
DS4	60%	50%	**100%**	72%	96% [PR01]/ 98% [PM07]
DS5	65%	90%	**98%**	89%	n.a.
DS6	40%	80%	**100%**	86%	n.a.
DS7	90%	90%	**100%**	89%	n.a.
DS8	85%	90%	**100%**	91%	90% [ZG02]
DS9	85%	77%	**96%**	58%	n.a.
DS10	11%	46%	75%	96%	**98%** [KS05]

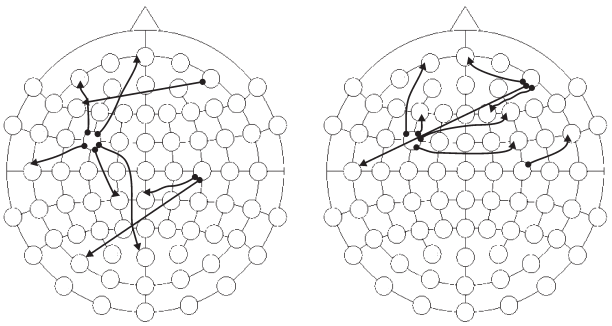
Table 2: Results.

Mode improved results on DS1-DS8. In all experiments, we use AdaBoost with J48 decision tree as learner. According to the experiments in [KS05], this is the third-best base learner for TClass. Two slightly better learners are Naive Segmentation and voting. Although Naive Segmentation is implemented, we could not repeat the very good results. We do not use voting since it is not implemented and it is unclear, which voting algorithm was used in [KS05]. Because TClass is not deterministic, unless otherwise specified, we report the mean of accuracy of 10 runs.

In addition we compare our method to [PM07, ZG02, PR01], 3 classifiers especially designed for EEG data (cf. Section 2). For DS1-DS4 train and test data are given. Classification results are therefore directly comparable. For DS5-DS8 we implement 5-fold cross-validation as in [ZG02]. We do not provide a comparison to [YSYT03] since the authors arbitrarily selected runs from the full EEG data set. We validate DS9 with leave-one-out, since there are only few instances. For DS10 we implement 10-fold cross-validation as in [KS05].

Results. Our model-based classifier demonstrates an excellent accuracy of 100% on 6 out of 10 data sets and strongly outperforms the two basic classifiers. Table 2 summarizes the results. On EEG data, our classifier even outperforms the classifiers especially designed for this type of data [PM07, ZG02, PR01]. These results demonstrate that the information on interaction patterns is highly relevant for the classification of EEG and fMRI data. On data set DS9 (fMRI) our model-based classifier shows very good results. Only one object is incorrectly classified. The most important models for classification decision, are judged as reasonable by domain experts.

Only on DS10, TClass performs significantly better than all comparison methods. But notice that DS10 has been originally published by the authors of [KS05] and TClass has been designed for motion stream data. With 75% in accuracy, our model-based classifier performs much better than the basic methods centroid and 1-NN. This indicates that attribute interactions are also useful for the classification of motion stream data. However, evidently, the interactions in this data set can not perfectly be captured by linear models.



(a) Alcoholic. (b) Control.

Figure 3: Illustration of Best Class-separating Models on UCI EEG Data.

We could not reproduce the accuracy of TClass reported in the publication (98%) which was obtained using voting, since it is not clear which voting strategy has been applied. In addition, TClass with voting is very inefficient. The run[3] without voting for DS10 lasts approximately 24 hours. For 11 voters, as recommended in [KS05] a naive voting algorithm would take 11 days. The model-based classifier run needs less than 40 minutes under equal conditions. Moreover, our algorithm is parameter-free and deterministic, whereas TClass exhibits a very high variation of results. E.g. for DS1 accuracies between 20% and 85% are obtained.

As an additional advantage, the interaction models generated by our classifier provide interesting information for interpretation. Figure 3 displays the 3 most relevant models for separating the classes on DS1. For both classes, the layout of the 64 electrodes on the scalp is displayed together with the best separating interaction patterns. The models of the alcoholic subjects involve different brain regions than those of the healthy control group. The subtle differences of interactions need to be systematically evaluated in future work.

5 Conclusion

In this paper, we proposed a novel model-based classifier for data with time series-valued attributes. Classification decisions are supported by class-specific interaction patterns within the time series of a data object. The experimental evaluation demonstrates that interaction patterns characterized by linear models are very useful for classification, especially of EEG and fMRI data.

In ongoing work we focus on interpretation of the identified models. We will interpret the models of fMRI for somatoform pain disorder patients to find the most important regions. In addition, we are currently evaluating the classifier on EEG data from our cooperation partners for the identification of subjects in different stages of anesthesia. It would also be

[3] 10-fold cross-validation; 33 runs; Intel Core 2 Quadro(2,66 GHz), 8 GB RAM

interesting to design model-based classifiers for fMRI data and combined fMRI-EEG data sets which is very challenging because of the large number of time series in fMRI data. We also plan to evaluate the classifier on time series data from other domains.

Furthermore, we want to extend our model notion to more general models. Among building blocks to be investigated are differential equations and kernel functions.

References

[CLP07] L. Khan C. Li and B. Prabhakaran. Feature Selection for Classification of Variable Length Multiattribute Motions. In V. A. Petrushin and L. Khan, editors, *Multimedia Data Mining and Knowledge Discovery*. Springer, 2007.

[Geo00] E. I. George. The variable selection problem. *J. Amer. Statist. Assoc*, 95:1304–1308, 2000.

[GVS+07] H. Gündel, M. Valet, C. Sorg, D. Huber, C. Zimmer, T. Sprenger, and T.R. Tölle. Altered cerebral response to noxious heat stimulation in patients with somatoform pain disorder. *Pain.*, 137:413–421, Nov 2007.

[Kad99] M. W. Kadous. Learning Comprehensible Descriptions of Multivariate Time Series. In *ICML*, pages 454–463, 1999.

[KKP+08] H.-P. Kriegel, P. Kröger, A. Pryakhin, M. Renz, and A. Zherdin. Approximate Clustering of Time Series Using Compact Model-Based Descriptions. In *DASFAA*, pages 364–379, 2008.

[KP97] A. Kehagias and V. Petridis. Predictive Modular Neural Networks for Time Series Classification. *Neural Networks*, 10(1):31–49, 1997.

[KS05] M. W. Kadous and C. Sammut. Classification of Multivariate Time Series and Structured Data Using Constructive Induction. *Mach. Learn.*, 58(2-3):179–216, 2005.

[Lar06] D. T. Larose. *Data Mining Methods and Models*. John Wiley & Sons, 2006.

[Man95] S. Manganaris. Learning to classify sensor data, 1995.

[PM07] R. Palaniappan and D. P. Mandic. EEG Based Biometric Framework for Automatic Identity Verification. *VLSI Signal Processing*, 49(2):243–250, 2007.

[PR01] R. Palaniappan and P. Raveendran. Single trial VEP extraction using digital filter. *Statistical Signal Processing, 2001. Proceedings of the 11th IEEE Signal Processing Workshop on*, pages 249–252, 2001.

[TMLP+02] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated Anatomical Labeling of Activations in SPM Using a Macroscopic Anatomical Parcellation of the MNI MRI Single-Subject Brain. *NeuroImageVolume 15*, pages 273–289, January 2002.

[YSYT03] Y. Yamada, E. Suzuki, H. Yokoi, and K. Takabayashi. Decision-tree Induction from Time-series Data Based on a Standard-example Split Test. In *ICML*, pages 840–847, 2003.

[ZG02] S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification, 2002.