

Learning to Classify Sensor Data ^{*}

Stefanos Manganaris [†]

*Dept. of Computer Science
Vanderbilt University
Box 1679, Station B
Nashville, TN 37235, U.S.A.*

Abstract

Learning how to classify sensor data is one of the basic learning tasks in engineering. Data from sensors are usually made available over time and are classified according to the behavior they exhibit in specific time intervals. I address the problem of classifying finite, univariate, time series that are governed by unknown deterministic processes contaminated by noise. Time series in the same class are allowed to follow different processes. In this context, I investigate the appropriateness of using induction algorithms not specifically designed for temporal data. I present a simple supervised induction algorithm that uses serial correlation as its inductive bias in a Bayesian framework, and compare it empirically to a popular general-purpose classifier, in a NASA telemetry monitoring application. Two comparisons were performed: one in which the general purpose classifier was applied directly to the data, and another in which features that captured serial correlations were extracted before the induction. Serial correlation appeared to be an important form of inductive bias, most effectively utilized as an integral part of the learning algorithm. Feature extraction occurs too early in the training process to utilize correlation knowledge effectively.

Keywords: electrical domain, classification, time series sensor data, serial correlation as inductive bias, supervised Bayesian learning, minimum description length.

^{*}This research has been supported in part by a grant from NASA Ames (NAG 2-834).

[†]Tel. (615)343-4111, (615)343-8006 (fax). E-mail: stefanos@vuse.vanderbilt.edu.

1 Introduction

The problem of classifying sensor data is common in engineering applications. For example, a monitoring system distinguishes between normal and abnormal sensor readings; a diagnosis system further classifies abnormal data according to the fault causing them. Over the last few years there has been a significant increase in the development of knowledge-based systems for classifying sensor data. At the same time, performance improvement in classification tasks has been a traditional area of machine learning research. In this paper, I discuss machine learning techniques for acquiring knowledge pertinent to the classification of sensor data.

In general, sensor data are represented by a multivariate time series, where each variable corresponds to a sensor. I limit the scope of the problem by only considering finite, univariate, time series that are governed by deterministic processes. In other words, I only consider a single sensor over a finite interval of time. Moreover, I assume that the sensor monitors deterministic phenomena: the state of the phenomenon at one time determines its state at any later time. Extraneous noise added to the data may hide the underlying process. Time series are to be classified according to the underlying phenomenon; note, however, that different processes may be associated with the same class. The problem is to infer a good classification procedure, in terms of classification accuracy and computational efficiency, from a set of preclassified training data.

In most of the machine learning literature, the objects involved in classification tasks are described in terms of properties that remain constant over time (attributes). Sensor data, however, are most naturally described in terms of properties that take a series of values over time; each property corresponding to a sensor. To use one of the well established learning algorithms with such temporal data, one can simply treat each property at each time point as an attribute. Alternatively, one can preprocess each object to extract other informative, time-invariant, properties; for instance, Smyth and Mellstrom in [SM92] extract autoregressive coefficients to classify stationary, stochastic, time series.

In this study, I investigate the appropriateness of using general purpose induction algorithms with temporal data, in the context of deterministic time series. I propose a learning algorithm that works directly with temporal data. The algorithm takes a Bayesian approach and uses serial correlation as its inductive bias. It has been implemented as part of an experimental system called CALCHAS. I empirically evaluate the

algorithm in a NASA telemetry monitoring problem, by comparing its performance, in terms of attained classification accuracy, with C4.5, a popular algorithm not designed specifically for temporal data [Qui93]. Two comparisons are performed: one in which C4.5 is applied directly to the original data, and another in which features that capture serial correlations are extracted.

C4.5 coped much better with noise when features were extracted. However, CALCHAS performed consistently better than C4.5 under moderate noise and not worse than C4.5 under low noise.

Serial correlation is an important form of inductive bias in temporal domains. I argue that, for deterministic time series, serial correlation is best utilized as an integral part of a specially designed learning algorithm.

2 A Bayesian Classifier for Sensor Data

A Bayesian classifier assigns an object to the class with highest posterior probability; in other words, it classifies an object described by data D to a class C when

$$P(C|D, I) \geq P(C_j|D, I)$$

for all C_j classes, where I stands for a state of prior knowledge. When the posterior probabilities can be computed precisely, this classifier is optimal, in terms of classification accuracy [BFOS84].

For the problem at hand, the probabilities can only be estimated with some confidence from the data and prior knowledge available. Using Bayes rule, the posterior probability of a class C is

$$P(C|D, I) = P(C|I) \frac{P(D|C, I)}{P(D|I)}, \quad (1)$$

where $P(C|I)$ is the prior probability of C , $P(D|C, I)$ is the likelihood of C , and $P(D|I)$ is the prior probability of the data. Note, $P(D|I)$, can be ignored when comparing the probabilities of alternative classifications.

The prior probability of a class, when not available as prior knowledge, can be estimated from the training data, as the frequency with which instances of the class

occur, by assuming that the training set is a random sample of the true population of instances.

To estimate the likelihood of a class for an instance, I rely on the existence of a deterministic process governing the evolution of the time series. I assume each instance follows one such process. Instances in the same class are allowed to follow separate, distinct, processes; in other words, two instances in the same class may look different—this is analogous to disjunctive concept learning. Let $\{M_1, M_2, \dots, M_k\}$ be a set of models associated with a class C , where each model M_i approximates one of the possible deterministic processes that govern the behavior of instances in C . Each model M_i is a function that approximates the value of the time series at time t with error e_t . The likelihood of a model M_i can be estimated by assuming that noise follows Gaussian distributions with mean zero and variance σ_t^2 , independent from point to point. In that case, the quantity

$$\chi^2 = \sum_{t=1}^N \frac{e_t^2}{\sigma_t^2}$$

follows a chi-square probability distribution with N degrees of freedom (denoted χ_N^2). I estimate the likelihood of a model M_i as the probability with which a χ_N^2 random variable exceeds the observed χ^2 value:

$$P(D|M_i, I) = \Pr(\chi_N^2 \geq \chi^2).$$

The likelihood of a class for an instance is then estimated as the maximum likelihood over the set of models associated with the class, since each instance follows a single process:

$$P(D|C, I) = \max_i P(D|M_i, I).$$

2.1 Inducing Disjunctive Class Models

The proposed algorithm for classifying instances relies on a set of approximation models for the processes characterizing each class. It also requires the variance σ_t^2 of the noise at time t . In this section, I propose a simple algorithm for inducing this information from the training set. Given a set of preclassified instances, the goal is to induce the models best supported by the evidence in the data.

The algorithm partitions the set of instances in each class into disjoint and non-empty subsets, so that each subset is characterized by a single process. For each of

the resulting subsets, it then induces an approximation of the underlying process. The end result for each class is a set of models in one-to-one correspondence with the subsets of the partition; I term this set of models a *k-model* for the class. I term each of the *k* models a *disjunct*. I will refer to the partition corresponding to a *k-model* as the *k-partition*.

There is an infinite number of plausible *k-models* for each value of *k* from 1 to the size of the training set. To make the problem manageable, I only seek to select the best *k-model* from a flexible family of models. My goal is to induce the *k-model* with highest posterior probability, given the training data and a state of prior knowledge. Bayes' rule expresses the posterior probability of a model in terms of the product of its likelihood and its prior probability. The likelihood provides an indication of the model's power to explain the data. At the same time, the prior probability can be used to capture the natural preference for simple models ("Occam's razor"). Bayes' rule balances, in effect, the complexity of a model and its ability to explain the data, mitigating the problem of overfitting. Additional model complexity is only justified when there is enough unaccounted structure in the data to benefit from it and the benefit exceeds the cost of the added complexity.

To assign priors, I use the minimum message length principle [Che90, Ris87, WF87]. Intuitively, consider the transmission of information from a sender to a receiver. If the messages to be transmitted are drawn from a population with some distribution of probability, then the most efficient code for communication assigns to each message *m* with probability $P(m)$ an encoding of length $-\log P(m)$. Based on this fact, one can estimate the probability of a message from its shortest possible encoding. In the problem at hand, messages describe models. The negative logarithm with base two of the prior probability of a model is computed as the length in bits of the shortest possible message communicating that model to a receiver aware of the prior knowledge.

The *k-models* in the space I consider are built from disjuncts represented by piecewise-polynomial functions. Each disjunct is a function that can be decomposed into a set of polynomials and intervals, with one polynomial per interval. Continuity is not enforced across intervals, since in practice abrupt transitions are common, for example when systems switch operating modes.

Inducing a *k-model* requires finding the value of *k* and the *k-partition* that yield the most probable *k-model*. The total number of partitions grows exponentially with the size of the training set. To search this space efficiently for a near optimal partition,

I use a greedy algorithm that avoids partitions unlikely to yield k -models with high posterior probability. The algorithm populates the subsets of the target partition in sequence. For each subset, it selects instances from the training set that are likely to belong to it. After an instance is incorporated into a subset, the algorithm induces the best disjunct for the subset, and uses it to select the next most likely instance to incorporate. When none of the remaining instances is appropriate, the algorithm starts populating a new subset. Instances are selected based on their rms-error, and are deemed appropriate or not by a standard chi-square test for fitness.

Disjuncts are induced by searching the space of piecewise-polynomial models of up to a specific degree. Each disjunct is defined over the longest time interval valid for instances in the corresponding subset of the training set. For each disjunct, the induction algorithm considers all possible partitions of its time interval into subintervals, finds the best polynomial for each subinterval, and selects the best overall piecewise polynomial. As for k -models, the “goodness” of a piecewise polynomial is judged by its posterior probability. This part of the algorithm is adopted from an algorithm presented by Pednault in [Ped89], originally used for surface reconstruction in computer vision. Although the general strategy is the same, the probability measures are computed differently.

The number of possible partitions for an interval grows exponentially with the length of the time series. The space of partitions can, however, be searched in $O(n^2)$ time (where n is the length of the time series) using dynamic programming: the optimal piecewise polynomial can be found incrementally, by considering progressively larger intervals, while storing intermediate results for reuse by concatenation.

The optimal polynomial for a subinterval is found by considering all polynomials whose degree does not exceed a set limit. To facilitate probabilistic predictions, I assume a Gaussian noise model and independence of sampling errors. I also assume that the variance of the noise distribution is constant over an interval. The likelihood of a given polynomial Π for n data points D is then

$$P(D|\Pi, I) = \int \dots \int \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \exp \left[-\frac{\sum_{i=1}^n e_i^2}{2\sigma^2} \right] de_1 de_2 \dots de_n, \quad (2)$$

where e_i is the error between the value predicted by the polynomial for point i and the actual data value. All integrals are over the uncertainty in the observed error. The best polynomial of degree ν is found in two steps. The algorithm first computes the $\nu + 1$ coefficients that minimize the sum of squared errors in equation (2), using

standard least-squares multiple regression; these coefficients maximize the model’s likelihood. Equation (2) is further maximized when the noise variance is set to

$$\sigma^2 = \frac{\sum_{i=1}^n e_i^2}{n}. \quad (3)$$

In the second step, the algorithm shifts its attention to the model’s posterior probability and considers the model’s shortest encoding. The computed coefficients can be represented with arbitrary precision. A very precise encoding of a coefficient requires many bits, decreasing the prior probability of the model, but minimizes the sum of squared errors, increasing its likelihood. The optimal precision trades message length for likelihood, and vice-versa, to maximize the posterior probability. To estimate the optimal precision I use a standard golden-section search algorithm [PTVF92].

To compute the length of the optimal encoding of a k -model, I first encode the number of disjuncts. For each disjunct I encode the number of subintervals and the start time of each (with accuracy depending on the sampling rate of data points), the degree and coefficients of all polynomials (each with its computed accuracy), and the variance of the noise for each interval.

A positive integer i is coded with $\log_2(m + 1)$ bits, when m is a known upper bound for i ; this is equivalent to a uniform prior. When there is no bound in our state of prior knowledge, I use the universal prior [Ris83, Eli75]: $\log_2^*(i + 1) + c$, where $c \approx 2.865$ and where $\log_2^* x \equiv \log_2 x + \log_2 \log_2 x + \dots$ up to, but not including, the first negative term.

The encoding length for floating point numbers is determined by their estimated optimal precision. For the noise variance I do not use a uniform prior; instead, I use an encoding scheme that penalizes large deviations from the expected variance provided as prior knowledge.

The algorithm is computationally efficient. An instance is classified in time $\Theta(m)$, where m is the total number of disjuncts in the domain. Training requires $O((n_1^3 + \dots + n_k^3)l^2p^4)$ time for each class, where k is the number of disjuncts, n_i is the number of training instances corresponding to disjunct i , l is the length of the time series, and p is the maximum polynomial degree considered.

3 Experiments in Telemetry Monitoring

The Electrical Generation and Integrated Loading (EGIL) controllers at NASA monitor telemetry data from the Shuttle to detect various events that take place onboard.

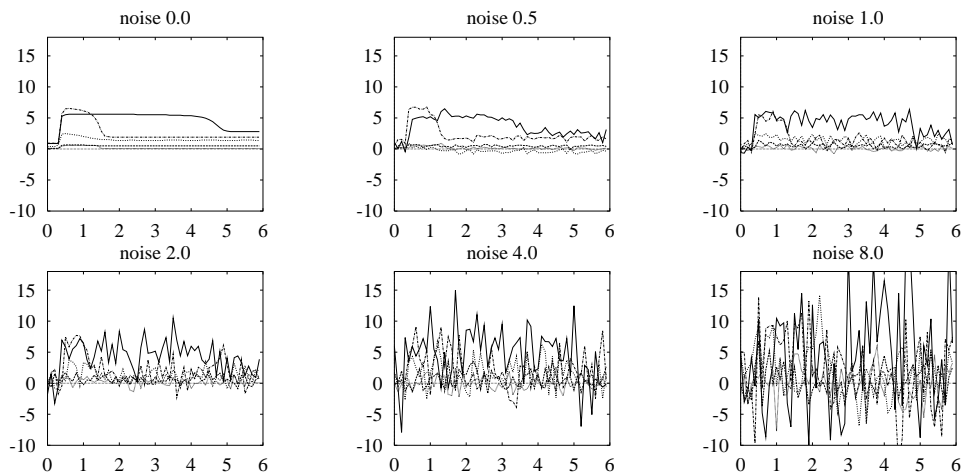


Figure 1: Typical signatures in the five classes considered in the paper, observed under various noise levels.

An event may be the onset or termination of operation of an electrical device on a power bus. Each event has a signature with a set of distinguishing characteristics, based on which the controllers identify them. There are over two hundred different events of interest, making their accurate identification a challenging task.

A signature is extracted from the telemetry stream whenever a monitored current undergoes a change that exceeds a preset threshold; it consists of 60 samples taken periodically over the six seconds following the triggering change (Figure 1). Signatures' baselines are normalized by subtracting the DC component. Each signature is a response to an underlying deterministic phenomenon, whose characteristics may vary, even for signatures of the same event (Figure 2).

I used six sets of signatures in the experiments. Each set had a total of 600 signatures, distributed in five classes with roughly the following proportions: 10%, 40%, 25%, 15%, 10%. Each of the six sets consisted of signatures observed under a given noise level. Noise level 0.0 was assigned to the original, very low-noise, data set, which was used as a basis for creating the other five sets, by adding white noise with variance proportional to the noise level.

With this data, a naive algorithm always picking the most frequent class can achieve a 40% classification accuracy, regardless of the level of noise. The optimal Bayes rate is not known, but the results below suggest it's very close to 100% under low noise levels.

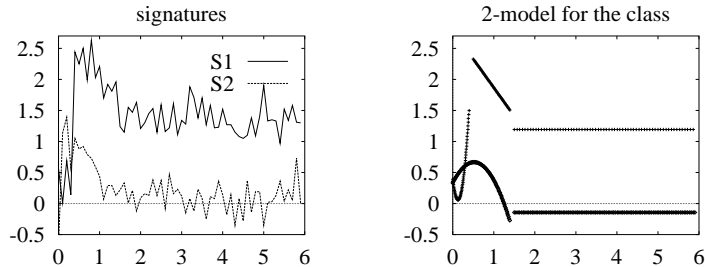


Figure 2: Two signatures, S_1 and S_2 , governed by different processes, although they are in the same class (left). The 2-model induced by CALCHAS for the class (right). The two signatures and the training data from which the model was induced were observed under noise level 0.5.

I used the EGIL data to evaluate empirically the performance of CALCHAS and compare it to the performance achievable by algorithms not specifically designed for temporal data. I chose C4.5 as a representative of such classifiers because of its popularity and availability.

I used the percentage of correctly classified instances as the dependent measure of performance. Each algorithm was evaluated under all six levels of noise, using stratified 10-fold cross-validation.

C4.5 was used in two different ways. In the first set of runs, C4.5 was directly applied to the raw data, treating, in effect, each data point as an attribute value. In the second set of runs, the data presented to C4.5 were preprocessed to extract informative features. A feature is informative when its values are nearly unique to certain classes. Naturally, there is some arbitrariness in selecting features. I extracted 12 features from each signature. I divided the interval over which a signature is defined into 12 subintervals of equal length. Each feature was the average of the data points in its corresponding subinterval. The rationale behind this set of features is that instances in the same class tend to have peaks and lows at roughly the same spots. By averaging points in groups some of the noise is smoothed out.

Both CALCHAS and C4.5 performed equally well at noise level 0.0. The performance of C4.5, when applied directly to the raw data, deteriorated rapidly as noise levels rose. With preprocessed data, C4.5 coped much better with noise; nevertheless, CALCHAS' performance degraded at a slower rate (Figure 3). The statistical significance of

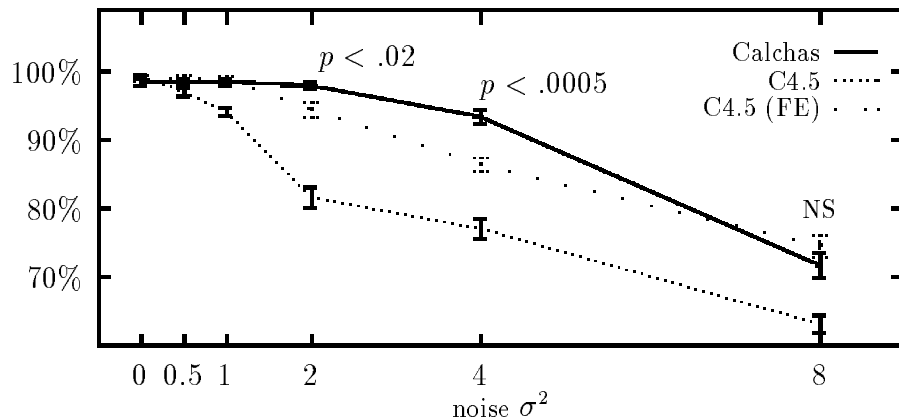


Figure 3: Classification accuracies under various noise levels for CALCHAS, C4.5 with feature extraction, and C4.5 without feature extraction. Each point indicates the mean and standard error of stratified 10-fold cross-validation runs. The p -values indicate the significance of the differences between CALCHAS and C4.5 with feature extraction; they were computed by the Wilcoxon rank sum test. “NS” denotes differences not statistically significant.

the differences in accuracy between CALCHAS and C4.5 with feature extraction was evaluated by a Wilcoxon rank sum test (a nonparametric test for comparing the medians of two independent samples of cardinal or ordinal data [HW73]); p -values are shown at each point. CALCHAS’ performance gains showed under moderate noise levels; these gains were all found to be statistically significant. Under low or high noise levels CALCHAS performed no worse than C4.5 (at noise level 8.0, CALCHAS performed slightly worse, but this difference was not found statistically significant.)

4 Discussion

I have presented a supervised induction algorithm for time series governed by deterministic processes. The algorithm is simple, utilizing well known techniques for conducting inferences from incomplete information. It is robust, providing an estimate of the confidence in each classification it makes. It is also efficient, and performed

well compared to a popular alternative, on data from a natural domain, under a wide range of reasonable amounts of noise.

The improved performance of the proposed algorithm can be attributed to its ability to utilize well serial correlation as inductive bias. Temporal data are treated as sequences of values generated by deterministic processes. CALCHAS is designed to search for models that characterize the processes associated with the classes. In contrast, C4.5 treats each data point as an independent attribute; it is not designed to take into account any correlation across data points.

The dimensionality of the space of concept instances in temporal domains is proportional to the length of the time interval. This usually leads to high-dimensional spaces, which have a great potential for complex concepts, and which can only be sparsely represented by training sets of reasonable size [Bel61]. Concept learning in high-dimensional spaces requires strong induction biases; serial correlation is a particularly appropriate one for temporal domains.

In the absence of noise, correlation prior knowledge does not offer a critical advantage. The characteristic peaks and valleys are clearly observable, and algorithms not specifically designed for temporal data can do well. Serial correlation provides a basis for dealing with noise. The averaging process I used in extracting features for C4.5 utilized, to some extent, this prior knowledge. Even though performance improved markedly as a result, C4.5 did not perform as well as CALCHAS. Feature extraction occurs too early in the learning process to allow effective use of the prior knowledge: at a preprocessing stage, instances are only considered individually. For deterministic time series, it appears that serial correlation is best utilized as an integral part of a specially designed learning algorithm.

Acknowledgments

I thank Doug Fisher for supervising and supporting this work, and Phil Laird, Robert Shelton, Ron Saul, Wayne Iba, and Deepak Kulkarni for providing the EGIL data and related information.

References

[Bel61] R. E. Bellman. *Adaptive Control Processes*. Princeton Univ. Press, 1961.

- [BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.
- [Che90] Peter Cheeseman. On finding the most probable model. In J. Shragar and P. Langley, editors, *Computational Models of Discovery and Theory Formation*, chapter 3, pages 73–95. Morgan Kaufmann, 1990.
- [Eli75] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Trans. on Information Theory*, IT-21(2):194–203, March 1975.
- [HW73] M. Hollander and D. Wolfe. *Nonparametric Statistical Methods*. Wiley, New York, 1973.
- [Ped89] Edwin P.D. Pednault. Some experiments in applying inductive inference principles to surface reconstruction. In *Proc. of the Eleventh International Joint Conf. on Artificial Intelligence*, pages 1603–1609, 1989.
- [PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.
- [Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [Ris83] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.
- [Ris87] Jorma Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, 49(3):223–239, 1987.
- [SM92] Padhraic Smyth and Jeff Mellstrom. Detecting novel classes with applications to fault diagnosis. In Derek Sleeman and Peter Edwards, editors, *Proc. of the Ninth Intl. Conf. on Machine Learning*, pages 416–425, 1992.
- [WF87] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society, Series B*, 49(3):252–265, 1987.