Non-Parametric Time Series Classification

Scott Lenser and Manuela Veloso

Carnegie Mellon University

5000 Forbes Ave
Pittsburgh, PA
{slenser,mmv}@cs.cmu.edu

***Abstract*— We present an improved state-based prediction algorithm for time series. Given time series produced by a process composed of different underlying states, the algorithm predicts future time series values based on past time series values for each state. Unlike many algorithms, this algorithm predicts a multi-modal distribution over future values. This prediction forms the basis for labelling part of a time series with the underlying state that created it given some labelled example signals. The algorithm is robust to a wide variety of possible types of changes in signals including changes in mean, amplitude, amount of noise, and period. We show results demonstrating that the algorithm successfully segments signals from several robotic sensors generated while performing a variety of simple tasks.**

***Index Terms*— time series, probabilistic models, sensors, Markov models**

I. Introduction

Segmentation of time series into discrete classes is an important problem in many fields. We approach the problem from the field of robotics where time series generated by sensors are readily available. We are interested in using these signals to identify sudden changes in the robot's environment allowing the robot to respond intelligently. For this application, the signal segmentation must be performed in real time and on line, which requires algorithms that are amenable to on-line use. Usually a mathematical model of the process that generates the sensor signal is unavailable as are the number of possible states in this process. Therefore, we focus on techniques that require little a priori knowledge and few assumptions.

In previous work [1], [2], we developed a technique for segmenting a time series into different classes given labelled example time series. In [1], we showed that our algorithm can successfully segment signals from robotic sensors. In this work, we improve on our previous technique by replacing a windowed approach to signal classification with a recursive solution based on a simple HMM.

We have named our new algorithm for classifying time series the Probable Series Classifier (PSC). It is based on a time series prediction component which we will refer to as the Probable Series Predictor (PSP). Unlike many other methods, PSP predicts a *multi-model* probability density over next values. PSC uses several PSP modules to classify a time series into one of a set number of pre-trained states. PSC uses one PSP module per state. Each PSP module is pre-trained from an example time series generated by one state. PSP uses an internal non-parametric model trained from an example time series to make its predictions PSC runs each PSP module on a time series to be classified and uses the one which best predicts the time series as the classification of the unknown time series.

There has been much interest in time series analysis in the literature due to the broad applicability of time series techniques. There have also been many approaches to time series predictions, most of which are focused on producing a single predicted value. For example, time series prediction has been done using AR, ARMA, IMA, and ARIMA models (e.g. [3]) and neural networks (NNs). All of these techniques produce a single estimated next value in the time series. These techniques can be converted into predicting a distribution over values by assuming a Gaussian deviation around the predicted value. This approach is used by Petridis and Kehagias for NNs [4], [5] and Penny and Roberts for HMM-AR models [6]. A related approach is that of using Gaussian Processes [7] for prediction. Unfortunately, this technique requires the inversion of an nxn matrix which takes $O(n^3)$ time for n observations. In contrast, our approach takes $O(n\log(n))$ time in our current implementation. The chief advantages of our approach over these previous approaches are that we are capable of predicting a multi-modal distribution over values and that our method is amenable to on-line training as more data from a particular state becomes available. While there are many methods from HMM research that predict a multi-modal distribution over values, we are not aware of any that condition on the previous time series value at the same time.

There are a wide variety of algorithms based on change detection, particularly in the domain of fault detection and identification (FDI). These FDI algorithms (e.g. [8], [9]) are usually specialized for the case of two states, one for normal system operation and one for failure cases. Because data can only be collected about the normal state of the system, these algorithms are attempting to solve a different problem and are generally more specialized for this task. We take a very general approach where we detect a wide variety of types of changes to the signal which sets PSC

This research was sponsored by the United States Army under Grant No. DABT63-99-1-0013. The content of the information in this publication does not necessarily reflect the position or the policy of the Defense Advanced Research Projects Agency (DARPA), the US Army or the US Government, and no official endorsement should be inferred.

apart from these other techniques.

There has also been a lot of interest in HMMs and switching state-space models, e.g. [6], [10]. These techniques require an a priori knowledge of the underlying structure of the system or extensive off-line training. PSC requires no knowledge about the system structure, as we only require labelled time series examples. PSC also requires negligible training time since almost all of the work is done at query time.

II. Probable Series Classifier Algorithm

Consider a time series of values $\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_t$ created by a generator with k distinct states. At each point in time, one of the states is active and generates the next data value in the time series based upon the previous time series values. Also assume that the frequency of switching between states is relatively low, such that sequential values are likely to be from the same state. We are interested in using the time series of values to recover which state was active at each point in time using only example time series created by each state.

The belief state at time j for state i is the probability of it being active at time j:

$$\begin{aligned} & B(s_j = i) = P(s_j = i | \vec{x}_j, \vec{x}_{j-1}, \ldots, \vec{x}_0) \\ &= \frac{P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_0, s_j = i) P(s_j = i | \vec{x}_{j-1}, \ldots, \vec{x}_0)}{P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_0)} \end{aligned}$$

We are interested in finding the state i that maximizes this probability. Note that $P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_0)$ is just a normalizing constant and thus doesn't affect which $s = i$ has the maximum likelihood. Furthermore, we will make the m^{th}-order Markov assumption that values $> m$ time steps ago are negligible, given more current readings. This assumption simplifies $P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_0, s_j = i)$ to $P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_{j-m}, s_j = i)$.

$$\begin{aligned} & P(s_j = i | \vec{x}_{j-1}, \ldots, \vec{x}_0) \\ &= \sum_l P(s_j = i, s_{j-1} = l | \vec{x}_{j-1}, \ldots, \vec{x}_0) \\ &= \sum_l P(s_j = i | s_{j-1} = l, \vec{x}_{j-1} \ldots \vec{x}_0) \cdot \\ &\quad P(s_{j-1} = l | \vec{x}_{j-1} \ldots \vec{x}_0) \\ &= \sum_l P(s_j = i | s_{j-1} = l) B(s_{j-1} = l) \end{aligned}$$

Here we have assumed the current state is independent of old observations (before time j) given the previous state. These assumptions simplify the problem to finding the state i that maximizes the following equations providing a recursive solution:

$$\begin{aligned} & B(s_j = i) \\ &\propto P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_0, s_j = i) P(s_j = i | \vec{x}_{j-1}, \ldots, \vec{x}_0) \\ &\approx P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_{j-m}, s_j = i) P(s_j = i | \vec{x}_{j-1}, \ldots, \vec{x}_0) \\ &= P(\vec{x}_j | \vec{x}_{j-1} \ldots \vec{x}_{j-m}, s_j = i) \cdot \\ &\quad \sum_l P(s_j = i | s_{j-1} = l) B(s_{j-1} = l) \end{aligned}$$

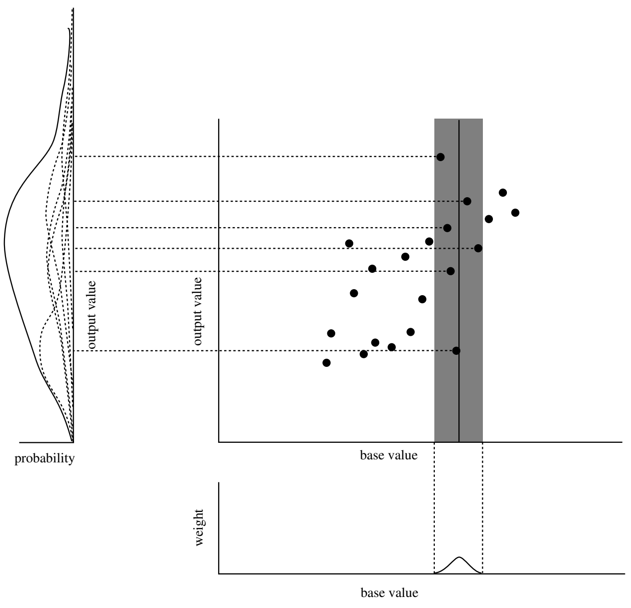

Fig. 1. Data prediction. The dots in the main graph show the data available for use in prediction. The grey bar shows the range of values used in the prediction. The bottom graph shows the weight assigned to each model point. The left graph shows the contribution of each point to the predicted probability of a value at time t as dotted curves. The final probability assigned to each possible value at time t is shown as a solid curve.

This belief update equation is useful for segmentation and classification. Our Probable Series Classifier algorithm uses the update equation for classification by finding the state that maximizes the probability of an unknown time series (using PSP for some key probability calculations). We assume a uniform distribution over the initial state of the generator. We also assume that $P(s_j = i | s_{j-1} = l) = .999$ if $i = l$ and a uniform distribution of the remaining probability over other states. The transition probability does not effect the most likely state at any given time much since the probability is dominated by the sensor readings. Our algorithm runs in real time on a Athlon XP 2700 processing data at 125Hz.

III. Probable Series Predictor Algorithm

We need a prediction of the likelihood of new time series values based upon previous values and the current state i.

$$P(\vec{x}_j | \vec{x}_{j-1}, \ldots, \vec{x}_{j-m}, s_j = i)$$

Note, that state i is known in this case, so we know what state we are predicting for. Assume we have previous time series values generated by this state. We can use these previous examples to generate an estimate at time j given the previous values of the time series. We will focus on the case where $m = 1$ and $\vec{x}$ is a single dimensional value.

We have: a set of value pairs $\vec{x}_i$,$\vec{x}_{i-1}$ and a value at time $j-1$ ($\vec{x}_{j-1}$). We need to generate a probability for each possible $\vec{x}_j$. We can use non-parametric techniques with a locally weighted approach. The problem is visualized in Fig. 1. We need to introduce some terminology to more easily discuss the problem.

base value(s) Those value(s) used in generating a predicted value. These are the time series values on which the output is conditioned. In the case of $m = 1$, this is just $\vec{x}_{j-1}$. The conditioning on the state is accomplished by having a separate model for each state.

output value The value output by prediction.

model points Points in base/output space in the training data for a state. These points form the model for this state. Each point is a pair of values: an output value $\vec{x}_j$ and associated base value(s) $\vec{x}_{j-1}, \dots, \vec{x}_{j-m}$.

prediction query A query of the model which provides $\vec{x}_{j-1}, \dots, \vec{x}_{j-m}$ as input and generates a probability density over $\vec{x}_j$ as output.

We will generate a probability density by generating a weighted set of output value predictions, one from each model point. A kernel is used that assigns more weight to model points with base value(s) near the query base value(s). The predicted output values must then be smoothed to form a continuous probability density.

We use a bandwidth limited kernel over base value(s) to weight model points for speed reasons. The kernel used is the tri-weight kernel:

$$K_t(x,h) = \begin{cases} (1-(x/h)^2)^3 & \text{if } |x/h| <= 1, \\ 0 & \text{otherwise} \end{cases}$$

This kernel is a close approximation to a Gaussian but is much cheaper to compute and reaches zero in a finite bandwidth. The finite bandwidth allows some points to be eliminated from further processing after this step. The bandwidth h is a smoothing parameter that must be selected that controls the amount of generalization performed. From non-parametric statistics, it is known that in order for the prediction to converge to the true function, as $n \to \infty$ (the number of model points), the following two properties must hold: $h \to 0$ and $nh \to \infty$. These properties ensure that each estimate uses more data from a narrower window as we gather more data. We use a ballooning bandwidth for our bandwidth selection. A ballooning bandwidth chooses the bandwidth as a function of the distance to the k^{th} nearest neighbor. Since the average distance between neighbors grows as $1/n$, we choose a bandwidth equal to the distance to the $\sqrt{n}$ nearest neighbor, ensuring that the bandwidth grows as $1/\sqrt{n}$ which satisfies the required statistical properties. Each model point is assigned a weight by the base kernel K_t which is used to scale its prediction in the next stage.

Fig. 1 illustrates the PSP algorithm. The dark circles represent model points that have already been seen. The x axis shows the base value. The y axis shows the output value. The dark vertical line shows the query base value. The grey bar shows the range of values that fall within the non-zero range of the base kernel. The graph underneath the main graph shows the weight assigned to each model point based on its distance from the query base value. A prediction is made based on each model point that is simply equal to its output value (we will refine this estimate later). The dotted lines leading from each model point used in the prediction shows these predicted output values. PSP is described in pseudo-code in Table I.

TABLE I
PROBABLE SERIES PREDICTOR ALGORITHM.

```
Procedure PredictOutput(generator_model,base_values)
    let OP ← generator_model.model_points
    let D ← dist(OP.base_values,base_values)
    Choose base_dist equal to the $\lceil\sqrt{n}\rceil$th smallest $d \in D$.
    let $h_b$ ← base_dist + noise_base
    let pred ← $\{z.output\_value \mid z \in OP \wedge$
            $dist(z.base\_values, base\_values) < h_b\}$
    Perform correlation correction on pred.
    let base ← $\{z.base\_values \mid z \in OP \wedge$
            $dist(z.base\_values, base\_values) < h_b\}$
    Choose $h_{i,o} \propto distance(\lceil\sqrt{n}\rceil^{\text{th}})$ nearest prediction.
    Return probability density equal to
        $pdf(z) = \sum_i K_g(pred_i - z, h_{i,o}) *$
            $K_t(base_i - base\_values, h_b)$
```

We need to smooth the predicted output values to get a continuous probability density. We will once again turn to non-parametric techniques and use a tri-weight kernel centered over each point. Because this kernel has a finite bandwidth, it may assign a zero probability to some points. This assignment is undesirable since we never have enough training data to be absolutely sure the data could not have occurred in this state. Hence, we assign a .0001 probability that the time series value is generated from a uniform distribution and a .9999 probability that it is generated according to the estimated distribution.

We need a method for selecting a bandwidth for K_g, the output kernel. We use a modified form of the ballooning method. For each output value prediction, we assign a bandwidth proportional to the distance to the $\sqrt{n}^{\text{th}}$ nearest output value with a minimum bandwidth. We used a proportionality constant of 0.5. We also experimented with selecting a bandwidth using the pseudo-likelihood cross validation measure [11], [12]. This alternative bandwidth selection had similar performance but took about 10 times as long to run.

As exemplified in Fig. 1, there is usually a strong correlation between the time series value at time t and the value at time $t-1$. This correlation causes a natural bias in predictions. Model points with base values below the query base value tend to predict an output value which is too low and model points with base values above the query base value tend to predict an output value which is too high. We can correct for this bias by compensating for the correlation between x_t and x_{t-1}. We calculate a standard least squares linear fit between x_{t-1} and x_t. Using the slope of this linear fit, we can remove the bias in the predicted output values by shifting each prediction in both base value and output value until the base value matches the query base value. This process can shift the predicted output value a substantial amount, particularly when using points far from the query base value. This process improves the accuracy of the algorithm slightly. This correlation removal was used in all the tests performed in this paper.

IV. Evaluation

We evaluated the Probable Series Classifier (PSC) using data logged by our robot as it performed various tasks. The data was hand classified as a baseline for comparison with the automatic classification. We used a standard Sony AIBO ERS-210 for gathering all of our data.

A. Methodology

We generated a set of data series from the sensors on our robot. We used two different sensors, a CMOS camera and an accelerometer. Since our PSC implementation currently only supports single dimensional data, we reduced each data series down to a single dimensional data series. Each camera image was reduced to an average luminance value (a measure of brightness), resulting in a 25Hz luminance signal. The accelerometer data is inherently three dimensional with accelerations along three axes. We chose to use the axis oriented towards the front of the robot (the other axes gave similar results). The accelerometer data has a frequency of 125Hz. For each task, PSC was trained on a segment of data for each possible class. PSC used a window of data to generate each classification starting at the data item to be classified and extending backwards in time, i.e. only data that would be available in an on-line scenario was used for classification. The PSC generated label was compared to a hand generated label to ascertain accuracy. In some of the signals, there were segments of the test signal that did not correspond to any of the trained classes. These segments were not used in calculating accuracy. We considered a total of five different classification tasks.

B. Results

Each figure show the results from one task. The bottom part of each figure shows the raw data signal used for testing. Each of the other figures corresponds to one of the trained classes. The class to which it corresponds is labelled to the left of each graph. The thick black line running through parts of each class graph indicates when this class is the correct class according to the human generated labelling. The small black dots show the probability that PSC assigned to this class at each point in time (based on a window of data prior to this time point). Ideally, the probability would be 1.0 when the thick black bar is present and 0.0 otherwise. In sections where the test data series does not correspond to any of the trained classes, the indicator bar is absent and the output of PSC for each class is irrelevant. Table II summarizes the results achieved by PSC. The column labeled “Windowed” shows the performance of our previous window based approach. The column labelled “PSC Accuracy” shows the accuracy of our new Markov model based approach. As the table shows, we see a significant reduction in errors by employing the Markov model.

Figure 2 shows the results from the first accelerometer task distinguishing between walking down a metal ramp, across a soft carpet, and into a low wooden wall. This task is labelled as “walk stability” in the results summary table. PSC was trained on one example sequence and tested on a completely separate sequence. Each class was trained using 500 examples, except 400 examples were used for training the “ramp” class. As the graphs show, PSC does an excellent job of distinguishing between these different walking conditions achieving and accuracy of 99.13%.

Figure 3 shows the results from the second accelerometer task distinguishing between walking in place on cement, hard carpet, and soft carpet.. This task is labelled as “walk floors” in the results summary table. PSC was trained on one example sequence and tested on a completely separate sequence. As the graphs show, PSC does an excellent job of distinguishing between these different walking conditions achieving and accuracy of 91.75% despite the similarity between the two types of carpet.

Figure 4 shows the results from the third accelerometer task distinguishing between playing soccer, walking into a wall, walking with one leg hooked on an obstacle, and standing still. The playing class includes a wide variety of signals including walks in several different directions and full body kicking motions such as diving on the ball. Small portions of the playing class, include standing in place making these sections look like the standing state. This task

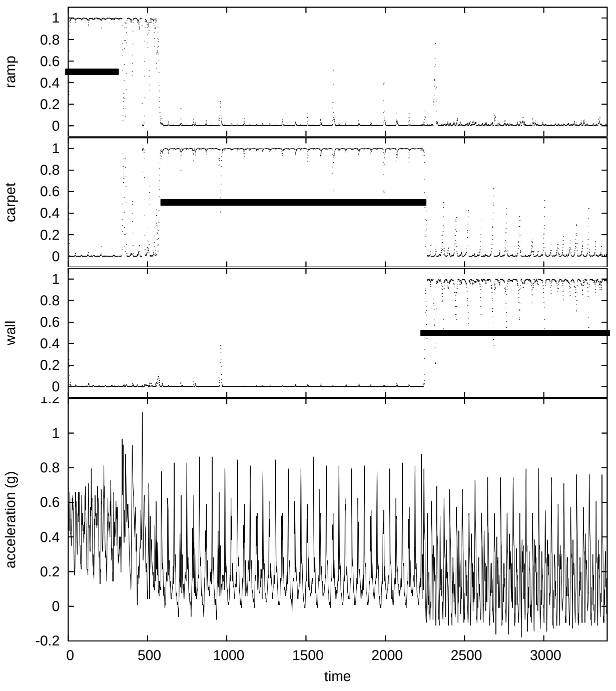

Fig. 2. Use of accelerometer data to distinguish between walking down a ramp, walking across a carpet, and walking into a wall.

TABLE II
Accuracy of PSC in various test classification tasks.

Task	Sensor	Windowed	PSC Accuracy
Walk stability	Accelerometer	99.19%	99.13%
Walk floors	Accelerometer	N/A	91.75%
Walk interference	Accelerometer	78.49%	82.92%
Lights playing	Camera	64.69%	74.02%
Lights standing	Camera	93.77%	99.11%

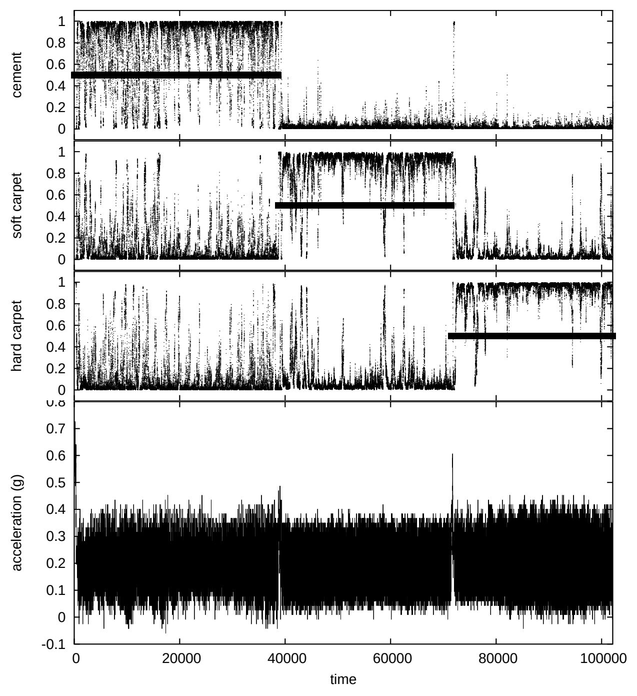

Fig. 3. Use of accelerometer data to distinguish between walking in place on cement, hard carpet, and soft carpet.

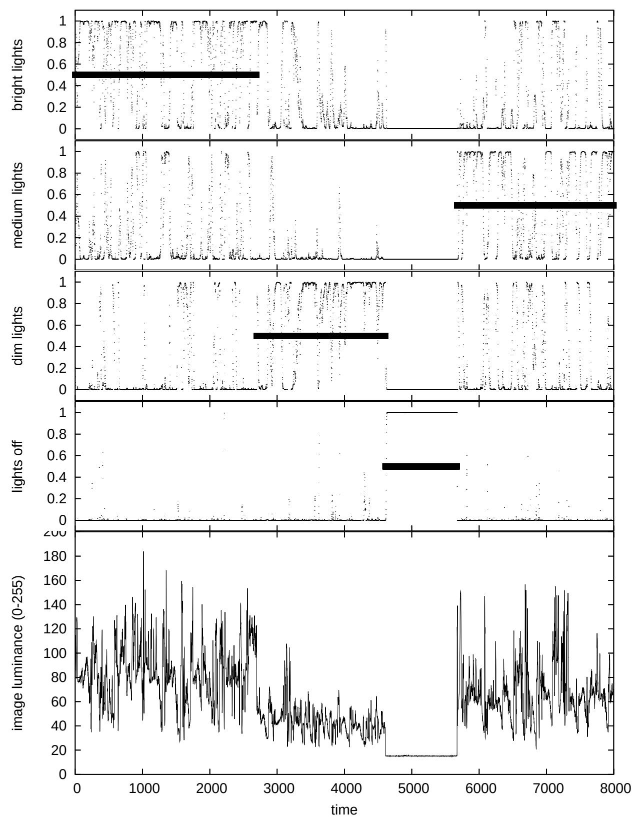

Fig. 5. Use of average luminance from images to distinguish between bright, medium, dim, and off lights while playing soccer.

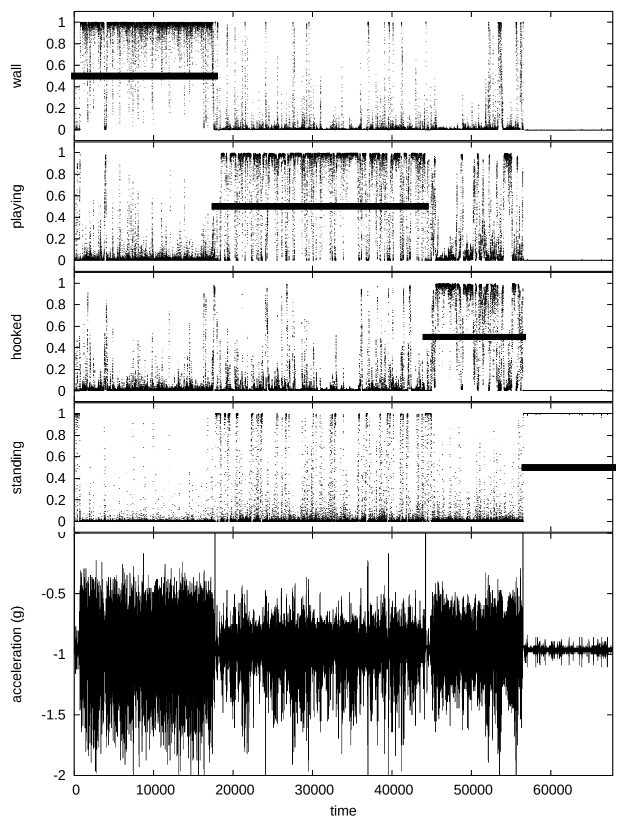

Fig. 4. Use of accelerometer data to distinguish between playing soccer, walking into a wall, walking with one leg caught on an obstacles, and standing still.

is labelled as "walk interference" in the results summary table. PSC was trained on example sequences from the test sequence. In other tests, we did not observe a noticeable difference between testing on training data and testing on separate testing data. Each class was trained using 5000 examples. PSC performed well overall, correctly classifying 82.92% of the data points. PSC performed perfectly on the standing still data. It had the most problems identifying hooked on an obstacle, often confusing it with playing.

Figure 5 shows the results from the first camera task distinguishing between bright, medium, dim, and off lights while the robot is playing soccer. This task is labelled as "lights playing" in the results summary table. PSC was trained on one example sequence and tested on a completely separate sequence. Each class was trained using 1000 examples. PSC performed fairly well overall, correctly classifying 74.02% of the data points. Most of the errors were due to problems distinguishing between bright lights and medium lights.

Figure 6 shows the results from the second camera task distinguishing between bright, medium, dim, and off lights while the robot is standing still. The robot moved its head to look at different objects. This task is labelled as "lights standing" in the results summary table. PSC was trained on one example sequence and tested on a completely separate sequence. Each class was trained using 150–200 examples. Although this is an easy classification task, it is important to test that the algorithm works on both easy and difficult

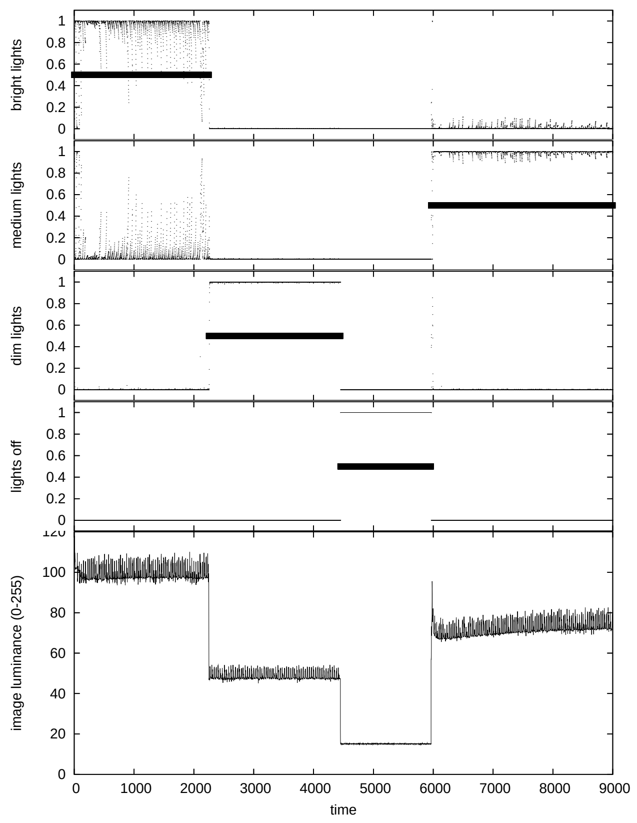

Fig. 6. Use of average luminance from images to distinguish between bright, medium, dim, and off lights while standing still.

tasks to ensure practicality of the algorithm. PSC passes easily achieving 99.11% accuracy.

V. Conclusion

We have presented an algorithm for generating predictions of future values of time series. We have shown how to use that algorithm as the basis for a classification algorithm for time series. We proved through testing that the resulting classification algorithm robustly classifies a wide variety of robotic sensor signals. We verified the performance of the algorithm for a variety of sensors and robot tasks. The algorithm runs in real-time and is amenable to on-line training.

References

[1] Scott Lenser and Manuela Veloso. Classification of robotic sensor streams using non-parametric statistics. In *it To appear: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 2004.

[2] Scott Lenser and Manuela Veloso. Automatic detection and response to environmental change. In *Proceedings of ICRA-2003*, 2003.

[3] Kan Deng, Andrew Moore, and Michael Nechyba. Learning to recognize time series: Combining arma models with memory-based learning. In *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, volume 1, pages 246–250, 1997.

[4] V. Petridis and A. Kehagias. Modular neural networks for MAP classification of time series and the partition algorithm. *IEEE Transactions on Neural Networks*, 7(1):73–86, 1996.

[5] V.Petridis and A.Kehagias. *Predictive modular neural networks: applications to time series*. Kluwer Academic Publishers, 1998.

[6] William Penny and Stephen Roberts. Dynamic models for nonstationary signal segmentation. *Computers and Biomedical Research*, 32(6):483–502, 1999.

[7] A. Girard, C. E. Rasmussen, J. Quionero-Candela, and R. Murray-Smith. Multiple-step ahead prediction for non linear dynamic systems — a gaussian process treatment with propagation of the uncertainty. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 529–536. MIT Press, 2003.

[8] Michèle Basseville and Igor Nikiforov. *Detection of Abrupt Change - Theory and Application*. Prentice–Hall, Englewood Cliffs, N.J., 1993.

[9] Masafumi Hashimoto, Hiroyuki Kawashima, Takashi Nakagami, and Fuminori Oba. Sensor fault detection and identification in dead-Reckoning system of mobile robot: Interacting multiple model approach. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2001)*, pages 1321–1326, 2001.

[10] Zoubin Ghahramani and Geoffrey E. Hinton. Switching state-space models. Technical report, 6 King's College Road, Toronto M5S 3H5, Canada, 1998.

[11] J. D. F. Habbema, J. Hermans, and K. van den Broek. A stepwise discrimination analysis program using density estimation. In *Proceedings of Computational Statistics (COMPSTAT 74)*, 1974.

[12] R. P. W. Duin. On the choice of smoothing parameters of Parzen estimators of probability density functions. In *Proceedings of IEEE Transactions on Computers*, volume 25, pages 1175–1179, 1976.