Active Learning on Non-Stationary Functions

Jeff Schneider
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Jeff.Schneider@cs.cmu.edu

Abstract

We consider the problem of active learning on a function that varies with time. A common approach is to employ a learning algorithm that reduces the contribution of data points as they become older. With this method an active learner will decide to resample regions of the space where all the data has become fairly old. We propose an alternative that calls for the learner to retain all data. As the function varies in time, it may stray away from a certain mapping, but may also return to it at some point in the future. Our algorithm reviews its historical data to see if the current mapping is similar to one held previously and re-uses the old data points when this occurs. Our results show that this can provide better approximation of time-varying functions and can free an active learner from having to resample regions of the space where historical data has become relevant again.

Motivation

Active learning is based on the assumption that experimentation is an expensive process, which is usually the case when learning is applied to real, physical systems. Physical systems often come with another property: the mapping from their inputs to their outputs varies through time. In this work, we consider active learning of a non-stationary function with memory-based local models. The usual way to approximate non-stationary functions is to implicitly or explicitly reduce the contribution of data points as they age. With memory-based learning the old data points are still available and we propose an algorithm to get extra information from them in order to improve the performance of an active learner. Rather than discarding old points via decaying weights on them, the algorithm identifies periods in history in which the function mapping was similar to the current one, and weights them heavily in order to improve the current model by providing better coverage of the input space.

There are several scenarios where this algorithm can improve an active learning system. One is when the learner is asked to provide an estimate of the entire current function in the face of it being non-stationary. The most common solution would involve the active learner weighting data less as it gets old. That would force it to consider experiments anywhere there is no current data. With an algorithm to evaluate and reuse relevant old data points it can consider experiments from a smaller space including only those regions where there is no current or relevant historical data.

We believe the gains from this algorithm will be greatest in a second scenario where it is used as part of a learning controller that must find and exploit optima in a non-stationary system. The active learner guides exploration until an optimum is found. Then the system spends most of its trials exploiting the optimum. When the function changes and that optimum disappears, active learning is again used to locate a new one. However, most of the recent data is centered around the old optimum and doesn't directly shed much light on the location of the new one. Based on the recent values near the old optimum, the proposed algorithm identifies likely locations for a new one and the active learner can begin its experimentation in those regions first.

A similar scenario occurs when the optimization takes place on a time-varying system that has some observable, but uncontrollable variables. Fluctuations in the uncontrollable variables can force the system into a region with no recent data support. Again, identification of useful historical data can greatly aid the active learner as it seeks the new optimum.

Description of the Algorithm

Our algorithm is based on the following assumptions:

1. The variation in the function over time is not unreasonably fast. As the average speed of variation in time increases the time spanned by the term *recent* decreases.
2. There are not enough *recent* data points to cover the whole region of interest in the input space. If there are enough, the usual methods of approximating non-stationary functions coupled with active learning will work fine.

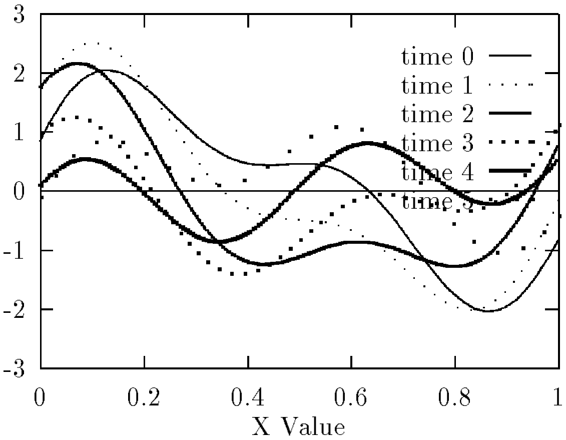

Figure 1: True function at several points in time

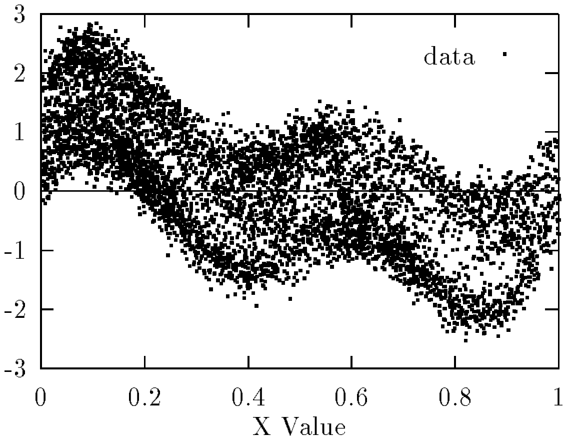

Figure 2: 8000 data points spanning time 0-20

3. Although the function mapping varies with time, it often returns to or passes near mappings it has had previously. If this is not true there is no information to be gleaned from old data points.

An outline of the algorithm to provide a single prediction and confidence interval is as follows:

1. Construct a *recent model* based only on the most recent data points. (The recent model represents part of the current mapping by assumption 1, but is not sufficient in itself by assumption 2.)
2. For every data point in memory, make a prediction of its output based on the recent model. If that data point is generated from the same underlying function that generated the recent model, its deviation from the predicted value (normalized by the recent model's confidence at that point) has a t distribution. Record this statistic for each point.
3. For every point in memory consider the sum of the squared t statistics of it and its temporal neighbors. (The neighbors' t statistics contribute to our confidence in the point by assumption 1.) Since the degrees of freedom for the t statistic is normally very high it can be approximated as a unit normal random variable, and the squared sum as a chi-square random variable. Comparing this value to the real chi-square distribution gives a confidence that the point is generated from the same underlying function as the recent model.
4. Perform a standard local weighted regression with the confidence values generated in step 3 contributing to the weight assigned to each data point. It is expected that some *non-recent* points will receive significant weighting by assumption 3.

Results

We tested the algorithm on the following one-dimensional, gaussian noise, time-varying function:

$$y = \sin(7x) + \sin(11x + 1) + \sin(5x + t) \qquad (1)$$

where y is the output variable, x is the input variable, and t is time. Fig. 1 shows the function at several different points in time. Note that the variation due to time cycles every 2π seconds. We generated 8000 training data points at equal intervals from time 0 to time 20. Gaussian noise with a standard deviation of 0.2 was added to the data. The sequence of x values for the data points was generated as a random walk between 0 and 1. This was done to mimic the situation where parts of the input space have not been visited recently. For the experiments run below we tested each model by asking it to predict the function at each of the x values: $[0.0, 0.1, ..., 1.0]$, at each of the times: [17, 18, 19, 20]. This gives a total of 44 test set points. For each prediction the model was given access only to the data points occurring before the time of the prediction.

If we are blind to the fact that the function varies in time, the entire set of data points appears as in fig. 2. It covers the entire region spanned by the fluctuation of the function in time. Using local linear regression to build a model for this data, the best RMS test set error we obtained was 0.72.

If we assume the function does vary through time, we can perform the local linear regression weighting recent data points more strongly. For example, fig. 3 shows the true function at time 19 and the data points occurring immediately prior to then. If we include time as a dimension in the local weighted regression (using a one-sided gaussian kernel since we're not allowed to look into the future), we can effectively discard all data points except the most recent. This is what many function approximators do either implicitly or explicitly. Using this method the best test set error we obtained was 0.63.

The obvious problem in the example at time 19 is that the recent data points do not cover the whole

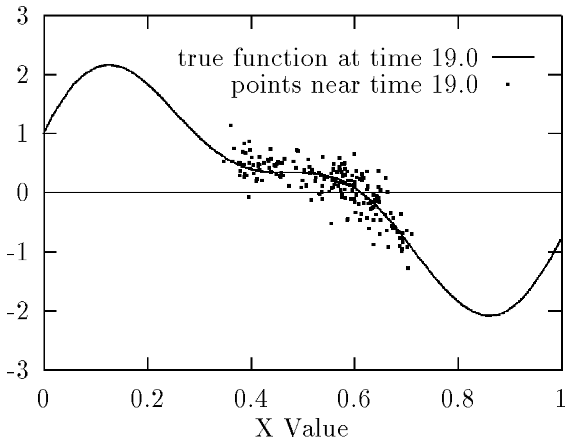

Figure 3: The function at time 19 and the data points from time [18.5:19.0]

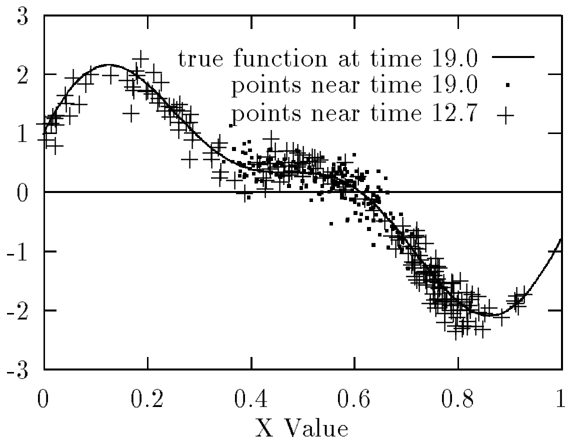

Figure 4: The function at time 19 and the data points near times 12.7 and 19.0

space we're interested in predicting. Referring back to the original function, we see that the time variation repeats every 2π seconds. If we look back that far and include the points occurring near time 12.7 with those near time 19.0, we increase the set of usable data. This is shown in fig. 4. This is what we hope to gain through the use of our algorithm. When historical data points become useful again, we want to include them with the more recent data. Using the algorithm outlined in the previous section we were able to achieve a test set error of 0.40.

The performance of the algorithm can be seen by observing the weights it places on the data points while attempting to predict the output for input value 0.8 at time 19. The weights for each data point are plotted against the time of that data point's occurrence in fig. 5. Note that none of the points near time 19 were given

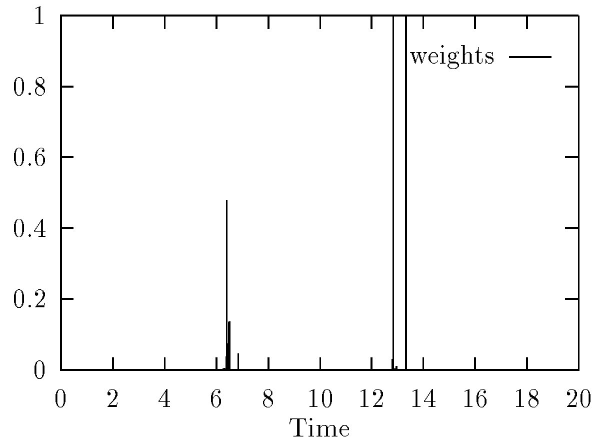

Figure 5: Data point weights as a function of their time for predicting the function value at input 0.8 and time 19.0

a significant weight because none of them are near input value 0.8 as seen in fig. 3. However, the algorithm found points near 12.7 and near 6.4 (2π and 4π seconds earlier) that it could use. The true function value is -1.84 and these extra points yielded a prediction of -1.91 while use of the recent points only (which only had x values from 0.4 to 0.7) yielded a prediction of -1.15.

We have presented an algorithm that looks for historical data that may be relevant to predictions in the present and uses it to fill out its current model of the function it is trying to approximate. Our preliminary results on a one-dimensional, gaussian noise, time-varying function are encouraging. When combined with an active learning system this algorithm may further improve its performance by allowing the learner to concentrate its experiments only in those regions where both no recent data and no identifiably relevant historical data exist.