A Reservoir Sampling Algorithm with Adaptive Estimation of Conditional Expectation

Vuk Malbasa and Slobodan Vucetic

Abstract— **Resource-constrained data mining introduces many constraints when learning from large datasets. It is often not practical or possible to keep the entire data set in main memory and often the data could be observed in a single run in the order in which they are presented. Traditional reservoir-based approaches perform well in this situation. One drawback of these approaches is that the examples not included in the final reservoir are often ignored. To remedy this situation we propose a modification to the baseline reservoir algorithm. Instead of keeping the actual target values of reservoir examples, an estimate of their conditional expectation is kept and updated online as new data are observed from the stream. The estimate is obtained by averaging target values of the similar examples. The proposed algorithm uses a paired t-test to determine the similarity threshold. Thorough evaluation on generated two dimensional data shows that the proposed algorithm is producing reservoirs with considerably reduced target noise. This property allows training of significantly improved prediction models as compared with the baseline reservoir-based approach.**

I. Introduction

In many real-life domains the amount of available data greatly exceeds the computational and storage capacity of devices used for learning from it. This challenge is often complicated by having data access only through a single sequential pass through the data. There are two major approaches in addressing this problem. The first is online learning that considers algorithms that are able to improve their quality as new examples arrive [1]. While some reasonably efficient algorithms exist [2, 3], they tend to be sensitive to the order in which examples are presented and there are often no guarantees that the learned model approaches the accuracy of the batch-mode alternatives [3].

The second, reservoir-based, approach relies on maintaining a representative sample of the observed data in memory and on applying standard batch-mode learning algorithms on the reservoir data. Assuming a reservoir with capacity to hold R examples, the simplest procedure [4] is to replace the t-th observed example from a stream with a randomly chosen reservoir example with probability $min(1, R/t)$. This procedure guarantees that the resulting reservoir is an unbiased sample of the observed data. The appeal of this approach is in its simplicity and insensitivity to example ordering. The drawback is that all observed examples, other than the R examples included in the reservoir, are simply ignored.

This paper proposes an enhancement to the reservoir-based algorithm for regression that makes use of all observed examples to generate a more informative reservoir. Let us denote the j-th reservoir example with (x_j, y_j), where x_j is a K-dimensional input vector, and y_j is a target variable. The basic idea is to replace the actual target value of the j-th reservoir example with its conditional expectation $E[y\,|\,x_j]$. If the estimate is successful, the resulting reservoir will have considerably reduced noise compared to the baseline reservoir algorithm. This will, in turn, lead to more accurate learning. To get the estimate for the j-th reservoir example, target values of all the similar observed examples are averaged. As examples are being observed the estimate is expected to approach the actual conditional expectation.

A major contribution to this paper is in a statistically-based approach for determining the similarity threshold. It is accomplished by introducing a modest memory overhead to the reservoir. The proposed reservoir algorithm does not require training of any predictor and has linear O(NMK) time complexity where N is the number of observed examples, M is reservoir size, and K is the data dimensionality. The experimental results indicate that the algorithm is successful in improving the learning accuracy.

II. Methodology

A. Problem Setup

The problem of reservoir sampling can be described in the following way. The original data set $D = \{(x_i, y_i),\ \ i=1\ldots N\}$, where N can be large, is observed sequentially in a single pass. The data should be used to decide on the content of reservoir that can hold a summary about R examples. Any observed example can be used to update the content of the reservoir but cannot be accessed after that. In this paper, we represent the j-th reservoir example as a tuple (u_j, z_j, a_j), where u_j is an input vector, z_j is a target variable, and a_j is an overhead vector containing ancillary information.

It is assumed that the data generating process that produced the original data can be described as

$$y_i = f(x_i) + \varepsilon_i,\ \varepsilon_i \sim N(0, \sigma^2) \qquad (1)$$

where $f(x_i)$ is a regression function, and ε_i is an additive noise term generated from Gaussian distribution with mean zero and standard deviation σ. In this case, the conditional mean $E[y\,|\,x_i]$ equals the regression function $f(x_i)$. The

Manuscript received January 31, 2007. This work was supported in part by the U.S. National Science Foundation under Grant IIS-0546155.

V. Malbasa and S. Vucetic are with the Center for Information Science and Technology, Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. (S. Vucetic phone: 215-204-5535; fax: 215-204-5082; e-mail: vucetic at ist temple edu).

objective of regression is to learn a prediction function from the data that resembles the regression function as close as possible. It is well known that the quality of learning deteriorates when target noise is large and improves with the size of training data. While in the reservoir sampling the data size is fixed, it is possible to reduce the target noise. The proposed algorithm accomplishes the noise reduction by estimating the conditional expectation.

B. *Basic Idea of the Algorithm*

Let us assume that tuple (u_j, z_j, a_j) is introduced to the reservoir at time t and that its initial values are set to $u_j = x_t$, $z_j = y_t$ (for now, we will ignore a_j). The basic idea of our approach is to update the value of z_j using examples observed after time t. Let us denote $I_j = \{i,\ i \ge t \wedge \|u_j - x_i\| < r_j\}$ as a set of indices of examples observed after time t that are at distance below r_j from u_j. We calculate z_j as an average of target values of these examples,

$$z_j = \frac{1}{|I_j|} \sum_{i \in I_j} y_i \,. \tag{2}$$

where $|I_j|$ is the number of examples in set I_j. It is evident that choice of distance threshold r_j influences the quality of updates.

C. *Theoretical Considerations of the Distance Threshold*

To better understand the influence of the distance threshold, we should consider the expectation $E[(z_j - f(u_j))^2]$, which measures how different the estimate z_j is from its desired value $f(x_j)$. By using the second-order Taylor expansion of the regression function near u_j

$$\begin{aligned} f(x) = {} & f(u_j) + \\ & (x-u_j)'\nabla f(u_j) + \frac{1}{2}(x-u_j)'\nabla^2 f(u_j)(x-u_j) \end{aligned} \tag{3}$$

and by observing from (2) and (1) that

$$z_j = \frac{1}{|I_j|} \sum_{i \in I_j} (f(x_i) + \varepsilon_i) \tag{4}$$

we can express $z_j - f(u_j)$ as

$$\begin{aligned} z_j - f(u_j) = {} & \frac{1}{|I_j|} \sum_{i \in I_j} \{(x_i - u_j)'\nabla f(u_j) + \\ & (x_i - u_j)'\nabla^2 f(u_j)(x_i - u_j) + \varepsilon_i\}. \end{aligned} \tag{5}$$

Using this expression, and by assuming that the distribution of input variables near u_j is uniform, it can be shown that $E[(z_j - f(u_j))^2]$ can be approximated as

$$\begin{aligned} E[(z_j - f(u_j))^2] = {} & \frac{\sigma^2}{|I_j|} + \frac{C_1 r_j^2}{|I_j|} \sum_k \left(\frac{\partial f(u_j)}{\partial x_k}\right)^2 + \\ & C_2 r_j^4 \sum_k \sum_l \left\{ 2\left(\frac{\partial^2 f(u_j)}{\partial x_k \partial x_l}\right)^2 + \frac{\partial^2 f(u_j)}{\partial x_k^2}\frac{\partial^2 f(u_j)}{\partial x_l^2} \right\}, \end{aligned} \tag{6}$$

where C_1 and C_2 are constants and the partial derivatives are with respect to the k-th (or l-th) input variable. The quantity $|I_j|$ depends on the observed data size N, radius r_j, and local data density around u_j. By assuming that data distribution around u_j is uniform, $|I_j|$ is proportional to

$$|I_j| \sim r_j^K \cdot (N-t) \cdot p(u_j), \tag{7}$$

where $p(u_j)$ is probability density at u_j and K is data dimensionality.

From (6) and (7) it can be concluded that the quality of z_j estimate increases with data size N and local data density and decreases with original data noise variance σ^2. It can be noted that, as $N \to \infty$, the first two terms approach zero, while the third term remains constant. Additionally, the estimation quality is inversely proportional to gradient and Hessian of the regression function at u_j.

The influence of distance radius r_j on the estimation is more complex. Increasing the distance radius increases $|I_j|$ and decreases the first term in (6). For $K > 2$ it also decreases the second term in (6). However, the third term in (6) is a bias term and it does not depend on $|I_j|$; it rapidly increases for sufficiently large r_j. It is evident that there is an optimal value of r_j that provides the best tradeoff between these two effects.

To derive the optimal r_j, we would need to know (1) the regression function, (2) the noise variance, and (3) the size of the available data. If the whole data set could be stored in the memory, these quantities could be estimated in an iterative manner similar to locally weighted regression [5] approach. However, in the reservoir sampling scenario, none of these quantities could be estimated due to memory constraints. Clearly, only heuristic approaches for estimation of the distance threshold r_j are acceptable in the streaming scenario. In the following section, we propose a statistically-motivated method for determination of the distance threshold.

D. *Statistical Method for Calculation of Distance Threshold*

Initially, very little is known about the data set. As the data stream is being observed, the goal is to rapidly understand the curvature (i.e. Hessian) of the regression function around each reservoir point because, based on (6), this is the crucial parameter for distance threshold determination. The heuristic proposed in this paper is to set the threshold to the largest value at which the curvature

```
Input: stream of N examples, reservoir of
size R, the initial radius $r_0$.

FOR i = 1 TO R
 // reservoir initialization
 read $(x_i, y_i)$ from stream
 $u_i = x_i$;  $z_i = y_i$;
 $r_{i2} = r_0$; $r_{i1} = r_0/(2^{1/k})$;
 $n_{i1} = 1$;  $n_{i2} = 0$;
 $m_{i1} = y_i$; $m_{i2}$ = NaN;
 $s_{i1} = 0$;  $s_{i2}$ = NaN;
END

WHILE not end of stream
  i = i + 1
  read $(x_i, y_i)$ from stream
  IF random(0,1) < R/I
  //replace a reservoir point
   j = random(1,R)
   $u_j = x_i$; $z_j = y_i$;
   $r_{j2} = r_0$; $r_{j1} = r_0/(2^{1/k})$;
   $n_{j1} = 1$; $n_{j2} = 0$;
   $m_{j1} = y_i$; $m_{j2}$ = NaN;
   $s_{j1} = 0$; $s_{j2}$ = NaN;
  ELSE
   FOR j = 1 TO R
    IF dist($x_i$,$u_j$) < $r_{j1}$
     update $a_{j1}$ using $(x_i, y_i)$
    ELSEIF dist($x_i$,$u_j$) < $r_{j2}$
     update $a_{j2}$ using $(x_i, y_i)$
    END
    IF $a_{j1}$ or $a_{j2}$ were changed
     evaluate the t-test
     IF t-test is rejected
      $a_{j2} = a_{j1}$;
      $r_{j1} = r_{j2}/(2^{1/k})$; $n_{j1} = 0$;
      $m_{j1}$ = NaN ;$s_{j1}$ = NaN
     END
    END
   END
  END
END
```

Figure 1. Pseudo code for the reservoir sampling algorithm with adaptive estimation of conditional expectation

remains negligible. A statistical test is proposed to determine such threshold.

The proposed method relies on maintaining two distance thresholds, r_{j1} and r_{j2}, where $r_{j2} > r_{j1}$. Initially, r_{j2} is set to some large value and r_{j1} to a smaller value that covers approximately half of the volume covered by r_{j2}. More specifically, upon insertion of the j-th reservoir point from the stream, the tuples $a_{j1} = (r_{j1}, n_{j1}, \mathrm{m}_{j1}, \mathrm{s}_{j1})$ and $a_{j2} = (r_{j2}, n_{j2}, \mathrm{m}_{j2}, \mathrm{s}_{j2})$ are created. The role of the first tuple is to maintain information about the smaller radius (r_{j1}), number of examples that fall within a sphere of radius r_{j1} around u_j (n_{j1}), the sum of target values of these examples ($m_{j1} = \Sigma_{Ij} y_i$), and the sum of squares of target values ($s_{j1} = \Sigma_{Ij} y_i^2$). Using m_{j1} and s_{j1} the mean and standard deviation of the target values can be calculated as $\mu_{j1} = m_{j1}/n_{j1}$ and $\sigma_{j1} = (s_{j1} - 2\mu_{j1}m_{j1} + \mu_{j1}^2)/n_{j1}$. Similar information is maintained for the larger radius r_{j2}, the only difference being that the statistics maintained are of points falling at a distance between r_{j1} and r_{j2} from u_j. Every subsequent stream example that is within distance r_{j1} from u_j is used to update tuple a_{j1}, while if it is at distance between r_{j1} and r_{j2} from u_j it is used to update a_{j2}.

After an update of either tuple a statistical test is used to evaluate the hypothesis that the two samples have equal means, $\mu_{j1} = \mu_{j2}$. The justification for this test is that if the regression function around u_j is approximately linear the means of the two samples should be equal; if a significant quadratic component exists in the regression function this will not be the case. Therefore, if the hypothesis is rejected it is concluded that smaller radius (r_{j1}) is more appropriate than the larger one. Then, tuple a_{j2} is overwritten with a_{j1}, and a_{j1} is re-initialized and used to explore an even smaller radius choice r_{j1}. In this paper, we used $r_{j1} = r_{j2}/(2^{1/K})$ that allows inner and outer spheres to contain approximately the same number of examples.

We decided to use the two-sample t-test because the distribution of target variables within a sufficiently small sphere can be approximated by Gaussian distribution. For this test we calculated the t-statistics with $n_{j1}+n_{j2}-2$ degrees of freedom as $T = (\mu_{j1} - \mu_{j2})/\sigma_j$, where σ_j is an unbiased estimator of the target variance derived from n_{j1}, σ_{j1}, n_{j2} and σ_{j2}. Given the value of T, the hypothesis is rejected if its p-value is below 0.05.

The proposed statistical method introduces a memory overhead to the reservoir in form of tuples a_{j1} and a_{j2}. More specifically, the j-th reservoir example is represented as (u_j, z_j, a_{j1}, a_{j2}). It is worth noting that this does not necessarily introduce a large burden to the reservoir. First, for the radii, assuming that we set the initial value of the larger radius to a known value for all reservoir examples, we only have to maintain how many times we rejected the hypothesis; this will be sufficient to determine r_{j1} and r_{j2}. The number of rejections is likely to be small and, therefore, easily compressible. Similarly, counts n_{j1} and n_{j2} are likely to be moderately small because after each hypothesis rejection the value of n_{j1} is set to zero. While values of μ_{j1} and μ_{j2} should be maintained with large precision, it is not the case with σ_{j1} and σ_{j2}. Finally, the value of z_j should not be maintained because it can be easily obtained from a_{j1} and a_{j2}. Overall, the resulting memory overhead is modest with an effect similar to adding a couple of additional attributes to the data.

E. The Algorithm

A pseudo code of the proposed reservoir sampling algorithm is shown in Figure 1. To update a_{j1}, m_{j1} is replaced with $m_{j1} + y_i$, s_{j1} with $s_{j1} + y_i^2$, and n_{j1} with $n_{j1}+1$. The t-test is evaluated by finding the p-value of the t-statistics $T = (\mu_{j1} - \mu_{j2})/\sigma_j$ with $n_{j1}+n_{j2}-2$ degrees of freedom, where σ_j is calculated as the total standard deviation of targets in a_{j1} and a_{j2}.

III. Experimental Reuslts

A. Data Description

We generated 2-dimensional data sets to evaluate the proposed reservoir algorithm. Input variables x_1 and x_2 were generated as uniformly distributed in a range between –1 and 1. The target variable y was generated using the following generating process:

$$y = sin(20x_1)/(20x_1) + x_2 + \varepsilon,$$

where ε is Gaussian additive noise with zero mean and standard deviation σ. The resulting regression function, with σ=0, is illustrated in Figure 2.

B. Experimental Design

The evaluation was performed on data streams of size 20,000 with three levels of noise, σ = 0, 0.2, 1, which represented noise-free, low noise, and high noise learning scenarios. We used two reservoir sizes, R = 100 and R = 500. This resulted in six combinations of reservoir size and noise levels in training data sets. For each of the 6 combinations we constructed 30 data sets and run the reservoir sampling algorithm on each of them.

For each reservoir example, we monitored three different labels. The first was the conditional mean value estimated by our algorithm. The second was the original target value from the training set that is maintained by the baseline reservoir algorithm. The third was the noise-free target value obtained directly from the regression function and it served to establish the lower bound on achievable learning accuracy. Ideally, our reservoir sampling algorithm would approach this lower bound as the number of observed example grows.

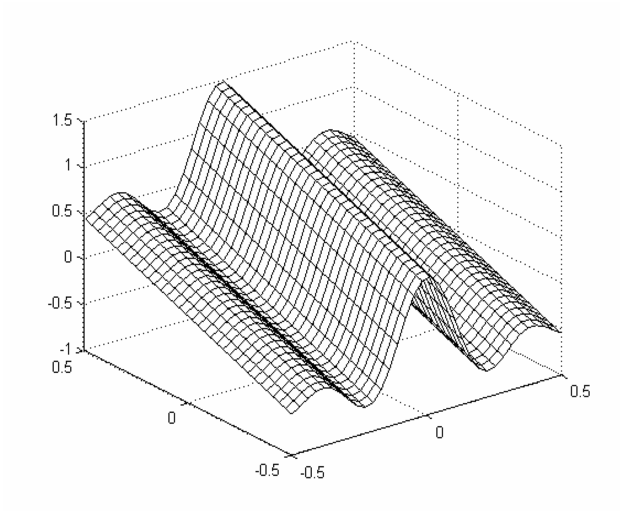

Figure 2. Visualization of the regression function used in the experiments

During each run, the content of the reservoir was examined at times N = 100, 200, 500, 1000, 2000, 5000, 10000, and 20000. At every reservoir snapshot and using each of the three types of target variables we trained 10 neural networks starting from different initial weights. As a result, for every choice of noise level σ, reservoir size R,

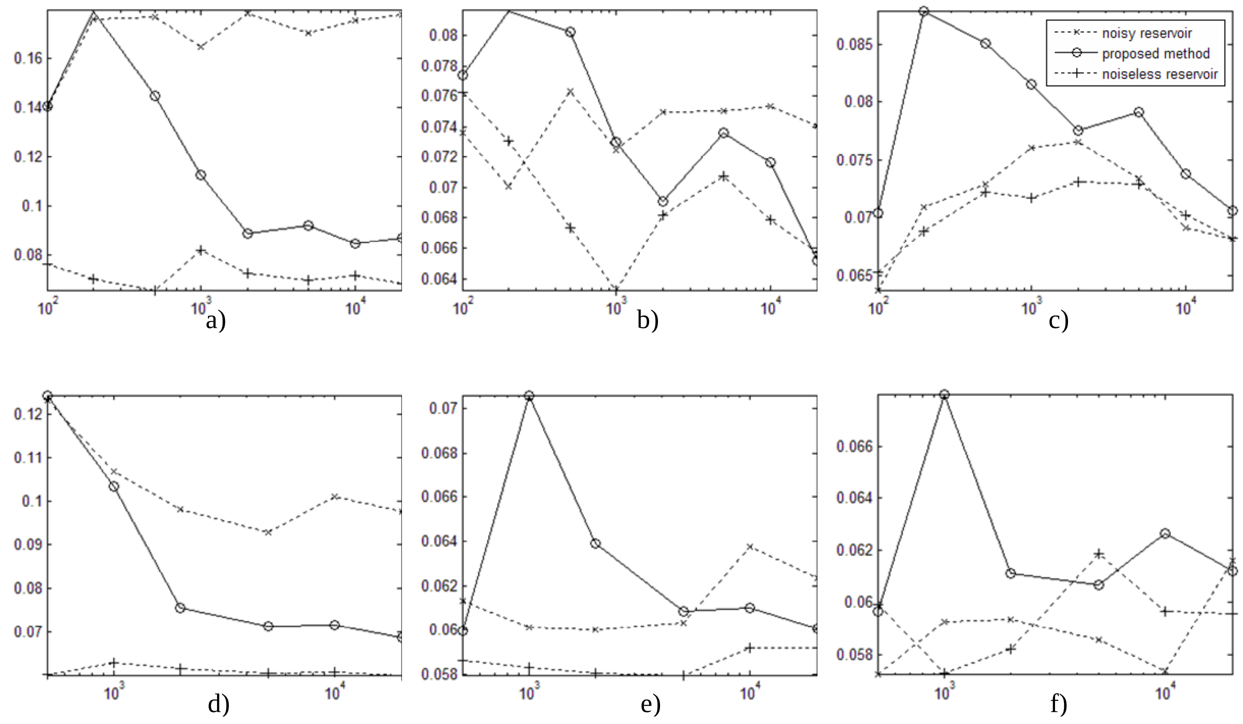

Figure 3. The graphs represent average MSE of networks trained on reservoir data (y-axis) as a function of a number of observed examples (x-axis). Top row corresponds to reservoir size of R=100 and the bottom row to a reservoir size of R=500. Noise levels in panels from left to right decrease (σ = 1, 0.2, 0). The lines correspond to networks trained on reservoirs generated from noisy data (‘x’), noiseless data (‘+’), and by the proposed method (‘o’).

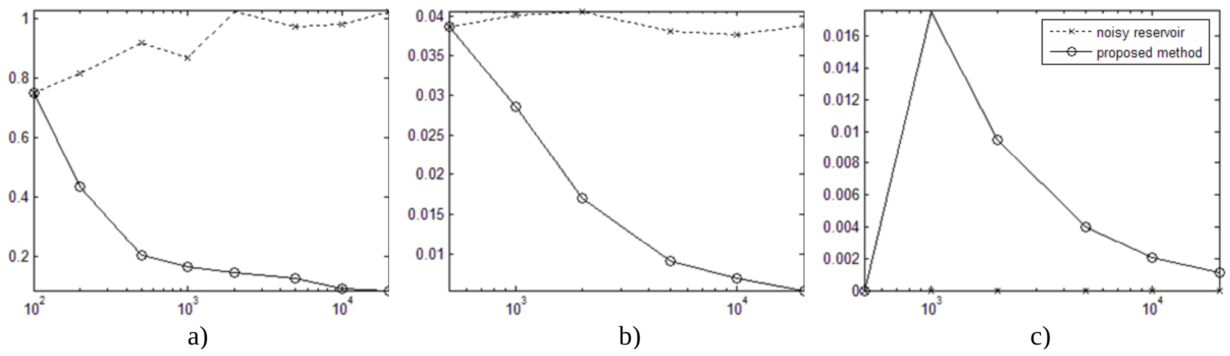

Figure 4. The graphs represent the Reservoir Noise (y-axis) as a function of the number of observed examples (the x-axis) for reservoir size R = 100 and with noise levels in panels from left to right decreasing, having values σ = 1, 0.2, 0. The lines correspond to Reservoir Noise of reservoirs generated from noisy data ('x') and by the proposed method ('o').

snapshot time *N*, and target type, we trained 10 neural networks and repeated this procedure 30 times; this required training of 67,200 neural networks. We used feedforward neural networks with 5 hidden nodes and trained them through 200 epochs of the resilient backpropagation training algorithm implemented by the Matlab Neural Networks toolbox.

The algorithm performance was evaluated by measuring the Mean Square Error (MSE) of neural networks on a noise-free test data set of size 20,000. We also measured the Reservoir Noise, defined as the average squared difference between noise-free targets and our conditional mean estimates. It is expected that this quantity decreases with *N*.

C. *Results*

Figures 3 and 4 illustrate how the quality of reservoirs improves as more and more points are read from the stream. The horizontal axis in both figures is in logarithmic scale and represents the number of examples observed from the stream. In Figure 3, the vertical axis is MSE accuracy. For very noisy data with σ = 1 (Figs 3.a, 3.d) we observe significant improvement in learning accuracy as more examples are being observed by the algorithm. Initially, the accuracy is equal to that achievable by the traditional reservoir sampling. Very rapidly, it approaches the accuracy achievable when learning on a noise-free data set. The performance with *R* = 100 (Fig 3.a) and *R* = 500 (Fig 3.d) is qualitatively similar. For smaller noise levels with σ = 0.2 (Figs 3.b, 3.e) the improvement in accuracy is evident with increase in the number of observed examples, and again, it approaches the noise-free scenario rather quickly and becomes superior to the traditional reservoir approach. In Figure 4 we show the Reservoir Noise levels and for both σ = 1 (Fig 4.a) and σ = 0.2 (Fig 4.b) and observe a strong decrease in noise levels that approaches zero in both cases.

The behavior for noise-free scenario with σ = 0 (Figs 3.c, 3.f, and 4.c) is expected. The averaging within a small radius introduces noise to the original noise-free target values. However, the algorithm is quickly realizing that the original data has low noise and is rapidly reducing the distance threshold. As a result, as the number of observed examples becomes large, the algorithm is approaching the performance of the noise-free scenario. This result shows that the proposed algorithm is robust even in noise-free scenarios.

IV. Conclusion

We proposed a reservoir-based algorithm that achieves accurate estimation of conditional expectation for each reservoir example. The reduction of target noise of reservoir examples leads to more accurate learning from them. The proposed statistical method for determining distance threshold for the estimation of conditional expectation is robust in both low and high noise scenarios. The algorithm is computationally efficient because it is proportional with the size of the reservoir and the number of observed examples. It introduces a slight memory overhead with each reservoir example due to the need to store ancillary information for the conditional expectation estimation.

While the experimental results on low-dimensional data are promising, the algorithm in its current form is likely to be less successful for the high dimensional data due to the curse of dimensionality. Future work is needed to design an algorithm suitable for high dimensional data. Such an algorithm would most likely have to rely on the sensitivity analysis of the regression function to the input variables. An interesting question would be how to perform the sensitivity analysis in the streaming data scenario.

References

[1] R.S. Sutton, S.D. Whitehead, "Online learning with random representations," *International Conference on Machine Learning*, 314-321, 1993.

[2] G. Cauwenberghs, T. Poggio, "Incremental and decremental support vector machine learning," *Neural Information Processing Systems*, 409-415 2000.

[3] P. Domingos, G. Hulten, "Mining high-speed data streams," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 71-80, 2000.

[4] J.S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software*, 11(1), 37-57, 1985.

[5] S. Schaal, C. Atkeson, "Robot juggling: An implementation of memory-based learning," *Control Systems*, 14, 57-71, 1994.