The Role of Discretization Parameters in Sequence Rule Evolution

Magnus Lie Hetland[1] and Pål Sætrom[2]

[1] Norwegian University of Science and Technology,
Dept. of Computer and Information Science,
Sem Sælands vei 9, NO–7491 Trondheim, Norway
magnus@hetland.org
[2] Interagon AS, Medisinsk-teknisk senter,
NO–7489 Trondheim, Norway
paalsat@interagon.com

Abstract. As raw data become available in ever-increasing amounts, there is a need for automated methods that extract comprehensible knowledge from the data. In our previous work we have applied evolutionary algorithms to the problem of mining predictive rules from time series. In this paper we investigate the effect of discretization on the predictive power of the evolved rules. We compare the effects of using simple model selection based on validation performance, majority vote ensembles, and naive Bayesian combination of classifiers.

1 Introduction

As raw data become available in ever-increasing amounts, there is a need for automated methods that extract comprehensible knowledge from the data. The process of knowledge discovery and data mining has been the subject of much research interest in later years, and one recent subfield is that of sequence mining. In our previous work [1] we have applied evolutionary algorithms to the problem of mining predictive rules from time series. Our method involves discretizing the time series, in order to be able to evaluate our rules on them (which, in turn, allows us to use genetic programming to find such rules). This process of discretization discards some information about the time series, and the parameters chosen (such as the width of the discretization window) will determine which features are available for the mining algorithm.

In this paper we investigate the effect of discretization on the predictive power of the evolved rules. We evolve rules for different discretization parameter settings and design predictors using the following methods:

1. We test rules evolved for different settings on a validation set. We then take the rule with the best performance to be our predictor.
2. We select the best rule for each of five discretization resolutions and combine them to form a predictor ensemble by simple majority vote.

3. We select the best rule for each of five discretization resolutions and combine them by means of a naive Bayesian model.

Majority vote ensembles are a simple way of combining several predictors to achieve increased predictive power. To make a prediction, all the component predictors make their predictions; a simple majority vote is used to decide which prediction “wins”.

The naive Bayesian predictor is a simple probabilistic model that assumes conditional independence among the predictor variables. Even though this assumption may be too strong in many cases, there is much empirical evidence suggesting that the method is quite robust.

2 Method

In this section we first describe the general approach, as developed in [1], as well as the extensions introduced here, that is, combining predictors for several resolutions. Finally, the discretization method is discussed.

2.1 The General Mining Approach

The basic method works by using an evolutionary algorithm (EA) to develop rules that optimize some score (or fitness) function, which describes what qualities we are looking for. Three components are of central importance: The rule format, the mechanism for rule evaluation, and the fitness function.

For the purposes of this discussion, we only consider one rule format: The rule consequent is some given event that may occur at given times, possibly extrinsic to the series itself, while the antecedent (the condition) is a pattern in a very expressive pattern language called IQL [2]. This language permits, among other things, such constructs as string matching with errors, regular expressions, shifts (latency), and Boolean combinations of expressions. In our EA system, these expressions are represented as syntax trees, in the standard manner.

In order to calculate the fitness of each rule, we need to know at which positions in the data the antecedent is true. To ascertain this, we use the antecedent as a query in an information retrieval system, the Pattern Matching Chip (PMC) [3], for which our pattern language was designed. Such a chip is able to process about 100 MB/s.

There are many possible fitness functions that measure the precision, degree of recall, or interestingness of a query or rule. For the relatively straightforward prediction task undertaken here, simple correlation is quite adequate. We use a supervised learning scheme, in which we mark the positions where want the rule to make a prediction, and then, for each rule, calculate the correlation between its behaviour and the desired responses. This can be calculated from the confusion matrix of the rule (true and false positives and negatives) [1].

It is worth mentioning that even though we focus mainly on predictive power here, one of the strengths of the method is the transparency of its rule format. Rather than a black-box predictor, the user receives a rule in a human-readable language. This can be an advantage in data mining contexts.

2.2 Selecting and Combining Predictors

We want to investigate the effect of the parameters of the discretization process, or, more specifically, of the resolution and the alphabet size, on the predictive power of the developed rules. The specific meaning of resolution and alphabet size is given in Sect. 2.3; informally, the resolution refers to the width of each (overlapping) discretized segment of the time series, while the alphabet size is simply the cardinality of the alphabet used in the resulting strings.

As mentioned in the introduction, we will compare three methods of exploiting the variability introduced by these parameters: model selection through cross-validation, majority vote ensembles, and naive Bayesian combination of predictors.

The rules that are developed by our system have their performance tested on a validation set (done through a form of k-fold cross-validation). In the simple case, a single resolution and alphabet size is used, and the rule that has the best performance on the validation set is selected and tested on a separate test set. Instead of this simple approach, we use several resolutions and alphabet sizes, run the simple process for each parameter setting, and choose the rule (and, implicitly, the parameter setting) that has the highest validation rating. This way we can discover the resolution and alphabet size that best suits a given time series, without having to arbitrarily set these beforehand. To demonstrate the effect of this procedure, we also select the rule (and parameter setting) that has the lowest validation score, as a baseline.

The next method is the majority vote ensemble. For this, we run the basic process for each parameter setting, and for each resolution we choose the rule (and, implicitly, the alphabet size) that has the best validation performance. We then combine these rules to form a single predictor, through voting. In other words, when we observe a time series (for example, the test set) we discretize it at all the resolutions, with the given alphabet sizes, (possibly incrementally) and run each rule in its own resolution. The prediction ("up" or "down") that is prevalent (simple majority) among the rules is taken to be the decision of the combined predictor. Ensembles are described in more detail in [4]. The basic idea behind them is that if each of a set of classifiers (or predictors) is correct with a probability $p \geq 0.5$ and the classifiers are generally not wrong at the same time (that is, they are somewhat independent) then a majority vote will increase the probability of a correct classification.[3] More sophisticated ensemble methods exist; see [4] for details.

The naive Bayesian predictor is a simple probabilistic model that assumes conditional independence among the predictor variables. Assume that we have a family $\{X_i\}$ of predictor variables, and one dependent (predicted) variable Y. Assume that $P(Y = y)$ is the *a priori* probability of a given value y being observed for Y, and $P(X_i = x_i \mid Y = y)$ is the conditional probability of observing x for X_i, given that $Y = y$. Then, the naive Bayesian model states

[3] Note that because of this requirement, rules with lower than 50% validation accuracy were excluded from the ensembles.

that our predicted class should be

$$\zeta(\boldsymbol{x}) = \arg\max_y P(Y = y) \prod_i P(X_i = x_i | Y = y) \ . \tag{1}$$

The independence assumption may in many cases be quite strong, but the naive Bayesian model can be surprisingly robust, and, as shown in [5], may even be optimal in cases where the independence assumption is violated.

2.3 Feature Extraction and Discretization

Many features may potentially be extracted from time series; for the task of prediction, we need to extract features in a way that lets us access the features of a sequence prefix. A natural approach is to divide the sequence into (overlapping) windows of width w, and to extract features from each of these. This discretization approach is described in the following.

Our basic discretization process is a simple one, used, for example, in [6]. It works as follows. A time series $\boldsymbol{x} = (x_i)_{i=1}^n$, where $x_i \in \mathbb{R}$, is discretized by the following steps:

1. Use a sliding window of width k to create a sequence $\boldsymbol{w}$ of m overlapping vectors. More formally: Create a sequence of windows $\boldsymbol{w} = (w_i)_{i=1}^m$, where $w_i = (x_j)_{j=l(i)}^{r(i)}$, such that $l(1) = 1$, $r(m) = n$, $r(i) - l(i) = k - 1$ for $1 \leq i \leq m$, and $l(i) = l(i-1) + 1$ for $1 < i \leq m$. The window width is also referred to as the *resolution*.
2. For some feature function $f : \mathbb{R}^k \to \mathbb{R}$, create a feature sequence $\boldsymbol{f} = (f(w_i))_{i=1}^m$.
3. Sort $\boldsymbol{f}$ and divide it into a segments of equal length. We refer to a as the *alphabet size*.
4. Use the limit elements in the sorted version of $\boldsymbol{f}$ to classify each element in $\boldsymbol{f}$, creating a new sequence $\boldsymbol{s}$ where s_i is the number of the interval where f_i is found.

After this discretization process, we have a sequence $\boldsymbol{s}$, where s_i is an integer such that $1 \leq s_i \leq a$, and each integer occurs the same number of times in $\boldsymbol{s}$. For convenience, we map these integers to lowercase letters (so that for $a = 3$ our alphabet is a...c). This discretization is performed on the training set, and the resulting limits are used to classify the features of the test set. (This means that there is no guarantee that each character is represented an equal number of times in the test set.) Many feature functions f are possible, such as average value or signal-to-noise ratio. Since we are interested in the upward and downward movement of the series, we have chosen to use the slope of the line fitted to the points in each window through linear regression.

3 Experiments

In our experiments we use six data sets, available from the UCR Time Series Data Mining Archive [7] (the first five) and the Federal Reserve Board [8] (the last data set):

Random walk. A standard random walk, where each value is equal to the previous one, plus a random number.

ECG. ECG measurements from several subjects, concatenated.

Earthquake. Earthquake-related seismic data.

Sunspots. Monthly mean sunspot numbers from 1749 until 1990. The series is periodic, with one period of eleven years, and one of twenty-seven days.

Network. Packet round trip time delay for packets sent from UCR to CMU.

Exchange rates. Daily exchange rate of Pound Sterling to US Dollars.

For each of the data sets, the target prediction was when the series moved upward, that is, when the next value was greater than the current. It is to be expected that good predictors can be found for the ECG data (as it is quite regular and periodic), whereas the random walk data, and, to some extent, the exchange rate data, function as baselines for comparison. Finding predictors for random data would clearly indicate that our experiments were flawed, and finding good predictors for exchange rate data would also be surprising, as this is considered a quite difficult (if at all possible) task.

For each of the data sets we performed experiments with alphabet sizes 2 (the minimum non-trivial case), 5, 10, and 20, as well as window sizes (resolutions) 2, 4, 8, 16, and 32. This was done with tenfold cross-validation[4] and early stopping; that is, rules were selected based on their performance on a separate validation set before they were tested. As described in Sect. 2.2, results were used to select the alphabet size that gave the best single-predictor performance for each resolution, and these rules were then used in constructing ensembles and Bayesian classifiers.

The rules in the ensembles were developed individually, that is, only their individual performance was used in calculating their fitness. The performance of the ensemble was then calculated by performing a simple majority vote among the rules for each position. The results are summarized in Fig. 1. The percentages (predictive accuracy) are averaged over the ten folds. For the ECG, sunspot, network, and earthquake data sets, the difference between the worst single classifier and the best single classifier is statistically significant ($p < 0.01$ with Fisher's exact test), while the differences between the best single classifier, the ensemble, and the Bayesian classifier are not statistically significant ($p > 0.05$ for all except the difference between the best single classifier and the Bayesian combination for the Network data, where $p = 0.049$). For the random and exchange rate data sets there are no significant differences, as expected.

Figure 2 shows how rule accuracy is related to window and alphabet size. For the random and exchange rate data, no clear trend is discernible. Although there are clearly problem specific differences, for the ECG, sunspot, and network data, there seems to be a rough inverse relationship between window size and accuracy, regardless of alphabet size. For the earthquake data, a large alphabet makes up

[4] Note that full cross-validation was not used, due to the temporal nature of the data. The validation was constrained so that the elements of the training data occurred at an earlier time than the elements of the testing set.

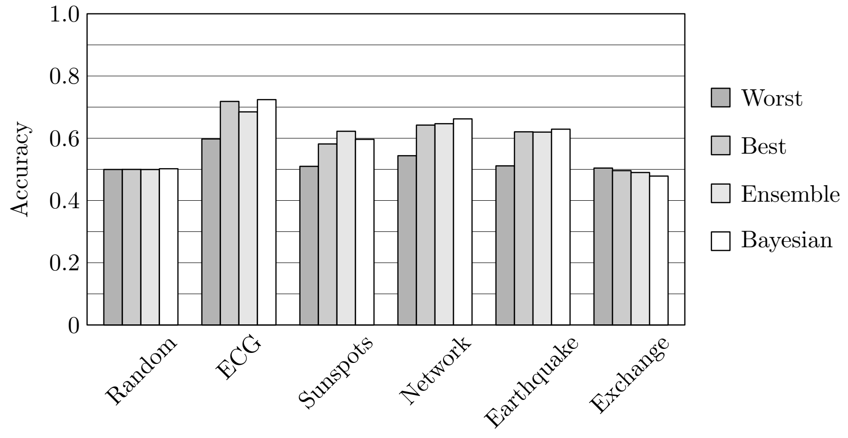

Fig. 1. Performance comparison

for poor resolution, giving peak performance for the two largest window sizes and the most fine-grained alphabet.

4 Discussion

In this paper we have examined the role of discretization when evolving time series predictor rules. We used three main techniques to improve the basic evolution presented in [1]: Model selection based on validation performance, majority vote ensembles, and naive Bayesian classifiers. Prior to our empirical study, we expected the ensembles to outperform the simple selection, and the Bayesian classifiers to outperform both of the other methods. As it turns out, on our data, there was no statistically significant difference between the three methods, even though the difference between the result they produced and that produced by unfavorable parameter settings (discretization resolution and alphabet size) was highly significant. This leads to the conclusion that, given its simplicity, plain model selection may well be the preferred method. Our experiments also showed that the relationship between discretization resolution, alphabet size, and prediction accuracy is highly problem dependent, which means that no discretization parameters can be found that work equally well in all cases.

References

1. Hetland, M.L., Sætrom, P.: Temporal rule discovery using genetic programming and specialized hardware. In: Proc. of the 4th Int. Conf. on Recent Advances in Soft Computing. (2002)
2. Interagon AS: The Interagon query language : a reference guide. `http://www.interagon.com/pub/whitepapers/IQL.reference-latest.pdf` (2002)

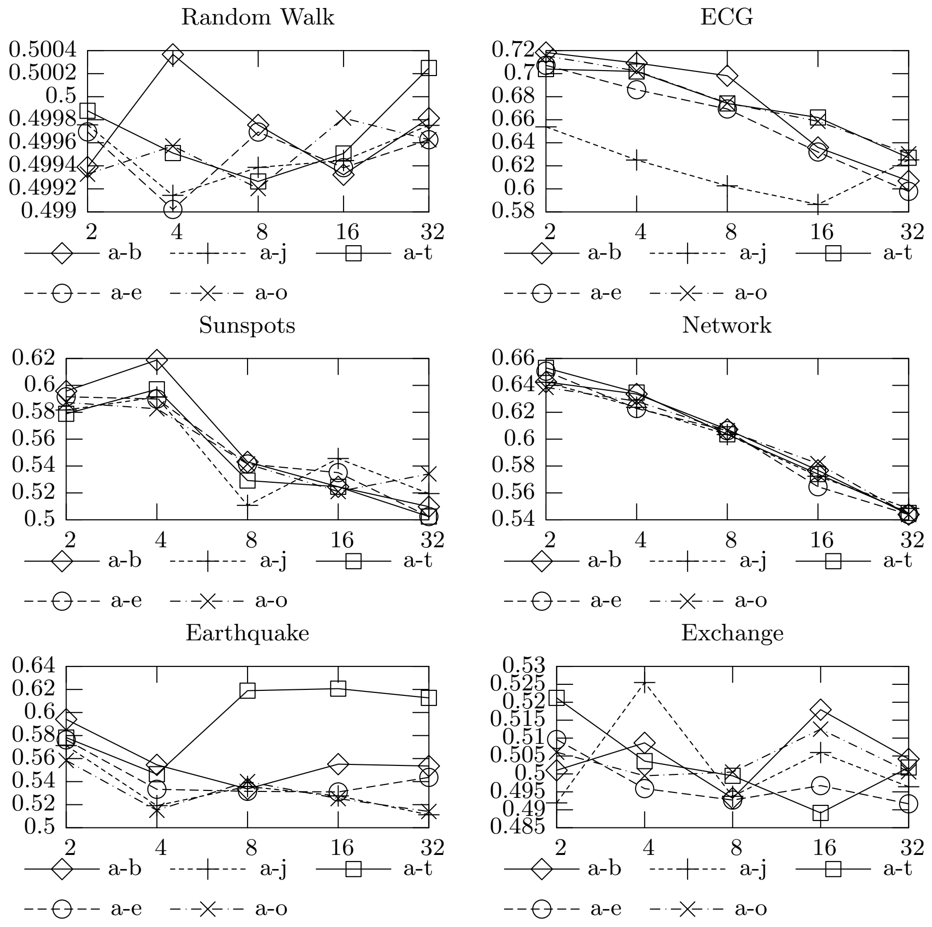

Fig. 2. Accuracy as function of window size and alphabet

3. Interagon AS: Digital processing device. PCT/NO99/00308 (2000)
4. Dietterich, T.G.: Ensemble methods in machine learning. Lecture Notes in Computer Science **1857** (2000) 1–15
5. Domingos, P., Pazzani, M.J.: On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning **29** (1997) 103–130
6. Keogh, E., Lonardi, S., Chiu, W.: Finding surprising patterns in a time series database in linear time and space. In: Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. (2002) 550–556
7. Keogh, E., Folias, T.: The UCR time series data mining archive. `http://www.cs.ucr.edu/~eamonn/TSDMA` (2002)
8. Release, F.R.S.: Foreign exchange rates 1971–2002. `http://www.federalreserve.gov/Releases/H10/Hist` (2002)