

Bayesian Induction of Features in Temporal Domains ^{*}

Stefanos Manganaris [†]

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

March 31, 1995

Abstract

Most concept induction algorithms process concept instances described in terms of properties that remain constant over time. In temporal domains, instances are best described in terms of properties whose values vary with time. Data engineering is called upon in temporal domains to transform the raw data into an appropriate form for concept induction. I investigate a method for inducing features suitable for classifying finite, univariate, time series that are governed by unknown deterministic processes contaminated by noise. In a supervised setting, I induce piecewise polynomials of appropriate complexity to characterize the data in each class, using Bayesian model induction principles. In this study, I evaluate the proposed method empirically in a semi-deterministic domain: the waveform classification problem, originally presented in the CART book. I compared the classification accuracy of the proposed algorithm to the accuracy attained by C4.5 under various noise levels. Feature induction improved the classification accuracy in noisy situations, but degraded it when there was no noise. The results demonstrate the value of the proposed method in the presence of noise, and reveal a weakness shared by all classifiers using *generative* rather than *discriminative* models: sensitivity to model inaccuracies.

Keywords: Bayesian feature induction, deterministic time series, supervised concept learning, minimum message length, discriminative/generative classifiers.

^{*}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.

Introduction

Most concept induction algorithms in the literature process concept instances described in terms of properties that remain constant over time: each property assumes a time-invariant value. In temporal domains, instances are best described in terms of properties whose values vary with time; in a given time interval each property assumes a series of values. Raw temporal data have structure that is transparent to induction algorithms designed for instances with time-invariant properties. Data engineering is called upon in temporal domains to transform the raw data into an appropriate form for concept learning.

A supervised learning algorithm seeks to induce a good classification procedure, in terms of classification accuracy and computational efficiency, from a set of preclassified training data. This procedure maps instances to classes, effectively compressing all the information in an instance to a scalar value: the class. Temporal data are engineered to assist the learning algorithm and/or the classification procedure in their task; this process should utilize knowledge about the temporal structure of the data, which is foreign to the learner. Data engineering may compress and eliminate completely the temporal dimension of the data, for example, by replacing time series with some appropriate global description of them (e.g., the mean and variance). Alternatively, it may just enhance the data, leaving it up to the learner to compress the temporal dimension along with all the other information in the instance. An example of this type of processing is noise filtering.

The appropriate form of data engineering depends on the type of temporal domain. In this paper, I investigate a method for inducing features suitable for classifying finite, univariate, time series that are governed by unknown deterministic processes contaminated by noise. In other words, I only consider a single property over a finite interval of time. Moreover, I assume that this property evolves over time in a deterministic manner: the state of the property at one time determines its state at any later time. When instances are observed, the true state of the property is usually hidden by added noise. Instances are to be classified according to the underlying, hidden, process; note, however, that different processes may be associated with the same class—this is analogous to disjunctive concept learning. When the number of different processes associated with a class is infinite, the domain is termed semi-deterministic.

In a supervised setting, I induce piecewise polynomials of appropriate complexity to characterize the data in each class, using Bayesian model induction and the minimum message length principle. The induced models are then used by a simple Bayesian classifier. Piecewise polynomials provide a flexible set of features that can

approximate the characteristics of any class in the studied domain. Moreover, their use introduces a form of inductive bias that takes into account the temporal structure of the data to improve classification performance.

In this study, I evaluate the proposed method empirically in a semi-deterministic domain: the waveform classification problem, originally presented in the CART book. I compared the classification accuracy of the proposed algorithm to the accuracy attained by C4.5 under various noise levels. The induction of piecewise polynomials improved the classification accuracy in noisy situations, but degraded it when there was no noise. The results demonstrate the value of the proposed method in the presence of noise, and reveal a weakness shared by all classifiers using *generative* rather than *discriminative* models: sensitivity to model inaccuracies.

Bayesian Induction of Features

I propose a simple algorithm for inducing piecewise polynomial models for the processes characterizing the classes. 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 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*.

The 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, balancing, in effect, the complexity of the model and its ability to explain the data.

To assign priors, I use the minimum message length principle [Che90, Ris87, WF87]. The negative logarithm with base two of the prior probability of a model is computed as the length in bits of its shortest possible encoding.

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. 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. Continuity is not enforced across intervals. As for k -models, “goodness” is judged by the posterior probability. This part of the algorithm is adopted from an algorithm presented by Pednault in [Ped89] for surface reconstruction in computer vision.

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. 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 a vector D of n data points is then

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

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 (1), using standard least-squares multiple regression; these coefficients maximize the model’s likelihood. Equation (1) is further maximized when the noise variance is set to $\sigma^2 = (1/n) \sum_{i=1}^n e_i^2$.

In the second step, the algorithm shifts its attention to the model’s posterior probability and considers the model’s shortest encoding. The coefficients of a polynomial can be represented with arbitrary precision. A very precise encoding 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 precision depending on the sampling rate of data points), the degree and coefficients of all polynomials (each with its computed precision), 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 . When there is no bound in the state of prior knowledge, I use a universal prior [Ris83, Eli75]. 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 models induced for the processes in the domain can be used by a simple Bayesian classifier for classifying new instances. The prior probability of a class can be estimated from its frequency in the training set. The likelihood of a class for an instance can be estimated as the maximum likelihood over the set of models associated with the class. The likelihood of a model, $P(D|M, I)$, can be computed by assuming that the noise e_t at time t follows a Gaussian distribution with mean zero and variance σ_t^2 (computed earlier), 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 distribution with N degrees of freedom (denoted χ_N^2) and

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

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.

The proposed algorithm has been implemented as part of an experimental system called CALCHAS.

Empirical Evaluation

The waveform recognition problem was introduced by Breiman *et al.* in [BFOS84]. In this domain there are three classes. Instances are generated by combining two of

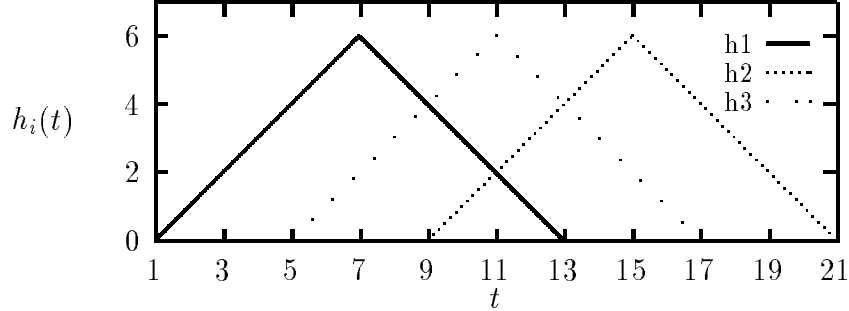


Figure 1: The three basic waveforms h_1 , h_2 , and h_3 used to generate waveform instances. Each instance is a random convex combination of two basic waveforms with added noise; the pair of basic waveforms determines its class.

three basic waveforms (Fig. 1). The pair of basic waveforms chosen determines the class of the instance: class 1 uses h_1 and h_2 ; class 2 uses h_1 and h_3 ; and class 3 uses h_2 and h_3 . An instance x in class i is generated as a linear combination of h_{i1} and h_{i2} :

$$x_i(t) = uh_{i1}(t) + (1 - u)h_{i2}(t) + e(t),$$

where u is a scaling uniform random deviate between zero and one, and $e(t)$ is a Gaussian noise process with mean zero and variance σ^2 . All instances are described by 21 samples at time points 1 to 21 (Fig. 2). Note, the waveform domain is semi-deterministic: there is an infinite number of deterministic processes associated with each class (indexed by the scaling random variable u). The original waveform data (with noise $\sigma^2 = 1$) are available at the UCI repository of machine learning databases [MA94].

I used the waveform domain to evaluate empirically the performance of CALCHAS. With the percentage of correctly classified instances as the dependent measure of performance, I compared CALCHAS to C4.5 under a range of noise levels. C4.5 was directly applied to the data, by treating each sample point as a feature. I chose C4.5 rather than CART, which was used in [BFOS84], because it was readily available; both C4.5 and CART are decision tree inducers.

Both systems were trained on 300 instances, created at random with equal priors for each class. The accuracy of each system was estimated using a disjoint set of 5,000 instances, created similarly. For this test data, and under noise with variance equal to one, the optimal Bayes rate was given in [BFOS84] to be 86%.

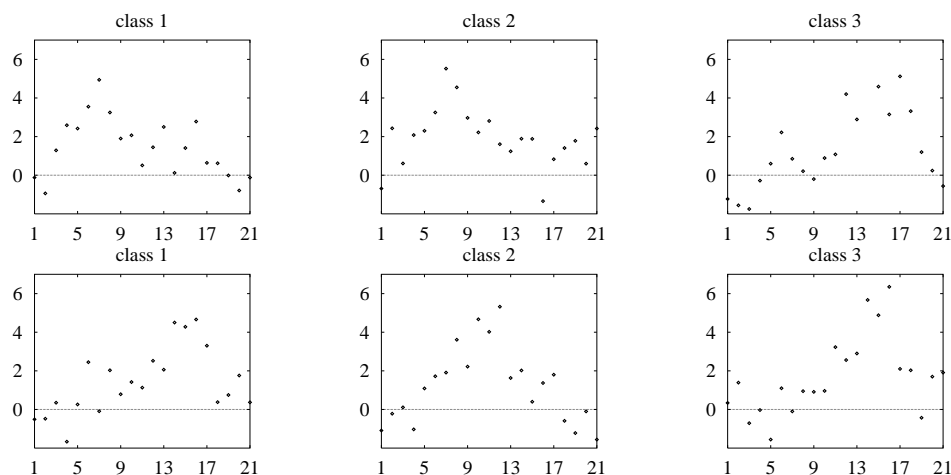


Figure 2: Typical waveform instances observed under noise ($\sigma^2 = 1$).

I ran each system with data under noise with variance 0, 0.5, 1, 2, 4, and 8, and estimated the mean classification accuracies and standard error of the means in five runs (Fig. 3). The statistical significance of the results was evaluated by the 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.

On the original waveform data (noise $\sigma^2 = 1$), C4.5 performed similarly to the 72% accuracy reported for CART in [BFOS84]; CALCHAS improved this accuracy to 77% ($p = 0.01$).

Discussion

It has been shown elsewhere that in domains where instances are governed by a few deterministic processes, piecewise polynomial models perform better than decision trees [Man95]. The models provide a representational inductive bias that can improve the classification performance under noise. In the absence of noise, piecewise polynomials still perform well in such domains, because an accurate model can be induced for each process.

In semi-deterministic domains, like the one used in this paper, the models induced are likely to be inaccurate for many instances, even after a lot of training: they only model the most probable features of the class (e.g., Fig. 4), while the actual characteristics of each instance vary (in the waveform example, they depend on the scaling random variable u). Model inaccuracies degrade the classification performance

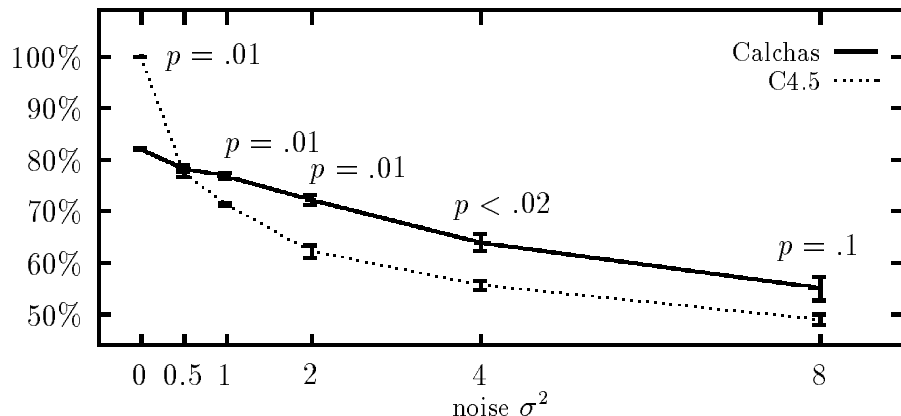


Figure 3: Classification accuracies under various noise levels for C4.5 and CALCHAS. Each point indicates the mean and standard error in five runs. The p -values were computed by the Wilcoxon rank sum test.

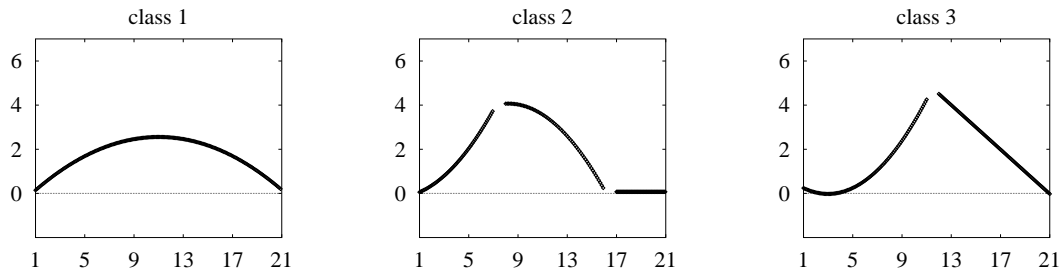


Figure 4: Typical models induced from the waveform data (noise $\sigma^2 = 1.0$).

and undermine the benefits of the stronger bias. In general, the stronger the inductive bias the greater the potential for inaccuracies. The exact circumstances under which the performance gain due to the models exceeds the losses due to inaccuracies depend on the domain. In the waveform problem, the comparison results with C4.5 indicate that the gains outweigh the losses when the noise variance exceeds 0.5.

Smyth and Mellstrom in [SM92] distinguish between classifiers that use generative and discriminative models. Generative models focus on the predictability of features given a class; they provide descriptions of how data can be generated for the

class. Discriminative models focus on the predictiveness of features; they pinpoint differences between classes. C4.5 and CART are examples of discriminative classifiers since they induce decision trees that discriminate classes using a few selected features. CALCHAS is a classifier based on generative models. The comparison of C4.5 and CALCHAS provides some insights regarding the advantages and disadvantages of discriminative and generative models in temporal domains.

In the absence of noise, C4.5 performed significantly better than CALCHAS. Discriminative classifiers have an advantage in noiseless semi-deterministic domains: they focus their attention on informative features, which discriminate classes, and effectively ignore all the variability within classes, which is responsible for inaccurate generative models. For example, in the noiseless waveform problem, the value at time 11 alone is very informative: a value less than two implies class 1; a value greater than two rules out class 1. Of course, finding informative features gets harder as the noise level increases, and this is when a stronger bias can help. One would certainly like to combine discriminative and generative models in a classifier and benefit from both. Generative models offer more than a stronger inductive bias. As Smyth and Mellstrom argue, generative models are better suited to on-line incremental learning, are considered to be cognitively more plausible, and can make a classifier robust (a robust classifier can detect instances from classes it has not been trained on.)

In conclusion, classification performance in noisy temporal domains governed by semi-deterministic processes can benefit from an inductive bias in the form of piecewise polynomial representations. In noiseless semi-deterministic domains, piecewise polynomial features degrade the classification performance and should thus not be used without additional guards, perhaps in the form of a discriminative model. Generative classifiers offer advantages, but are sensitive to inaccuracies in their models.

Acknowledgments

I thank Doug Fisher for supervising and supporting this work. Patrick Murphy and David Aha maintain the UCI repository of machine learning databases. This research has been supported in part by a grant from NASA Ames (NAG 2-834).

References

- [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. Shrager 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.
- [MA94] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. Machine-readable data repository. Irvine, CA: University of California, Department of Information and Computer Science., 1994.
- [Man95] Stefanos Manganaris. Learning to classify sensor data. Technical Report CS-95-10, Vanderbilt University, March 1995.
- [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.
- [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.