DATA STREAM MINING WITH MULTIPLE SLIDING WINDOWS FOR CONTINUOUS PREDICTION

Omer Mimran
Ben-Gurion University of the Negev, Beer-Sheva, Israel, omer.mimran@intel.com

Adir Even
Ben-Gurion University of the Negev, Beer-Sheva, Israel, adireven@bgu.ac.il

Omer Mimran and Adir Even, 2014, "DATA STREAM MINING WITH MULTIPLE SLIDING WINDOWS FOR CONTINUOUS PREDICTION", Proceedings of the European Conference on Information Systems (ECIS) 2014, Tel Aviv, Israel, June 9-11, 2014, ISBN 978-0-9915567-0-0
http://aisel.aisnet.org/ecis2014/proceedings/track08/1

DATA STREAM MINING WITH MULTIPLE SLIDING WINDOWS FOR CONTINUOUS PREDICTION

Complete Research

Mimran, Omer, Ben-Gurion University, Beer-Sheva, Israel, Omer.mimran.om@gmail.com

Even, Adir, Ben-Gurion University, Beer-Sheva, Israel, adireven@bgu.ac.il

Abstract

Data stream mining (DSM) deals with continuous online processing and evaluation of fast-accumulating data, in cases where storing and evaluating large historical datasets is neither feasible nor efficient. This research introduces the Multiple Sliding Windows (MSW) algorithm, and demonstrates its application for a DSM scenario with discrete independent variables and a continuous dependent variable. The MSW development emerged from the need to dynamically allocate computational resources that are shared by many tasks, and predicts the required resources per task. The algorithm was evaluated with a large real-world dataset that reflects resource allocation at Intel's global data servers cloud. The evaluation assesses three MSW treatments: the use of multiple sliding-windows, a novel iterative mechanism for feature selection, and adaptive detection of concept drifts. The evaluation showed positive and significant results in terms of prediction quality and the ability to adapt to swift and/or graduate changes in data stream characteristics. Following the successful evaluation, the adoption of the proposed MSW solution by Intel led to cost savings estimated in millions of dollars annually. While evaluated in a specific context, the generic and modular definition of the MSW permits implementation in other domains that deal with DSM problems of similar nature.

Keywords: Data Stream Mining, Sliding Windows, Concept Drift, Dynamic Resource Allocation, Real-Time Business Intelligence.

1 Introduction

Data stream mining (DSM) has evolved from the more traditional data mining (DM) paradigm, motivated by the growing need to process and analyzed fast-accumulating online data. DSM introduces novel techniques and high-performance algorithms that deal with scenarios in which storing and/or training models with the entire historical data is neither feasible nor efficient. The growing interest in DSM, both in research and in practice, arises from the growing velocity of data generation in many different information systems (IS) settings, and the need to process and analyze newly-generated data within a short period of time - what is often referred to as "near real-time" analysis. The more traditional DM often mandates lengthy pre-processing of large pre-stored datasets. In "near real-time" scenarios, processing or storing the entire dataset is often infeasible performance-wise. Moreover "near real-time" evaluations are often subject to "concept drifts" - statistical behavior that changes rapidly over time, with newly-emerging patterns. In datasets that reflect rapid concept drifts, older records might quickly become irrelevant and can no longer be used for DM.

This study develops and evaluates the Multiple Sliding Windows (MSW) algorithm, designed for (but not limited to) decision scenarios with discrete independent variables (IDV's) and continuous dependent variable (DV). The MSW development has emerged from the domain of dynamic resource allocation (DRA), which deals with the need to allocate computational resources for an ongoing sequence of incoming tasks - an acute problem especially in data centers where a limited number of resources are shared by many users and tasks. Efficient DRA might become critical for firms that offer Data-as-a-Service (DaaS) capabilities and wish to comply with customers' demands for server resources within reasonable costs. The MSW deals particularly with an important pre-step of a typical DRA process - predicting the required resources per task, based on certain known task characteristics.

This study offers in-depth evaluation of three specific treatments that are included in the MSW:

- **Multiple sliding windows:** The integration of sliding-windows technique within the MSW implies running a few time windows in parallel, where only the recent observations in each are used to generate predictions while older ones are discarded.
- **Feature selection:** The MSW introduces a novel method for forward-selection of features, which is suitable for problems with discrete independent variables and continuous dependent variable.
- **Concept-drift management:** The MSW introduces a novel mechanism for detecting changes in the statistical characteristics of the data stream, and adapting the window size accordingly.

Those treatments were tested with a large dataset (~80 million records) that reflects DRA in Intel's global data centers cloud. The evaluation showed powerful results in terms of prediction quality and adaptation to concept drifts. The promising results have led Intel to adopt the solution, where the cost-saving that could be attributed to the incorporation of the MSW algorithm is estimated in millions of dollars annually. Beyond the successful application at Intel, the generic and modular nature of the MSW permits its implementation in other domains as well, for similar non-stationary DSM problems.

The remainder of this work is organized as follows: the background section reviews relevant DSM studies that influenced the MSW, and highlights some of the techniques that were incorporated during its development. The MSW-development section introduces its high-level architecture first, and then its specific components in more detail. The following section describes the empirical evaluation of the three treatments described above. The evaluations results, which demonstrate the potential contribution of the MSW within the evaluated context, are summarized and discussed. The concluding section summarizes the key contributions if this study and proposes directions for future research.

2 Background

This research proposes a data stream mining (DSM) algorithm and evaluates in the dynamic resource allocation (DRA) domain. DRA deals with large task sequences that must be supported by limited resources within a reasonable time. DRA applies particularly in large data centers, such as the one evaluated. Powerful servers are relatively expensive, and firms that handle extensive demands for data services (e.g., cloud-service providers) must apply DRA methods toward cost-effective server utilization. A key DRA challenge is the required pre-step of estimating the demand for resources per task. A few studies addressed this issue using regression methods (e.g., Zhang et al., 2007). Others have bypassed it by assuming negligible, fixed, or pre-known demands (e.g., Hu et al., 2009). This study offers assessment of per-task demands using predictive business analytics and DSM techniques.

The terms Business Intelligence (BI) and Business Analytics (BA) are broadly used to describe systems and technologies that support data-driven decision making (Chen et al., 2012). Core BI capabilities include means for data storage and processing, as well as tools for reporting, analysis, and visualization (Watson, 2009). The so-called "traditional BI" is reactive and descriptive in nature, being geared toward evaluating pre-stored and pre-processes data. Future BI system will be required to take

a more proactive and prescriptive approach - respond to situations online and propose actions (Watson, 2009). Accordingly, future BI systems will need to incorporate predictive analytics that can guide decisions and recommend actions accordingly (Chen et al., 2012). The term "Real-Time Business Intelligence" (RTBI) reflects systems that provide, in addition to traditional-BI functionality, means for analyzing data and performing corrective actions in a "near real-time" fashion, with minimum latency from data acquisition (Azvine et al., 2006; Sahay and Ranjan, 2008). The business scenario evaluated in this study can be associated with RTBI – a need to process an online data stream that must be analyzed rapidly, toward efficient resource allocation. The nature of the problem does not permit lengthy evaluation of historical data; hence, cannot be addressed by the traditional BI solutions.

Most statistical and data mining (DM) techniques for generating predictive analytics apply for static datasets. However voluminous data is often produced continuously - manufacturing, network monitoring, sensor readings, E-commerce transactions, and others. Standard database systems and statistical techniques often fail to handle high-volume data streams (Domingos and Hulten, 2001; Gama and Rodrigues, 2007). Moreover, many such techniques assume stationary behavior – i.e., stable statistical distributions over time. In reality, distributions are likely to change, and the validity of predictive models deteriorates – what is often referred to as "concept drifts" (Gaber et al., 2005). This issue is addressed by data-stream mining (DSM) - "An ordered sequence of instances that can be read only once or a small number of times using limited computing and storage capabilities. These sources of data are characterized by being open-ended, flowing at high-speed, and generated by non-stationary distributions in dynamic environments" (Gama and Rodrigues, 2007). DSM research (e.g., Gaber et al., 2005; Gama and Rodrigues, 2007; Bifet, et al., 2010) has identified a few key challenges:

- **Performance:** given limited memory, storage and/or processing capability - performance is possibly the foremost DSM challenge. With the increasing volumes, processing data in multiple passes might become infeasible. With extremely rapid pace of data generation, once a record is processed, it has to be discarded or archived; hence, cannot be reprocessed.
- **Quality:** with the need to process data rapidly, there is only very limited ability to validate data correctness and integrity. With no validation, the accuracy of decision models and their usability for prediction might become questionable.
- **Adaptability:** data streams often reflect dynamically-changing settings; hence the need to handle "concept drifts". DSM models must therefore be designed to detect drift in the underlying data and find the right balance between reflecting past behavior versus adjusting to changes.
- **Availability:** with the more traditional DM approaches, decision and prediction models become available only after lengthy training processes. DSM models have to be constantly trained online, and become available within a short time period after the data is being processed.

DSM literature has explored a few possible approaches for addressing the challenges discussed above. We describe next a few approaches that were incorporated into the development of the MSW:

- **Sliding Windows (SW)**: as processing the entire dataset is often infeasible, DSM solutions often use time windows that limit the data subsets observed. Gama and Rodriguez (2007) describe three main approaches for setting time windows: a) *Landmark* - maintaining only data that starts from certain relevant points (i.e. beginning of day), b) *Tilted* - maintaining data at different aggregation scales, where recent data is kept as-is, while older data is compressed, and c) *Sliding* – processing only the most recent data. The latter approach has the advantage of underscoring recent data, which is assumed to be more important and relevant (Babcock et al., 2002). Setting the sliding-window size may have substantial effect on DSM performance, processing time, and memory usage (Hua-Fu and Suh-Yin, 2009). It may use heuristics that are based on tree-root values, target entropy and error rates (Last, 2002), simulated evaluation (Fares et al., 2002), or the K-Nearest-Neighbors (kNN) approach (Chen et al., 2006).

- **Feature Selection (FS):** complex classification problems often require reduction of the number of input variable as a pre-step for training the model toward improving classification performance – a phenomenon that has been named the "Hughes effect" (Hughes, 1968) or the "curse of dimensionality". FS may grant some additional benefits, such as facilitating data understanding, accelerating model calculation, and improving prediction performance (Guyon and Elisseeff, 2003). Literature has offered a plethora of FS methods, where probably the most popular one is the stepwise regression - a greedy iterative algorithm that adds or removes input variables per iteration, based on ranking criteria such as Pearson Correlation, Adjusted R^2, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and others.
- **Concept Drift (CD)**: some statistical-inference methods are based on the assumption of stationary distribution. Distribution change over time – e.g., in class prior P(c), class distribution P(X|c), and/or posterior distribution P(c|X) - affect statistical inference, a phenomenon termed as "concept drift". Zliobaite (2009) highlights a few possible drift patterns: a) *Sudden drifts*, which are steep and easy to relatively easy to detect. b) *Gradual drifts*, in which the older distribution blends with the new, in manner that might be mistakenly detected as random noise, c) *Incremental drifts*, in which the change occurs in small steps over a period of time, and d) *Recurring context*, in which a concept re-occurs in a cyclic manner. To handle concept drifts, DSM algorithm often incorporate change-detector components, which trigger adaptive action upon detection of growing error rates. Adaptation can be based upon resetting the training data (Klinkenberg, 2004; Cohen et al., 2008), or training a shadow model (Domingos and Hulten, 2000; Ikonomovska and Gama, 2008; Bifet and Gavaldà; 2009). Some adaptive-learning methods apply ensembles – training several models and choosing the best-performing one (or a subsets\s) for prediction (Tsymbal et al., 2008).
- **Summary Statistics (SS)**: the use of sliding windows and the detection of concept drifts raise the need to maintain summary statistics for data that has been processed and analyzes, but exited the window. Datar et al. (2002) point out the inherent tradeoff with respect to statistics granularity – maintaining detailed summaries consumes space and slows-down processing, while highly-aggregated statistics reduce the ability to query and understand past behavior. Several methods for maintaining distribution histograms address this tradeoff. Simpler methods suggest using equal-length buckets (Babcock et al., 2002), or setting bucket boundaries by approximating variable distribution and minimizing SSE (Ijagadish et al., 1998). The MSW algorithm developed in this study is designed to support the Exponential Histograms (EH) approach (Datar et al., 2002) – a histogram-merging mechanism, that lets the impact of older histograms decay over time.

DSM literature has offered a few algorithms that apply the principles discusses above:

- **Incremental Online Info-Fuzzy Network (IOLIN) - (Cohen et al., 2008):** The IOLIN is based on info-fuzzy network in which each node includes one attribute and splits to two leaves. The network maximizes the mutual information between the input and the target attributes, and uses a pre-pruning strategy which prevents new splits that do not reduce the conditional entropy of the target attribute. The model uses sliding windows with a dynamically-changing size. Upon the arrival of new records, their classification accuracy is compared to the training-window accuracy. A statistically-significant degradation in accuracy indicates a concept drift and triggers the incremental training of a new model – updating only the last network layer, or adding a new layer.
- **Hoeffding Tree (HT), Very Fast Decision Trees (VFDT) and Concept-Adapting VFDT (CVFDT) - (Domingos and Hulten, 2000; Hulten et al., 2001):** The HT is an incremental decision-tree classifier for data streams, based on Hoeffding's bound (1963), for determining the number of samples required per node. The VFDT extends the HT by adding mechanisms for preventing addition of unnecessary nodes, avoiding extension of the training sample beyond necessary, and using RAM efficiently. The CVFDT extends VFDT, by handling concept drifts. It runs the VFDT algorithm over a sliding window, recalculates the validity of tree-splits, and adds new sub-trees in cases of statistically significant changes in prediction accuracy.

- **Adaptive Windowing (ADWIN) and Hoeffding Adaptive Tree (HAT) – (Bifet and Gavaldà, 2007; 2009):** The ADWIN deals with concept drifts by maintaining a window of varying length and dropping the older portion of it when a concept drift occurs. The HAT manages statistics in the nodes, which are used to detect concept drifts. This permits using varying-size windows per node and creating alternate trees at the node level as soon as change is detected.
- **Fast Incremental Model Tree (FIMT) and Fast Incremental Regression Tree with Drift Detection FIRT-DD - (Ikonomovska and Gama, 2008; Ikonomovska et al., 2009):** The FIMT handle continuous prediction by using perceptron learning with incremental updated of weights. The FIRT-DD, an adaptive version of the FIMT, includes change detectors at the nodes. It uses Statistical Process Control (SPC) methods for tracking absolute loss as a performance measure.

Most algorithms described above address classification problems and accordingly handle discrete output variables. The FIMT and FIRT-DD handles continuous input and output variables. This study addresses a gap in the DSM literature – decision scenarios in which the input variables are discrete, but the output is continuous. However, as described in the following section, the development of the MSW incorporated some of the principles proposed by previous DSM algorithms.

3 The Multiple Sliding Windows (MSW) Algorithm

The multiple sliding windows (MSW) algorithm targets data stream mining scenarios in which the input variables are discrete and the output variable is continuous (but can be adjusted to scenarios in which the output is discrete). A scenario as such is described and evaluated in the next section – predicting the amount of resources required per task, given certain task characteristics. As described here – the general MSW approach and some of its components offer novel contributions, while other components adopted state-of-the-art techniques described in earlier DSM literature. Figure 1 shows the MSW high-level architecture, and highlights the treatments were evaluated.

Mode of operation: A novel concept that was implemented in the MSW is the profiling of observations, and the handling of multiple separate sliding windows, one per profile. A sliding window stores and processes only the most recent data, which is assumed to be more important and relevant (Babcock et al., 2002). Sliding windows have been applied in earlier studies; however, in most previous implementations, all data stream observations were included in the same window.

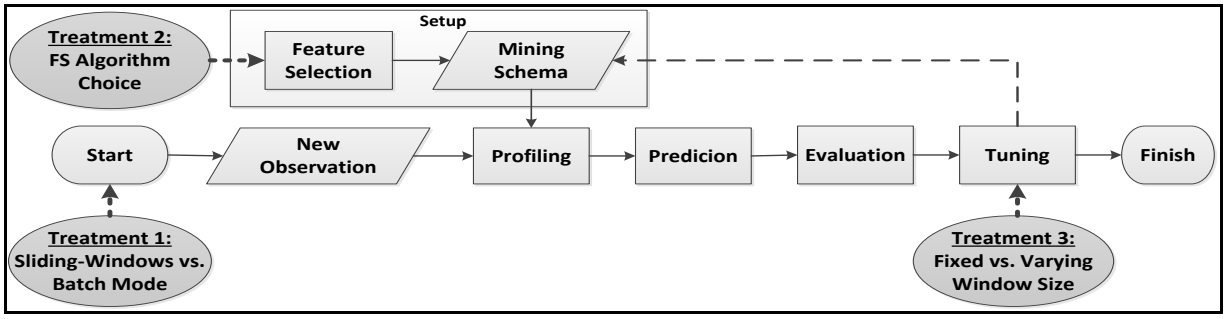

Figure 1. The Multiple Sliding Windows (MSW) Architecture and the Tested Treatments

The first evaluated treatment reflects the mode of operation - multiple sliding windows versus processing the entire dataset in one batch. A sliding-windows mode implies that the algorithm runs for relatively short time intervals, and that the model is trained incrementally, nearly on-line. Given this mode, the algorithm is expected to respond to changes in data behavior and generate predictions based on recent observations. The sliding-windows approach is therefore expected to significantly outperform that batch mode - running the model for the entire dataset is expected to provide less accurate predictions due to the effect of outdated observations, and have a much slower response time.

Setup: Each profile reflects a category of observations with some common properties and business meaning. The definition of profiles and the rules for classifying observations to profiles are managed in the mining schema, where the classification is based upon a set of profile features (or attributes). The feature set is defined in advance, based on understanding the business domain, or by applying a feature selection (FS) algorithm in cases where the observations include a large number of features. The goal of FS is to reduce the number of input variable toward improving classification performance, facilitating data understanding and accelerating model calculations (Guyon and Elisseeff, 2003).

The second evaluation treatment in this study aims at assessing the performance impact of FS algorithms. The test evaluated the use of the entire set (i.e., no FS), versus the ensemble FS algorithm in (Tuv et al., 2009), and versus the Data Dictionary Selection (DDS) FS algorithm - another novel contribution of this study. The DDS is influences by the Akaike Information Criterion (AIC), which measures the goodness-of-fit of statistical models, by grading models on their variance and panelizing on too-high parameter complexity (Akaike, 1974). Similarly, the DDS criterion measures the weighted variance of profiles with some penalty on a too-high number of profiles:

$$V_0 = \sigma^2; \quad P_0 = 1; \quad V_i = \sum_{j=1}^{J} \begin{cases} \left(\sigma_j N_j\right)/N & N_j \geq \alpha \\ 0 & Otherwise \end{cases}$$
$$DDS_i = \left(V_i / V_{i-1} - \sqrt{P_i / P_{i-1}}\right)\left(\sum_{j=1}^{J} \begin{cases} N_j & N_j \geq \alpha \\ 0 & Otherwise \end{cases}\right)\Big/ N \qquad (1)$$

- **[i]-** The iteration-step number
- **N-** The total number of dataset observations
- **J-** The number of profiles considered
- **P_i-** The number of profiles generated in step [i]
- **V_i-** The variance in step [i]
- **σ_j-** The standard deviation of profile [j]
- **N_j-** The number of observations in profile [j]
- **α -** The minimum number of observations required per profile (i.e., minimum support)

The algorithm is applied iteratively (with iteration index [i]), where the iterations terminate when the current DDS is no longer smaller than the previous DDS. At each step the DDS expression is normalized to represent the variance reduction and the increase in the number of profiles from the previous step. This normalization allows using the criteria for different problems with different value scales. Reducing the expression by the square root of the profile-number proportion prevents the criteria from being too sensitive to the number of profiles. The sensitivity to the number of profiles is reduced further by using the minimum-support filter α discards profiles with too-few observations.

Profiling: Upon arrival, each observation is matched to the mining schema that represents profile definitions, or creates a new profile if no matching profile can be found. Once assigned to a profile, the observation is directed to the appropriate sliding window. As each window may need to handle large numbers of observations, maintained row-level data may not be feasible performance-wise; hence, data must be compressed to an extent. Further, it can be assumed that observations' contribution to accurate prediction may decreases significantly over time. The MSW handles data compression and reduces the impact of older observations by using exponential histograms (EH) – a mechanism that has been implemented in previous DSM works (e.g., Datar et al., 2002). MSW configuration requires setting up the following, toward reducing the number of profiles and maintaining statistical validity:

- **Validation Filter:** a set of rules, based on knowledge of the business domain, for detecting invalid observations that cannot enter the sliding-window and be used for learning.
- **Minimum Size:** the minimum number of observations for starting a profile, toward preventing too many profiles, and filtering out rare cases that are not likely to gain statistical support.
- **Minimum Support:** the minimum number of observations required for using the profile for prediction – also refer to as "sufficient statistics" (Bifet et al.; 2010). Predicting with insufficient number of observations might lack statistical validity and cause high error rates.

Prediction: Once a profile window has reached the required size (the "minimum support"), it can be used for prediction. The MSW modularity permits the use of various predictions, depending on the problem's nature. With classifications, for example, the prediction would assign each instance to a class, based on distance or statistical similarity. For continuous output, for example, the prediction could be based on regression. In the scenario evaluated in this study, as described further in the next section, the output was continuous and the prediction function matched each observation to a certain percentile of the output range. When sliding windows are used, the prediction per profile is based only on active observations (the most recent within the window), while older predictions are discarded.

Evaluation: during the MSW operation, various indicator types are collected, which are later used to for tuning certain model parameters, usage reporting, and analysis of prediction quality:

- **Usage indicators:** help assessing and monitoring the ongoing model operation – e.g., the overall number of observations evaluated at certain time windows, the number of profiles, and the number of observations entering each profile (based on the "learning criteria").
- **Performance:** help assessing model outcomes and prediction quality – e.g., prediction accuracy, mean square error (MSE), and statistical validity per profile.

Tuning: The window-size setup - i.e., the number of observations within a window, usage for prediction - governs the inherent tradeoff between fast reaction to concept drift (motivating a smaller window size) and statistical validity (motivating a larger window). DSM literature reflects two main approaches for setting the window size – a fixed pre-defined size, versus a variable size that changes dynamically during operation. The third treatment that was evaluated in this study tests the performance impact of a novel MSW concept-drift management, which is based setting the window size per profile dynamically, using an iterative change-detector mechanism (the test is versus a fixed pre-set window size). The change detector requires setting the following parameters:

- **W, ΔW:** The minimum window size that enters the change detector, and the window size increment (greater than the minimum support parameter), respectively
- **W_i:** The evaluated window size at step [i]
- **S_i, ΔS:** The confidence level at step [i], and the confidence-level increment, respectively

The change detector is triggered when the window size reaches W. The first-iteration window size is set to $W_1=\Delta W$, and the first-iteration confidence level is set to $S_1=1-\Delta S$. Iteration [i] is defined by:

1. Split the window into two sub-windows – sub-window A with the W_i most recent observations versus sub-window B with the $W-W_i$ older observations
2. Compare the output-variable distributions of A versus B, with a confidence level of S_i.
3. If the change is significant – remove the observation in sub-window B, and exit iterations
4. If the change is insignificant – set $W_{i+1} = W_i + \Delta W$, $S_{i+1} = S_i - \Delta S$
5. If $W_{i+1} > W$ exit iterations, otherwise move to the next iteration at step [i]

The iterative increase in window size and decrease in confidence level reflects a shift from the detection of sudden versus gradual concept drifts. A sudden drift requires discarding older data; hence, the need for a high confidence level. A gradual drift requires discarding fewer observations; hence, permits lower confidence levels. The distribution comparison of newer versus older observations may use various change-detector functions – the evaluation here applies the Hoeffding bound, which has been used successfully in previous DSM studies (e.g., Hulten et al., 2001; Bifet and Gavaldà, 2009). Obviously, the modular nature of MSW and the various parameters involved require a more precise configuration, which may depend on the DSM scenario as described in the following section.

4 Empirical Evaluation

The MSW was evaluated for a non-stationary data stream that reflects real-world dynamic resource allocation (DRA). Intel's chip-design efforts are supported by a global "cloud" of servers, which at the time of the evaluation included over 300,000 cores and provided over 45 million computation hours weekly. Chip-design jobs (billions annually) are sent to the cloud, enter a queue, and allocated to available servers based on some complex decision rules. The demand for servers often exceeds the supply, with severe impact on cost and productivity. Further, many jobs fail due to mismatches between memory capacity and actual job requirements. To expedite the queue and optimize allocation, users are required to estimate the required memory consumption per job. Overestimation may lead to inefficient utilization, while underestimation might cause execution failures and excessive power use. Studies at Intel indicated that ~90% of the jobs are overestimated; hence, improvement of estimations is seen as a critical mission. Beyond reducing direct costs and job-loss damages, expediting the queue would also improve engineers' productivity, shorten time to market of new products, and reduce investments in hardware. Attempts to educate users to provide precise estimations gained only limited success so far, hence the motivation to improve and automate this task.

The evaluated dataset included 80 million observations collected over a period of 87 days, each reflecting a single job. The first three days were used for setup, and each of the remaining 84 was evaluated separately. Four indicators (described later) were measured per day, so that for each indicator and for each combination of treatments 84 values were obtained. Each observation included 17 independent categorical variables (V01-V17) that reflect job characteristics (e.g., location and group of origin, type of task, urgency rank), a job-submission timestamp, the memory allocated in GBs (based on users estimations), the actual memory consumed, and the total time of execution in seconds (Table 1). The allocation of an observation to a profile window is based on the combination of values in variables V01-V17, or a subset of those variables, if feature selection was applied.

A preliminary data analysis reflected a "long tail" distribution of the memory consumed (median of ~0.5 GBs) and the execution time (median of ~6 minutes). Memory estimation was shown to be wasteful – the actual consumption was mostly much lower that the allocation (median of ~ 2 GBs). Memory prediction had to balance between *a) accuracy* – the ratio of jobs with memory predicted greater than the consumed, and *b) saving* – the gap between memory predictions versus estimations, considering runtime. High predictions increase accuracy but reduce saving and vice versa. The desired memory prediction target was defined by setting the lowest possible predicted value (i.e., maximize saving) that still guarantees accuracy greater than a certain threshold φ (set to 0.95 in this experiment).

Var.	**Val.**	**Var.**	**Val.**	**Var.**	**Val.**	**Var.**	**Val.**	**Var.**	**Val.**
V01	[536]	**V05**	[76631]	**V09**	[320]	**V13**	[909]	**V17**	[404]
V02	[816]	**V06**	[2]	**V10**	[2]	**V14**	[37]	**Memory Allocated**	GBs
V03	[269]	**V07**	[2]	**V11**	[11]	**V15**	[3]	**Memory Consumed**	GBs
V04	[517]	**V08**	[11]	**V12**	[11]	**V16**	[3622]	**Execution Time**	Seconds

Table 1. Dataset Variables ([Number of Distinct Categories])

The MSW algorithm was pre-configured as follows:

- **Validation Filter:** the servers-cloud managers indicated that only jobs with execution time of 30 seconds or more are valid, as a shorter runtime indicates early termination. Invalid jobs are kept for some indicator calculation, yet they are not used for generating prediction models.
- **Minimum size:** to prevent profiles that reflect rare cases, starting a new profile required at least 6 observations, and among them at least 1 valid.
- **Minimum Support:** to prevent too weak statistical confidence, a profile-window required at least 200 observations to be used for generating predictions.

The prediction function for profile-windows that gained the minimum support was defined as:

$$f(W,\varphi) = x_i + C; \quad i = \lfloor \varphi * N \rfloor, \quad x_i \in W \qquad (2)$$

- **W-** The window evaluated
- **N-** The total number of observations in W
- **[i]-** Observation index, sorted in an increasing order of memory consumed
- **φ-** The required percentile (set to 0.95 in this experiment)
- **X_i-** The memory consumption of observation [i]
- **C-** Prediction stability constant (set to 0.1 GBs in this experiment)

The prediction selects a memory consumption value that guarantees accuracy level of at least φ (here – 0.95) – i.e., a value that guarantees that at least φ of the N observations within window W will get sufficient memory allocation. The prediction stability constant C adds a small margin to the predicted value (here set 0.1 GBs) that lowers the prediction variance and improves prediction stability.

As discussed previously, the evaluation regards 3 treatments that reflect MSW configuration decision:

- **Treatment 1:** Applying sliding windows versus execution in one batch
- **Treatment 2:** DDS-based feature selection (FS) versus ensemble-based FS and no FS
  - **DDS-based selection:** V02,V03,V06,V07,V09,V10,V11,V14,V15,V17
  - **Ensemble-based selection:** V02, V03, V08,V12,V15,V16,V17
- **Treatment 3:** Concept-drift management by varying window size, versus fixed window size

The evaluation tested each treatment separately, and all the possible interactions between them. Using the timestamps, the evaluation simulated a DSM process over the jobs queue, where the observations were processed in sequence. Most observations entered a profile-window that permitted prediction, and for those that did not, the estimated prediction was used instead. Based on attribute values and the predicted memory consumption, the following observation-level indicators were calculated:

Indicator	Value
Predicted	1 if the job received a prediction, 0 otherwise
Accurate	1 if the memory predicted is greater or equal to the memory consumed, 0 otherwise
Utilization	Memory allocated * Execution Time
Potential Saving	For accurate jobs: (memory allocated–memory consumed)*Execution Time, 0 otherwise
Actual Saving	For accurate jobs: (memory allocated–memory predicted)*Execution Time, 0 otherwise

Table 2. Observation-Level Indicators

The indicators above served as a baseline for model-level performance indicators (Table 3). All mode-level indicators were calculated, one per day (N=84), for each model configuration (a certain combination of treatments), and configurations were then compared using paired t-tests.

Indicator	Value
1. Impact: the ratio of observations that received predictions	Σ (Predicted) / [#Count]
2. Accuracy: the ratio of observations with sufficient allocation	Σ (Accurate) / [#Count]
3. Saving: the proportion of actual utilization that was saved	Σ (Actual Saving) / Σ (Utilization)
4. Realization: saving out of the maximum potential saving	Σ (Actual Saving) / Σ (Potential Saving)

Table 3. Model-Level Indicators

4.1 Evaluation Results

Treatment 1 – Use of Sliding Windows (SW): The batch mode (SW=0) was testing by processing the entire training set per day, while the sliding-window mode (SW=1) was tested by using hourly batches. As expected, the SW outperformed the batch in all indicators (Table 4). The improvement in accuracy was relatively minor, as both models were required to adhere to accuracy level of at least 0.95. However, in all other indicators the results showed strong and significant improvements.

Indicator	Batch Mode (SW=0)	Sliding Windows Mode (SW=1)	Mean Difference	Std. Error	Significance
1. Impact	0.299	0.742	0.443	0.021	< 0.001
2. Accuracy	0.931	0.951	0.020	0.003	< 0.001
3. Saving	0.006	0.057	0.051	0.002	< 0.001
4. Realization	0.021	0.193	0.171	0.008	< 0.001

Table 4. Treatment 1 Results

Treatment 2 - Feature Selection (FS): The DDS-based FS (FS=1), developed in this study, was tested versus the ensemble algorithm (FS=2), proposed by (Tuv et al., 2009), and versus no feature selection (FS=0). For batch mode (SW=0), the DDS-based FS had stronger impact than the two other options, slightly but significant higher saving and realization, but slightly lower accuracy (Table 5).

Indicator	All (FS=0)	DDS (FS=1)	Ens. (FS=2)	Mean Diff. (1 / 0)	Std. Error (1 / 0)	Sig. (1 / 0)	Mean Diff. (1 / 2)	Std. Error (1 / 2)	Sig. (1 / 2)
1. Impact	0.299	0.531	0.288	0.232	0.022	< 0.001	0.243	0.024	< 0.001
2. Accuracy	0.931	0.928	0.934	-0.003	0.001	< 0.001	-0.006	0.001	< 0.001
3. Saving	0.006	0.008	0.006	0.002	0.001	< 0.001	0.002	0.000	< 0.001
4. Realization	0.021	0.026	0.020	0.005	0.001	< 0.001	0.006	0.002	< 0.001

Table 5. Treatment 2 Results, Batch Mode (SW = 0)

With sliding-windows turn on (SW=1), DDS-based FS outperformed the "all features" option (FS=0), where the improvement was insignificant for accuracy, and statistically significant for all other indicators. The DDS-based FS had stronger impact than the ensemble-based FS, slightly lower accuracy, and slightly higher (statistically insignificant) saving and realization (Table 6).

Indicator	All (FS=0)	DDS (FS=1)	Ens. (FS=2)	Mean Diff. (1 / 0)	Std. Error (1 / 0)	Sig. (1 / 0)	Mean Diff. (1 / 2)	Std. Error (1 / 2)	Sig. (1 / 2)
1. Impact	0.741	0.892	0.788	0.151	0.009	<0.0001	0.104	0.009	<0.0001
2. Accuracy	0.950	0.951	0.954	0.001	0.001	0.493	-0.003	0.001	0.016
3. Saving	0.056	0.063	0.052	0.007	0.001	<0.0001	0.004	0.001	0.326
4. Realization	0.192	0.211	0.209	0.019	0.003	<0.0001	0.002	0.004	0.510

Table 6. Treatment 2 Results, Sliding-Windows Mode (SW = 1)

Treatment 3 - Concept Drift (CD) Management: The concept-drift management mechanism (CD=1), which was developed in this study, was tested against the use of a fixed-size window (CD=0). First, the comparison was done for the batch mode (SW=0) and with no feature selection (FS=0). The result show statistically significant improvements in all indicators (Table 7).

Indicator	Fixed Window (CD=0)	Adjustable Window (CD=1)	Mean Difference	Std. Error	Significance
1. Impact	0.299	0.482	0.183	0.022	< 0.001
2. Accuracy	0.931	0.937	0.006	0.002	0.010
3. Saving	0.006	0.019	0.013	0.001	< 0.001
4. Realization	0.021	0.068	0.046	0.005	< 0.001

Table 7. Treatment 3 Results, Batch Mode (SW=0), No Feature Selection (FS=0)

With sliding-windows (SW=1) and DDS-based feature selection (FS=1) applied, the effect on performance, in this case, was less noticeable compared to the two other treatments. The impact, saving, and realization show slight improvement, while the accuracy slightly declines (Table 8).

Indicator	Fixed Window (CD=0)	Adjustable Window (CD=1)	Mean Difference	Std. Error	Significance
1. Impact	0.892	0.894	0.002	0.006	0.002
2. Accuracy	0.951	0.941	-0.010	0.002	< 0.001
3. Saving	0.063	0.064	0.001	0.0005	0.066
4. Realization	0.211	0.214	0.003	0.002	0.019

Table 8. Treatment 3 Results, Sliding-Windows Mode (SW=1), DDS Feature Selection (FS=1)

	Impact		Accuracy		Saving		Realization	
Indicator	**Chi-2**	**Sig.**	**Chi-2**	**Sig.**	**Chi-2**	**Sig.**	**Chi-2**	**Sig.**
Intercept	872	< 0.001	430706	< 0.001	337	< 0.001	342	< 0.001
SW	461	< 0.001	59	< 0.001	403	< 0.001	453	< 0.001
FS	217	< 0.001	17	< 0.001	20	< 0.001	9	0.003
CD	83	< 0.001	3	0.094	80	< 0.001	79	< 0.001
SW*FS	6	0.012	1	0.399	49	< 0.001	42	< 0.001
SW*CD	74	< 0.001	21	< 0.001	131	< 0.001	121	< 0.001
FS*CD	20	< 0.001	17	< 0.001	28	< 0.001	29	< 0.001
SW*FS*CD	15	< 0.001	2	0.152	12	0.001	15	< 0.001

Table 9. Interactions Evaluation Results

All interactions were also tested using a GEE (General Estimating Equations) model, with auto-regressive correlations, and confidence interval of 0.95. The results (Table 9) show that most interactions are significant, with the exception of some interactions in the accuracy model. With respect to impact, saving, and realization – the influence of SW appears to be the strongest. FS had relatively strong effect on the impact indicator, while the interaction between SW and CD appears to have relatively strong effect on saving and realization.

Overall, the evaluation results are positive and highlight the potential contribution of the MSW and the associated treatments within the evaluated context. The first treatment reflects the core MSW contribution – the use of multiple sliding windows, rather than running the entire dataset in a batch. Not surprisingly, it had the strongest impact on all indicators. The two other treatments were also shown to have some added contribution. The effects on the impact indicators were shown to be steep and significant – meaning that the MSW and the associated treatments increased the number of instances that could be classified. The saving and the realization of potential improved significantly, in most cases, meaning that with the MSW the estimations are much closer to the actual use. The results for accuracy were inconsistent – in some cases it slightly improved and in other it slightly declined. This can be attributed to the evaluation setup, which enforced relatively high accuracy levels to begin with, what had possibly made the effects less noticeable.

Following those encouraging results, the MSW was adopted by Intel and incorporated into its server-cloud environment. A substantial impact was noted almost immediately – an increase of ~9% in the number of chip-design tasks that can be executed and validated with the same amount of computational resources. At Intel's large and complex servers environment, such increase in productivity translate to savings of millions of USD annually, in terms of cutting hardware investments, cut operation and maintenance costs, and reduce power consumption. Further, the ability to speed-up validation cycles has major positive impact on designing new chips and on Intel's ability to deliver high-quality products and speed up time-to-market (TTM). Following this success and contribution, the project was granted with Intel's IT-excellence award.

5 Conclusions

The growing pace of data accumulation and the demand for real-time BI capabilities raise the attention to algorithmic DSM solutions. This study offers contribution to that end by developing and evaluating the Multiple Sliding Windows (MSW) algorithm for continuous prediction. The MSW, in its current form, fits prediction problems with continuous output and discrete inputs – a configuration that has not been discussed much in DSM literature. However, due to its high modularity, the MSW may apply with some adaptations to a broader range of configurations. The MSW embeds a few novel concepts, besides the adoption of others that were previously introduced in DSM literature – the use of multiple sliding windows, the DDS criterion for feature selection, and the mechanism for adapting window size toward improvement of concept-drift detection. The algorithm in entire and the novel concepts in particular were tested with a large real-world dataset that reflects dynamic resource allocation at Intel. The positive evaluation results show that the MSW meets the required DSM characteristics, as presented earlier in the background section:

- **Performance:** The MSW algorithm offers a few performance-enhancing mechanisms. First, its modular nature permits parallel execution. Second, discarding rare occurrences and applying profile criteria reduces the number of windows significantly toward faster computation. Third, the use of exponential histograms reduces the amount of data that has to be processed, without much performance penalty.
- **Quality:** The evaluation at Intel demonstrated the ability to produce high-quality predictions.

- **Adaptability:** The concept-drift management mechanism enhances the ability to adapt to both abrupt and gradual changes in data stream characteristics, by adjusting the window size.
- **Availability:** The MSW output is stored in accessible database tables, and can be made available at any time with the most up-to-date prediction per profile.

The recognition of the potential MSW benefits has led to its successful integration into Intel's large cloud-server environment. Notably, beyond the operational and financial benefits, the reduction in the use of computational resources and power consumption makes some important contribution in terms of lessening environmental damages. With the growing attention to environmental responsibility and green IT (Loos, 2011), any contribution in that direction is obviously welcome.

Obviously, the MSW has some limitations that relate to mathematical-modeling assumptions and configuration constraints. Table 10 highlights a few examples for such MSW assumptions and configuration decisions, and proposed possible alternatives that require further investigation.

Current MSW Approach	**Alternatives**
Profiles, and accordingly predictions, are generated one per each feature set occurrence.	Profile arranges in a dynamic tree-structure, with automatic splitting and pruning of nodes
Using Exponential Histograms for maintaining summary statistics , toward data compression	Applying other methods for maintaining summary statistics – e.g., bucket-based configurations (Ijagadish et al., 1998; Babcock et al., 2002), or applying no compression
Using the Heoffding's bound for concept-drift detection	Applying concept-drift ensembles, such as the concept proposed in (Tsymbal et al., 2008)
Fre-setting fixed minimum size and minimum support parameters	Adjusting the parameter to follow changes in statistical characteristics of the stream

Table 10. Alternative MSW Modelling and Configuration Approaches

Finally, it is important to note that this study evaluated the MSW algorithm in the specific business context, and for a specific DRA problem. The encouraging results and the successful implementation highlight the potential benefits; however, we suggest that the MSW can be relevant and beneficial within a broader range of business domains and DSM problems – what may direct future extensions to our research. With the growing demand for real-time BI capabilities, which often mandate online processing of fast-accumulating data streams – algorithmic DSM solutions, such as the MSW described in this study, become a necessity and may potentially offer important contributions.

Acknowledgement

We would like to thank Intel's Memory Prediction Project team for their support and contribution to success of our research.

References

Akaike, H. (1974). A new look at the statistical model identification. IEEE Transactions on Automatic Control, 19(6), 716–723.

Azvine, B., Cui, Z., and Majeed, B. (2006). Real time business intelligence for the adaptive enterprise. E-Commerce Technology, 2006. In Proceeding of the 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, San Francisco, CA, USA.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA.

Bifet, A., and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the SIAM International Conference on Data Mining, Minneapolis, MN, USA

Bifet, A., and Gavaldà, R. (2009). Adaptive learning from evolving data streams. In Advances in Intelligent Data Analysis VIII. Springer, Berlin, Germany.

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., and Jansen, T. (2010). MOA: Massive Online Analysis, a framework for stream classification and clustering. Journal of Machine Learning Research, 11(1), 1601-1604.

Chen, K., Kurgan, L., and Ruan, J. (2006). Optimization of the sliding window size for protein structure prediction. In Proceedings of IEEE CIBCB 2006 Symposium on Computational Intelligence and Bioinformatics and Computational Biology, Toronto, Canada.

Chen, H., Chiang, R.H.L., Storey, V. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. MIS Quarterly, 36(4), 1165-1188

Cohen, L., Avrahami, G., Last, M., and Kandel, A. (2008). Info-fuzzy algorithms for mining dynamic data streams. Applied Soft Computing, 8(4), 1283-1294.

Datar, M., Gionis, A., Indyk, P., and Motwani , R. (2002). Maintaining stream statistics over sliding windows. In Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA.

Domingos, P., and Hulten, G. (2000). Mining high-speed data streams. In Proceedings of the 6th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining, New York, NY, USA.

Domingos, P., and Hulten, G. (2001). Catching up with the data: Research issues in mining data streams. In Proceeding of the Workshop on Research Issues Data Mining and Knowledge Discovery, Santa Barbara, CA,USA

Fares, M. A., Elena, S. F., Otriz, J., Moya, A., and Barrio, E. (2002). A sliding window based method to detect selective constraints in protein-coding genes and its application to RNA viruses. Journal of Molecular Evolution, 55(5), 509-521.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: a review. ACM SIGMOD Record, 34(2), 18-26.

Gama, J., and Castillo, G. (2006). Learning with local drift detection. Advanced Data Mining and Applications, Springer, Berlin, Germany.

Gama, J., and Rodrigues, P. P. (2007). Processing data streams. Learning from Data Streams: Processing Techniques in Sensor Networks, Springer, Berlin, Germany.

Guyon, I., and Elisseeff, A. (2003). An introduction to variable and feature selection. The Journal of Machine Learning Research, 3 (1), 1157-1182.

Hoeffding, W. (1963, March). Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58 (30), 13-30.

Hu, Y., Wong, J., Iszlai, G., and Litoiu, M. (2009). Resource provisioning for cloud computing. In A. W. Patrick Martin (Ed.), In Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research (CASCON '09), Toronto, Canada.

Hua-Fu, L., and Suh-Yin , L. (2009). Mining frequent itemsets over data streams using efficient window sliding techniques. Expert Systems with Applications, 36 (2), 1466-1477.

Hughes, G. F. (1968). On the mean accuracy of statistical pattern recognizers. IEEE Transactions on Information Theory, 14(1), 55-63.

Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA.

IJagadish, H., Poosala, V., Koudas, N., Sevcik, K., Muthukrishnan, S., and Suel, T. (1998). Optimal histograms with quality guarantees. In Proceedings of 24th Annual International Conference on Very Large Data Bases, New-York, NY, USA

Ikonomovska, E., and Gama, J. (2008). Learning model trees from data stream. Discovery Science, Springer, Berlin, Germany.

Ikonomovska, I., João, G., Raquel, S., and Dejan, G. (2009). Regression trees from data streams with drift detection. Discovery Science, Springer, Berlin, Germany.

Last, M. (2002). Online classification of non-stationary data streams. Intelligent Data Analysis, 6 (2), 129-147.

Loos, P., Nebel, W., Gomez, J.M., Hasan, H., Watson, R.T., Brocke, J.V., Seidel, S., and Recker, J. (2011). Green IT: A Matter of Business and Information Systems Engineering?. Business and Information Systems Engineering, 3(4), 245-252

Sahay, B., and Ranjan, J. (2008). Real time business intelligence in supply chain analytics. Information Management and Computer Security, 16 (1), 28-48.

Tsymbal, A., Pechenizkiy, M., Cunningham, P., and Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. Information Fusion, 9(1), 56-68.

Tuv, E., Borisov, A., Runger, G., and Torkkola, K. (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. Journal of Machine Learning Research, 10 (1), 1341-1366.

Watson, H.J. (2009). Tutorial: Business Intelligence – Past, Present, and Future. Communications of the AIS, 25(39), 488-510.

Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In Proceedings of the 4th International Conference on Autonomic Computing, Jacksonville, FL, USA.

Zliobaite, I. (2009). Learning under concept drift: an overview. Vilnius University, Faculty of Mathematics and Informatics, Vilnius, Lithuania.