PERUSE: An Unsupervised Algorithm for Finding Recurring Patterns in Time Series

Tim Oates OATES@CS.UMBC.EDU
Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD 21250

Abstract

This paper describes PERUSE, an unsupervised algorithm for finding recurring patterns in time series. It was initially developed and tested with sensor data from a mobile robot, i.e. noisy, real-valued, multivariate time series with variable intervals between observations. The pattern discovery problem is decomposed into two subproblems: (1) a supervised learning problem in which a teacher provides exemplars of patterns and labels time series according to whether they contain the patterns; (2) an unsupervised learning problem in which the time series are used to generate an approximation to the teacher. Experimental results show that PERUSE can discover patterns in audio data corresponding to recurring words in natural language utterances and patterns in the sensor data of a mobile robot corresponding to qualitatively distinct outcomes of taking actions.

1. Introduction

How do children discover sound patterns corresponding to words in their native language? Adults perceive spoken utterances as containing discrete words, sometimes leading to the false impression that word boundaries are somehow marked in the acoustic signal, much as word boundaries in written text are marked with spaces. It becomes clear that this is not the case when listening to someone speak in an unfamiliar language. It is usually impossible to tell where one word leaves off and another begins. The PERUSE algorithm (**P**attern **E**xtraction from **R**eal-valued sequences **US**ing **E**xpectation maximization) was designed to address this problem, and thus to allow artificial agents, such as mobile robots, to discover words in spoken utterances.

Consider the difficulties posed by the word discovery problem. The learner does not know how many distinct sound patterns (i.e. words) occur in the utterances, nor does it know whether or where a given pattern occurs in a given utterance. Different patterns have different temporal extents, e.g. the word "red" takes less time on average to utter than the word "conflagration", and different occurrences of the same pattern are never exactly the same, i.e. the audio waveforms corresponding to utterances of the same word by the same speaker are never identical. Some patterns are wholly contained in other patterns, e.g. "red" and "redolent". Finally, the learning problem is unsupervised. There is no teacher providing information beyond that contained in the utterances themselves.

PERUSE was developed as part of a larger program of research with the goal of building representations and algorithms that will make it possible for a robot to learn language given qualitatively the same inputs available to children, i.e. utterances and sensory information about the contexts in which the utterances are generated (Oates, 2001). However, PERUSE is a general-purpose algorithm for finding recurring patterns in real-valued, multivariate time series with variable intervals between observations. The word discovery problem described above simply exhibits all of the difficulties that PERUSE is designed to handle.

The pattern discovery problem addressed by PERUSE can be conceptually decomposed into two subproblems. The first is a supervised learning problem in which a teacher provides exemplars of the patterns to be identified and labels time series according to whether they contain the patterns. Our solution to this problem is based on the Expectation Maximization (EM) algorithm and results in a representation of each pattern that can be used to determine the probability that a new time series contains a pattern and to localize the occurrence in time. The second subproblem is an unsupervised learning problem in which the time series are used to generate an approximation to the teacher. By composing the solutions to these two subproblems we obtain an unsupervised learning algorithm.

2. Problem Description

We assume that a complete time series is generated by repeatedly selecting a pattern from a set of patterns according to some distribution and generating an exemplar. Some patterns may be chosen more frequently than others depending on this distribution. The goal of PERUSE is to discover the patterns used most frequently to generate segments of the time series data that it receives as input. This section makes precise the terms *time series* and *pattern*, and the next two sections describe how PERUSE identifies the latter in the former.

Let an *observation* be a pair of the form $(\mathbf{x}, t)$, where $\mathbf{x} \in \Re^n$ and $t \in \Re$. That is, $\mathbf{x}$ is a vector of length n whose elements are real numbers, and t is a real number that represents the time at which $\mathbf{x}$ was recorded. The vector $\mathbf{x}$ might contain the values produced by a set of n sensors attached to an ICU patient or the closing prices of a basket of n stocks.

A time series of length l is a set of l time-ordered observations:

$$\mathcal{S} = \{(\mathbf{x}_i, t_i) | 1 \leq i \leq l\}$$

The fact that the observations are time-ordered means that $t_i < t_j$ whenever $i < j$. Let $\mathbf{x}_i^k$ denote the value of the k^{th} element in the vector observed at time t_i. A multivariate time series contains the following n univariate constituent time series:

$$\mathcal{S} = \{\{\{x_i^k, t_i\} | 1 \leq i \leq l\} | 1 \leq k \leq n\}$$

We define patterns in terms of a representation and a metric. The representation defines a space of possible patterns, and the metric makes it possible to identify "good" patterns in the data. Let a *pattern element* be a pair of the form $(\hat{\mathbf{x}}, \Delta t)$, where $\hat{\mathbf{x}} \in \{\Re^2 \cup \texttt{nil}\}^n$ and $\Delta t \in \Re^2$. Patterns occur in multivariate time series containing n constituent time series, so both observation vectors and pattern element vectors must contain n elements. A pattern of length l is represented as a totally-ordered set of l pattern elements:

$$\mathcal{P} = \{(\hat{\mathbf{x}}_j, \Delta t_j) | 1 \leq j \leq l\}$$

The above is a precise specification of pattern syntax. We now turn our attention to pattern semantics, i.e. the roles of $\hat{\mathbf{x}}$ and Δt.

Let $\hat{\mathbf{x}}^k$ be the k^{th} element of pattern element vector $\hat{\mathbf{x}}$. When $\hat{\mathbf{x}}^k = \texttt{nil}$, the pattern element says nothing about the value in the k^{th} constituent time series. When $\hat{\mathbf{x}}^k \in \Re^2$, its value specifies the mean and standard deviation of a normal distribution to which the value in the k^{th} constituent time series can be compared, i.e. $\hat{\mathbf{x}}^k = (\mu^k, \sigma^k)$. Let $\mu(\hat{\mathbf{x}}^k)$ and $\sigma(\hat{\mathbf{x}}^k)$ be the mean and standard deviation associated with $\hat{\mathbf{x}}^k$, respectively. Let $p(x|\mu, \sigma)$ be the value of the normal probability density function with parameters μ and σ at x. Given an observation vector, $\mathbf{x}$, and a pattern element vector, $\hat{\mathbf{x}}$, it is possible to compute the probability of $\mathbf{x}$ given $\hat{\mathbf{x}}$ as follows:

$$p(\mathbf{x}|\hat{\mathbf{x}}) = \prod_{k=1}^{n} \left\{ \begin{array}{ll} 1 & \text{if } \mathbf{x}^k = \texttt{nil} \\ p(\mathbf{x}^k | \mu(\hat{\mathbf{x}}^k), \sigma(\hat{\mathbf{x}}^k)) & \text{otherwise} \end{array} \right\}$$

Given a time series, $\mathcal{S}$, and a pattern, $\mathcal{P}$, it is natural to ask whether and where $\mathcal{P}$ occurs in $\mathcal{S}$. Let γ be a mapping from pattern elements in $\mathcal{P}$ to observations in $\mathcal{S}$. That is, if $\gamma(i) = j$, then the i^{th} pattern element maps to the j^{th} observation. γ specifies where $\mathcal{P}$ might occur in $\mathcal{S}$.

Given a mapping, it is possible to compute the time interval between two consecutive observations *in the mapping*, i.e. $t_{\gamma(i+1)} - t_{\gamma(i)}$. The two real numbers that comprise Δt_i are the mean and standard deviation of a normal distribution to which this time interval can be compared. When γ maps onto an actual occurrence of $\mathcal{P}$ in $\mathcal{S}$, it is assumed that the time between two observations in $\mathcal{S}$ that correspond to two consecutive pattern elements in $\mathcal{P}$ is normally distributed (or at least that the normal is a good approximation to the true distribution).

It is now possible to compute the probability that the observations in $\mathcal{S}$ onto which $\mathcal{P}$ is mapped represent an occurrence of the pattern as follows:

$$\begin{aligned} p(\mathcal{S}|\mathcal{P}, \gamma) \quad = \quad & p(\mathbf{x}_{\gamma(l)}|\hat{\mathbf{x}}_l) \prod_{i=1}^{l-1} p(\mathbf{x}_{\gamma(i)}|\hat{\mathbf{x}}_i) * \\ & p(t_{\gamma(i+1)} - t_{\gamma(i)} | \mu(\Delta t_i), \sigma(\Delta t_i)) \quad (1) \end{aligned}$$

Equation 1 is simply the product of the probabilities of each observation vector given the corresponding pattern element vector and the probabilities of the time intervals between consecutive observations in the mapping given Δt for the corresponding pattern element. The probability $p(\mathbf{x}_l|\hat{\mathbf{x}}_{\gamma(l)})$ is outside of the product because $\mathbf{x}_l$ is the last observation in the mapping and there is thus no corresponding Δt_l.

Note that PERUSE must search over two spaces to find candidates with high scores - the space of candidate patterns and, for each candidate, the space of mappings from candidates onto time series (for the purpose of evaluating equation 1). The EM algorithm lies at the core of the search over candidate space, and dynamic programming is used to find optimal mappings (i.e. those that maximize equation 1) efficiently.

3. A Supervised Subproblem

Given a set of exemplars of a particular pattern, estimating the parameters of the pattern is trivial. Let $\mathcal{P}$ be a pattern and let $\mathcal{E} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$ be a set of m exemplars of $\mathcal{P}$. Each element of $\mathcal{E}$ contains a single pattern, i.e.$\mathcal{P}$. Let $\mathbf{x}_i(\mathcal{S})$ denote the i^{th} observation vector in time series $\mathcal{S}$, and let $\Delta t_i(\mathcal{S})$ be defined analogously. Let $\mu(\hat{\mathbf{x}}_i^k)$ denote the mean specified by the k^{th} element of the i^{th} vector in $\mathcal{P}$, and let $\sigma(\hat{\mathbf{x}}_i^k)$ be defined analogously. The maximum likelihood (ML) estimate of $\mu(\hat{\mathbf{x}}_i^k)$ is as follows:

$$\mu(\hat{\mathbf{x}}_i^k) = \frac{\sum_{\mathcal{S}\in\mathcal{E}} \mathbf{x}_i^k(\mathcal{S})}{m}$$

That is, the ML estimate of $\mu(\hat{\mathbf{x}}_i^k)$ is simply the mean of the values generated from distribution $\hat{\mathbf{x}}_i^k$. One such value appears in position $\mathbf{x}_i^k$ in each of the exemplars. Likewise, the ML estimate of $\sigma(\hat{\mathbf{x}}_i^k)$ is as follows:

$$\sigma(\hat{\mathbf{x}}_i^k) = \left(\frac{\sum_{\mathcal{S}\in\mathcal{E}}(\mathbf{x}_i^k(\mathcal{S}) - \mu(\hat{\mathbf{x}}_i^k))^2}{m}\right)^{1/2} \qquad (2)$$

The ML estimate of $\mu(\Delta t_i)$ is as follows:

$$\mu(\Delta t_i) = \frac{\sum_{\mathcal{S}\in\mathcal{E}}(t_{i+1} - t_i)}{m} \qquad (3)$$

That is, the ML estimate of $\mu(\Delta t_i)$ is simply the mean of the values generated from distribution Δt_i. One such value appears in each exemplar as the time interval between observation i and observation $i+1$. Finally, the ML estimate of $\sigma(\Delta t_i)$ is as follows:

$$\sigma(\Delta t_i) = \left(\frac{\sum_{\mathcal{S}\in\mathcal{E}}((t_{i+1} - t_i) - \mu(\Delta t_i))}{m}\right)^{1/2} \qquad (4)$$

Given a set of exemplars generated from $\mathcal{P}$, estimating the parameters of $\mathcal{P}$ is straightforward.

Unfortunately, we do not have access to a set of clearly delineated exemplars of any patterns. In this section we assume that we have access to a single such exemplar and a set of time series that are labeled according to whether or not they contain an exemplar as well. Note that the time series that do contain an exemplar will in general contain additional unrelated data, perhaps generated by some other pattern or patterns.

It should be clear that given a set of time series containing exemplars of a pattern and the mappings for each of the time series, we can extract the exemplars and compute the maximum likelihood estimates of the means and standard deviations of the pattern elements as described above. The resulting formula for $\mu(\hat{\mathbf{x}}_i^k)$, which applies the mapping function γ to translate positions in pattern elements to positions of exemplars in time series, is as follows:

$$\mu(\hat{\mathbf{x}}_i^k) = \frac{\sum_{\mathcal{S}\in\mathcal{E}} \mathbf{x}_{\gamma_{\mathcal{S}}(i)}^k(\mathcal{S})}{m}$$

The value generated by distribution $\hat{\mathbf{x}}_i^k$ is found at position $\mathbf{x}_{\gamma_{\mathcal{S}}(i)}^k$ in the time series. The formulas for $\sigma(\hat{\mathbf{x}}_i^k)$, $\mu(\Delta t_i)$, and $\sigma(\Delta t_i)$ can be obtained in an analogous manner from equations 2, 3, and 4, respectively.

This is a classic example of a hidden data problem. Given the mappings, it is easy to compute the maximum likelihood estimates of $\mu(\hat{\mathbf{x}}_i^k)$, $\sigma(\hat{\mathbf{x}}_i^k)$, $\mu(\Delta t_i)$, and $\sigma(\Delta t_i)$. However, we are not given access to the mappings. This difficulty is overcome with the EM algorithm.

Suppose that all of the mappings are of the form $\gamma_{\mathcal{S}}(i) = i + C_{\mathcal{S}}$ for some constant $C_{\mathcal{S}}$ that differs for each time series $\mathcal{S}$. Such mappings will be called *consecutive mappings* because consecutive pattern elements are mapped to consecutive observations in the time series. Let $z_{\mathcal{S},i}$ be an indicator variable defined for time series $\mathcal{S}$. We will use the simpler notation z_i when the identity of $\mathcal{S}$ is clear. By definition, $z_i = 1$ if an an occurrence of the pattern ends at position i in $\mathcal{S}$, otherwise $z_i = 0$. Under the assumption that all mappings are consecutive, knowledge of $z_{\mathcal{S},i}$ for all $\mathcal{S}$ and all i is sufficient to determine the maximum likelihood estimates of the pattern's parameters.

The E step involves computing the expected value of $z_{\mathcal{S},i}$ given the current estimate of $\mathcal{P}$. Recall that $z_{\mathcal{S},i} = 1$ if an exemplar of $\mathcal{P}$ ends at position i in time series $\mathcal{S}$ and it equals zero otherwise. Therefore, the expected value of $z_{\mathcal{S},i}$ is 0 times the probability that an exemplar of $\mathcal{P}$ does not end at position i in $\mathcal{S}$ plus 1 times the probability that an exemplar of $\mathcal{P}$ ends at position i in $\mathcal{S}$, which is simply the latter probability (see equation 1).

In the M step, new parameter estimates are computed given the expected values of the hidden data. The formula for computing $\mu(\hat{\mathbf{x}}_i^k)$, which will be explained in detail shortly, is given below:

$$\mu(\hat{\mathbf{x}}_i^k) = \frac{\sum_{\mathcal{S}\in\mathcal{E}} \sum_{j=|\mathcal{P}|}^{|\mathcal{S}|} (z_{\mathcal{S},j} * \mathbf{x}_{j-|\mathcal{P}|+1+i}^k(\mathcal{S}))}{\sum_{\mathcal{S}\in\mathcal{E}} \sum_{j=|\mathcal{P}|}^{|\mathcal{S}|} z_{\mathcal{S},j}} \qquad (5)$$

Consider the numerator in the above expression. The outer sum is just a sum over all of the time series. For each time series, $\mathcal{S}$, the inner sum is over all locations in $\mathcal{S}$ where a consecutive mapping of pattern $\mathcal{P}$ can end. Inside this double sum is a product of two quantities. The first is simply $z_{\mathcal{S},j}$, the probability that an exemplar of $\mathcal{P}$ ends at position j in $\mathcal{S}$. The second quantity is the value that would have been generated by $\mu(\hat{\mathbf{x}}_i^k)$ in that exemplar. If the exemplar ends at position j, then it begins at position $j - \texttt{length}(\mathcal{P}) + 1$, so $j - \texttt{length}(\mathcal{P}) + 1 + i$ is the index in $\mathcal{S}$ to which the i^{th} element of $\mathcal{P}$ is mapped. The denominator is the same except the double sum is just over the values of $z_{\mathcal{S},j}$. The formula for computing $\sigma(\hat{\mathbf{x}}_i^k)$, which is similar to equation 5, is given below:

$$\sqrt{\frac{\sum_{\mathcal{S}\in\mathcal{E}}\sum_{j=|\mathcal{P}|}^{|\mathcal{S}|}(z_{\mathcal{S},j} * (\mathbf{x}_{j-|\mathcal{P}|+1+i}^k(\mathcal{S}) - \mu(\hat{\mathbf{x}}_i^k))^2)}{\sum_{\mathcal{S}\in\mathcal{E}}\sum_{j=|\mathcal{P}|}^{|\mathcal{S}|} z_{\mathcal{S},j}}}$$

The formula for $\mu(\Delta t_i)$ is an analogous extension to equation 3, and the formula for $\sigma(\Delta t_i)$ is an analogous extension to equation 4.

This section has assumed that we are given an exemplar of a particular pattern to be discovered, as well as a set of time series labeled according to whether or not they contain exemplars of the pattern. To ensure that EM identifies the parameters of the target pattern, rather than some other common pattern, our initial guess as to the parameters of $\mathcal{P}$ is guided by the exemplar. In particular, the means of the pattern are initialized to be the values in the exemplar, i.e. $\mu(\hat{\mathbf{x}}_i^k) = \mathbf{x}_i^k$, where $\mathbf{x}_i^k$ is the k^{th} item in the i^{th} element of the exemplar. The standard deviations are all initialized to be a small constant value. The goal is to start EM in an area in parameter space that is close to the set of parameters that we are seeking.

There is one last issue to contend with. All of the discussion to this point has assumed consecutive mappings. In general, such mappings will be inadequate. Given a time series $\mathcal{S}$, a pattern $\mathcal{P}$, and the constraint that mapping γ must end at location i is $\mathcal{S}$, it is possible to use dynamic programming to identify the mapping that maximizes $p(\mathcal{S}|\mathcal{P},\gamma)$ in $O(|\mathcal{S}| * |\mathcal{P}|)$ time. When computing parameters in the M step, we use the mapping identified by dynamic programming that ends at a given position rather than the consecutive mapping that ends at that position.

4. Approximating the Teacher

Given an exemplar of pattern $\mathcal{P}$ and a set of time series labeled according to whether or not they contain exemplars, the preceding section described how the EM algorithm can be used to estimate the parameters of $\mathcal{P}$. This section describes how an approximation to this training information, i.e. exemplars and labels, can be obtained in an unsupervised manner from the data.

Recall that we assume a time series is generated by repeatedly selecting a pattern according to some distribution over patterns and then generating an exemplar of the chosen pattern. The time series is the concatenation of these exemplars. Time series generated in this manner have two useful properties. First, by definition, there are many exemplars in the data of frequently occurring patterns. We expect there to be one or more "good" exemplars of frequently occurring patterns. Second, pattern boundaries are characterized by a lack of predictability. Within a pattern there is regularity that can be used to predict future data. But when a pattern ends the next observation depends entirely on which pattern is chosen from the distribution over patterns.

PERUSE uses these observations to identify windows of data in the time series that fall within frequently occurring patterns. This is accomplished by passing a window that spans *window_size* time steps over each of the time series. *window_size* is a user-defined parameter that must be set appropriately for each problem domain. For each such window of data, a pattern $\mathcal{P}$ is created by initializing its parameters from the data inside the window as described in section 4. For each time series, $\mathcal{S}$, including the one from which the window of data was taken, dynamic programming is used to identify the mapping that yields the highest value of $p(\mathcal{S}|\mathcal{P},\gamma_{\mathcal{S}})$ for that time series. That is, dynamic programming is used to find the mapping that is most likely to represent an exemplar of $\mathcal{P}$ in each of the time series. This mapping will be called the *maximum probability mapping.*

Next, the *min_matches* time series whose maximum probability mappings yield the highest values of $p(\mathcal{S}|\mathcal{P},\gamma_{\mathcal{S}})$ are identified. *min_matches* is a user-defined parameter, and typical values are small, such as three or four. Because these time series contain the best matches to the data in the window from which $\mathcal{P}$ was initialized, they are assumed to contain exemplars of $\mathcal{P}$. The window of data from which $\mathcal{P}$ was initialized is considered an exemplar, and this exemplar and the *min_matches* time series just identified are passed to the algorithm described in section 3 to obtain an ML estimate of the parameters of the underlying pattern.

When the parameters of $\mathcal{P}$ have been estimated, dynamic programming is once again used to identify the maximum probability mapping for each of the time series used in the estimation process. The sum of the logarithms of the probabilities associated with these mappings is recorded. This value is plotted as a function of window position for each of the time series.

The time series and window location that yield the highest value are identified. The data within this window are assumed to fall completely within a frequently occurring pattern (i.e. one that occurs more than *min_matches* times) because several "high quality" matches to the pattern occur in the data.

Let $\mathcal{E}$ be the set of *min_matches* time series used to estimate the parameters of the pattern underlying the data in the window, and let $\mathcal{P}$ be the result of the estimation process. Because the window spans a fixed time interval, the next step is to identify the true temporal extent of the underlying pattern. This is accomplished by extending the window used to initialize $\mathcal{P}$ by *grow_size* units of time. This new window of data and the time series in $\mathcal{E}$ are then handed to the algorithm described in section 4, resulting in a new pattern $\mathcal{P}'$.

Suppose $\mathcal{P}$ was initialized from time series data spanning time t_1 to time t_2. That means that $\mathcal{P}'$ spans t_1 to $t_2 + grow\_size$. If the quality of the maximum probability mappings for $\mathcal{P}'$ are higher for the portion of the pattern initialized from the data spanning time t_1 to t_2 are significantly higher than for the portion initialized from the data spanning time t_2 to $t_2 + grow\_size$, this is taken as an indication that growing the window resulted in a pattern boundary being crossed. The window is repeatedly extended by adding *grow_size* time steps to t_2 until a boundary crossing is detected, and then the window is extended in the other direction by subtracting *grow_size* time steps from t_1 until a boundary crossing is detected. The final values of t_1 and t_2 are taken to be the true temporal extent of the pattern underlying the original window of data.

How is it possible to determine when the quality of one portion of a maximum probability mapping is significantly higher than the quality of another portion? Recall that when $\gamma_{\mathcal{S}}(i) = j$, it is the case that the i^{th} pattern element maps to the j^{th} observation in $\mathcal{S}$. That is, it is the case that $\hat{\mathbf{x}}_i$ maps to $\mathbf{x}_j$. The quality of this part of the total mapping is evaluated as follows (see equation 1):

$$q(\mathcal{P}, \mathcal{S}, i, j) = \log\left(p(\mathbf{x}_j|\hat{\mathbf{x}}_i)p(t_{j+1} - t_j|\Delta t_i)\right)$$

Let Q_{old} be the set of quality values associated with the portion of pattern $\mathcal{P}$ that was initialized from the data spanning time t_1 to t_2.

$$Q_{old} = \{q(\mathcal{P}, \mathcal{S}, i, \gamma_{\mathcal{S}}(i)) | \mathcal{S} \in \mathcal{E}, t_1 \leq t_i < t_2\}$$

Let Q_{new} be the set of quality values associated with the portion of pattern $\mathcal{P}$ that was initialized from the data spanning time t_2 to $t_2 + grow\_size$. To compare the quality of the matches due to these two portions of $\mathcal{P}$ we can compare the values in Q_{old} and Q_{new}. This comparison is made with a standard one-tailed t-test using a significance level of $\alpha_{temporal}$ which is specified by the user. If the result of the t-test is not significant at the specified level, then a pattern boundary has not been crossed and the bounds of the window are extended again.

The next step is to determine whether and where the pattern occurs in the other time series. This is accomplished as follows. Dynamic programming is used to determine the maximum probability mapping for each of the time series. The $min\_matches + 1$ time series with highest associated probabilities are identified. The pattern is then retrained on these time series. Then a large number of exemplars are generated from the pattern by sampling from the distributions in each $\hat{\mathbf{x}}$ and Δt and their probabilities given the pattern are calculated. The result is a distribution of such probabilities. Then the probability of the maximum likelihood mapping for each of the *min_matches* time series given the pattern is compared to this distribution. If any of these values is smaller than $100 * (1 - \alpha_{series})\%$ of the values in the distribution, then it is assumed that the corresponding data were not generated by the pattern. In this case, it is assumed that the original *min_matches* time series are the only ones that contain exemplars. Otherwise, the value of *min_matches* is increased by one and the procedure is repeated until either *min_matches* equals the total number of time series or the statistical test fails. The end result is the identification of which time series contain exemplars of the pattern and where they occur, as indicated by the maximum probability mappings.

5. Experiments

This section describes the results of two sets of experiments that show PERUSE can discover patterns in audio data corresponding to recurring words in natural language utterances and patterns in the sensor data of a mobile robot corresponding to qualitatively distinct outcomes of taking actions.

In the first experiment, four subjects were asked to create random configurations of styrofoam blocks of various sizes, shapes and colors. Fifty of these configurations were then shown to a native speaker of English, a native speaker of German, and a native speaker of Mandarin. They were asked to generate natural language utterances describing what they saw. The only restriction placed on the utterances was that they had to be truthful statements about the scenes. This, coupled with the limited number of colors, sizes and shapes exhibited by the blocks, ensured that some words would be used in multiple utterances. One of the configurations was described by the English speaker as follows: "The cone is on top of the rectangle and to the left of the red ball." This is typical of the kind of utterances that were recorded. The English speaker used 60 unique words in the 50 utterances and each utterance contained on average 11.06 words. These quantities are 89 and 11.46 for the German speaker, and 37 and 19.04 for the Mandarin speaker.

Utterances were recorded with a head-mounted, noise-canceling microphone at a sampling rate of 8000 Hz. The raw audio signals were pre-processed using the publicly available RASTA-PLP package to extract 20 coefficients every 80 samples (Hermansky et al., 1991).

All 50 utterances for a given language, appropriately pre-processed, were passed to the PERUSE algorithm to find recurring patterns, i.e. words. The algorithm requires the user to specify a number of parameters. These parameters were tuned on the English utterances and were used unchanged on the German and Mandarin utterances. The performance of the algorithm appeared to be robust with respect to the choice of parameters, with small changes in parameters leading to little or no effect on performance. The specific parameter values used were as follows: $window\_size = 2000$ samples, thereby spanning one-fourth of one second of speech; $min\_matches = 4$ time series; $grow\_size = 500$ samples and spans one-sixteenth of a second; $\alpha_{temporal} = 0.00001$; $\alpha_{series} = 0.00001$.

To understand the performance of PERUSE on this task, consider table 1. Of the 60 words used by the English speaker the table shows those words that occurred four or more times. Because PERUSE is designed to find frequently recurring patterns, it has no hope of finding words that occur only once or twice. Inflected variants of words are not shown. For the purpose of assessing the performance of PERUSE, the algorithm is considered to be successful if it discovers the base form of a word and identifies inflected variants as instances of the base form. For example, it is correct for the algorithm to say that the word "balls" represents an occurrence of the word "ball".

Those words identified by PERUSE are shown in bold. Intuitively, a word is said to be identified when a pattern is discovered that can be used with high accuracy both to determine whether an occurrence of the word occurs in an utterance and to localize such occurrences in time. More concretely, each pattern P discovered by PERUSE was used to determine whether and where it occurred in each of the input time series. An occurrence of pattern P is said to correspond to an occurrence of word W if the starting and ending points of the occurrence of P match the starting and ending points of the occurrence of W within a temporal window no greater than 5% of the total length of the occurrence of W. Given all of the locations in the time series where the pattern has been identified to occur, word W is said to be identified by pattern P if 90% or more of the occurrences of P correspond to occurrences of W and if no more than 10% of the occurrences of P do not correspond to occurrences of W. That is, the ratio of hits to misses must be at least 9-to-1 and the ratio of hits to false positives must be at least 9-to-1.

Table 1. The list of the words used frequently by the English speaker with those words that were discovered by PERUSE shown in bold.

A	**Green**	**Small**
Above	Is	**Square**
And	**Large**	**That's**
Balanced	**Little**	The
Ball	**Next to**	To
Below	Of	Top
Big	**On top of**	**Touching**
Blue	One	Two
Circle	**Rectangle**	**Yellow**
Cone	**Red**	

Several aspects of table 1 are noteworthy. First, PERUSE discovered more than 65% of the frequent words used by the English speaker. This is a remarkable feat given that the algorithm has access to a mere 50 utterances and that it does not make use of any knowledge about the English language. In particular, it has no knowledge of characteristics of human speech that indicate word boundaries and it has no knowledge of phonemes. Virtually all past work on word discovery with real utterances has assumed access to some knowledge of this type.

Second, both "next to" and "on top of" were discovered as single word units. On the face of it, this is an error. However, on close inspection of the transcripts of the English utterances, it is clear that the algorithm did exactly the right thing in these two cases. Every time the English speaker uttered the word "next" he said "next to", and every time he uttered the word "on" he said "on top of".

In virtually all cases, patterns discovered by PERUSE erred on the side of misses rather than false positives. That is, the patterns would more often fail to identify an occurrence of a word than say that a word occurred where it didn't. It is possible that raising the value of α_{series} can shift this balance in the other direction. Also, the patterns tended to match too much of the time series rather than too little. That is, the patterns would often match all of the data corresponding to an occurrence of the underlying word plus some amount of additional adjacent data.

The results for German and Mandarin were similar, with 72.9% of the German words that occurred four or more times being discovered and 78.4% of the mandarin words. Again, these results are impressive given the relatively small sample of training data and the total lack of any knowledge of language and linguistics on the part of the algorithm.

In the second experiment PERUSE was used to identify patterns in the sensor data of a mobile robot corresponding to qualitatively distinct outcomes of taking actions. The robot was given a controller parameterized by two real numbers that influenced how its movements would be affected by objects visible in the images returned by a CCD camera (Braitenberg, 1984). The robot explored its two-dimensional action space by stochastically selecting a pair of parameters for the controller and running it for a fixed amount of time. Prior to running the controller, a single object was placed in a random location in the robot's environment. Sensory information included the size of the object in pixels, the coordinates of its centroid in the visual field, its mean hue/saturation/intensity, the height and width of the bounding box around the object, and a wavelet-based representation of the shape of the object. Each of these values was recorded at a rate of approximately 10 Hz during invocations of the controller.

Data from a total of 50 invocations of the controller were gathered, and PERUSE was applied to the resulting time series in an effort to find patterns, i.e. possible effects of running the controller. Note that there were a total of 250 time series with one set of 50 time series for each of the five groups of sensors described above. PERUSE was applied to each of these five sets individually. The parameters used were as follows: $window\_size = 15$ samples, thereby spanning one and a half seconds; $min\_matches = 3$ time series; $grow\_size = 5$ samples, i.e. one half of a second; $\alpha_{temporal} = 0.001$; $\alpha_{series} = 0.001$.

Given the set of patterns discovered by PERUSE in the sensor data, each experience (i.e. controller invocation) was labeled according to the pattern that had the highest likelihood match on one of its constituent time series. This induced a partition of the robot's experiences. A human subject was shown video of the robot's experiences and asked to create their own partition. The similarity of these partitions was assessed as follows. Given two pairs of experiences, we can ask whether they were placed together or apart in the partition created by the human. We can also ask whether they were placed together or apart in the partition induced by PERUSE.

The human and PERUSE agreed on whether two experiences belonged together or apart 88.7% of the time. Accordance with respect to putting experiences together is 91.9% and accordance with respect to putting them apart is 88.0%. The subsequences matched by patterns were on average one fifth the size of the full time series. Clearly, PERUSE discovered those portions of the time series that were central to the judgments made by the human in determining whether two experiences are qualitatively alike or not.

6. Discussion

The line of work that most directly influenced the development of PERUSE comes from the bioinformatics literature. Hertz *et al.* developed a greedy algorithm for discovering a single, fixed-width pattern shared by each member of a set of DNA sequences (Hertz et al., 1990). Lawerence and Reilly built on this work to develop an algorithm that solves the same problem using EM (Lawrence et al., 1993). All of this work served as the foundation for Bailey and Elkan's MEME algorithm (Bailey & Elkan, 1995), which uses data from the sequences to serve as seed patterns and uses a heuristic modification to EM that allows the number of occurrences of a pattern to be different than the total number of sequences.

PERUSE differs from MEME in a number of important ways. First, MEME is restricted to working with univariate, discrete data in which there is really no notion of time. Nucleotides simply follow one another spatially. PERUSE works with continuous, multivariate time series in which time is represented explicitly. This leads to very different considerations, and significant additional complexity, in determining how to represent patterns and apply EM. For example, PERUSE allows pattern elements to map to the same time step (i.e. insertions) or to skip over time steps (i.e. deletions) when being matched to the data. MEME does not allow either insertions or deletions. The patters discovered by PERUSE can have varying widths, compared to the fixed-width patterns discovered by MEME. Finally, to discover multiple patterns, the user of MEME must specify a parameter that tells the algorithm roughly how many patterns there are. PERUSE continues searching for patterns as long as the data support their existence.

The word discovery problem has been studied extensively, but virtually all existing approaches assume that utterances are represented as text with the spaces removed (Brent, 1999; de Marcken, 1996; Elman, 1990; Harris, 1954). Those algorithms that work with audio data typically incorporate knowledge of the target language, such as the phonemes that it contains (Roy, 1999). PERUSE does not require such knowledge and therefore moving from one language to another is trivial.

There has been a tremendous amount of work devoted to discovering patterns in time series, but in almost every case assumptions are made that are not made by PERUSE, such as categorical data (Agrawal & Srikant, 1994; Zaki, 2001) or univariate time series (Weigend et al., 1995).

Future work will involve creating an incremental version of the algorithm and applying it to additional problem domains.

Acknowledgments

The author would like to thank Paul R. Cohen for his support, both financial and intellectual, of the work described in this paper, which is based on the author's Ph.D. research. This research was supported by DARPA/USASMDC (Defense Advanced Research Projects Agency/U.S. Army Space and Missile Defense Command) under contract number DASG60-99-C-0074. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DARPA/USASMDC or the U.S. Government.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases.*

Bailey, T. L., & Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, *21*, 51–83.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology.* The MIT Press.

Brent, M. R. (1999). An efficient and probabilistically sound algorithm for word discovery. *Machine Learning*, *34*, 71–105.

de Marcken, C. (1996). *Unsupervised language acquisition.* Doctoral dissertation, MIT.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*, 146–162.

Hermansky, H., Morgan, N., Bayya, A., & Kohn, P. (1991). *Rasta-plp speech analysis*Technical Report TR-91-069). International Computer Science Institute.

Hertz, G. Z., Hartzell, G. W., & Stormo, G. D. (1990). Identification of concensus patterns in unaligned DNA sequences known to be functionally related. *Computer Applications in Biosciences*, *6*, 81–92.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., & Wooton, J. C. (1993). Detecting subtle sequence signals: A gibbs sampling strategy for multiple alignment. *Science*, *262*, 208–214.

Oates, T. (2001). *Grounding knowledge in sensors: Unsupervised learning for language and planning.* Doctoral dissertation, The University of Massachusetts, Amherst.

Roy, D. (1999). *Learning words from sights and sounds: a computational model.* Doctoral dissertation, MIT.

Weigend, A. S., Mangeas, M., & Srivastava, A. N. (1995). Nonlinear gated experts for time series: discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, *6*, 373–399.

Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, *42*, 31–60.