Incremental Quantiles Estimators for Tracking Multiple Quantiles

Hugo Lewi Hammer and Anis Yazidi

Department of Computer Science
Oslo and Akershus University College of Applied Sciences

Abstract. In this paper, we investigate the problem of estimating multiple quantiles when samples are received online (data stream). We assume that we are dealing with a dynamical system, i.e. the distribution of the samples from the data stream changes with time. A major challenge arises when simultaneously maintaining multiple quantile estimates using *incremental* type of estimators. In fact, a naive implementation where multiple incremental quantile estimators are updated in *isolation* might lead to violation monotone property of quantiles, i.e, an estimate of a lower target quantile might erroneously overpass that of a higher one. Surprisingly, the related work on countering those violations is extremely sparse [3, 1] and almost absent.
Our work tries to fill this literature gap by proposing two solutions to the problem that build on the deterministic update based multiplicative incremental quantile estimator (DUMIQE) recently proposed by Yazidi and Hammer [5], which was shown to be the most efficient incremental quantile estimator in the literature.
Experimental results show that the modified DUMIQE methods perform very well and have a superior performance to the DUMIQE. Moreover, our proposed methods satisfy the monotone property of quantiles. The methods outperform the state of the art multiple incremental quantile estimator of Cao et al. [3, 1].

1 Introduction

Quantiles are key indicators for monitoring the performance of a system in an online fashion. For instance, system administrators are interested in monitoring the 95% response time of a web-server so that to hold it under a certain threshold. Quantile tracking is also useful for detecting abnormal events and in intrusion detection systems in general.

In the context of large data streams, quantile estimators have a major computational and memory complexity disadvantage as even linear computational complexity is not affordable. Several algorithms have been proposed to deal with this challenges.

The most efficient and lightweight quantile estimator reported in the literature are the so-called incremental estimators [4, 2]. An incremental quantile estimator by definition resorts to only the last data sample in order to update the

current estimate. The informed reader will observe that the exponential moving average is a type of incremental estimator but rather for the average and not the quantile where the current estimate is a weighted average of the old estimate and the last observation.

From a practical point of view it is often useful to estimate many quantiles of the dynamic data stream. A simple approach is to estimate the different quantiles independent of each other by running incremental estimators in parallel, one for each quantile to be estimated. Unfortunately, such an approach often leads to unrealistic estimates as the monotone property of quantiles might be violated, e.g. that the estimate of a lower quantile can at some time instants overpasses the estimate of a higher quantile. As a way of illustration, we know that 50% quantile can not overpass 70% quantile due to the monotone property of the cumulative distribution and consequently the respective estimates of both quantities should maintain this monotone property too.

Please note that the latter major disadvantage is inherent in any incremental quantile estimator without exception since, by design, they do not enforce the monotone property.

To the best of our knowledge, Cao et al. [3] is the only solution found in the literature. The main idea is to rather resort to linear interpolation to yield an increasing approximate of the cumulative function. Consequently, updated quantile estimates are obtained from the approximate cumulative distribution. Unfortunately, such operation is usually expensive.

In this paper, we tackle the problem of estimating multiple quantiles from a dynamically changing data stream. To achieve this, we extend the DUMIQE method proposed by Yazidi and Hammer [5]. The choice of DUMIQE method as a core for our current work is deliberate since it was shown to be the most performant method in the literature. In this paper, we thus focus on extending the DUMIQE method in order to accommodate the case of tracking multiple quantiles. DUMIQE presents an efficient extension of randomized update based multiplicative incremental quantile estimator (RUMIQE) proposed in [6].

It is worth mentioning that the algorithms presented in this paper are not limited to the incremental quantile estimator developed in [5] and their essence can be easily transferred and generalized for other types of incremental estimators.

The reminder of the article is organized as follows. In Section 2, we present two different algorithms for designing parallel incremental quantile estimates satisfying the monotone property. Section 3 presents some thorough experimental results where we catalogue the performance of the two proposed algorithms and compare them to the state-of-art. Section 4 concludes the article.

2 Estimation of multiple quantiles

Let X_n denote a stochastic variable denoting the possible outcomes from the data stream at time n and let x_n denote a random sample of X_n. We assume that X_n is distributed according to some distribution $f_n(x)$ that varies dynamically

with time n. Further let $Q_n(q)$ denote the quantile associated with probability q at time n, i.e. $Q_n(q) = F_X(q) = P(X_n \leq q)$.

In this paper we focus on simultaneously estimating the quantiles for K different probabilities $q_1, q_2, \ldots, q_K$ at each time step. We assume an increasing order of the probabilities, i.e. $q_1 < q_2 < \cdots < q_K$. The straight forward approach to estimate the quantiles would be to simply run DUMIQE (or some other online estimation procedure) for every target quantile

$$\begin{aligned} \widehat{Q_{n+1}}(q_k) &\leftarrow (1 + \lambda q_k)\widehat{Q_n}(q_k) && \text{if } \widehat{Q_n}(q_k) < x_n \\ \widehat{Q_{n+1}}(q_k) &\leftarrow (1 - \lambda(1 - q_k))\widehat{Q_n}(q_k) && \text{if } \widehat{Q_n}(q_k) \geq x_n \end{aligned} \tag{1}$$

for $k = 1, 2, \ldots, K$. Unfortunately, this may lead to a violation of the monotone property of quantiles, i.e. we may not satisfy

$$\widehat{Q_{n+1}}(q_1) \leq \widehat{Q_{n+1}}(q_2) \leq \ldots \leq \widehat{Q_{n+1}}(q_K) \tag{2}$$

In order to further shed the light on the eventuality of violating the monotone property, we provide a simple example. Assume at time n that the monotone property is satisfied and that the sample x_n gets a value between $\widehat{Q_n}(q_k)$ and $\widehat{Q_n}(q_{k+1})$, i.e.

$$\widehat{Q_n}(q_1) \leq \cdots \leq \widehat{Q_n}(q_k) < x_n < \widehat{Q_n}(q_{k+1}) \leq \cdots \leq \widehat{Q_n}(q_K) \tag{3}$$

Then according to (1) the estimates are updated as follows

$$\begin{aligned} \widehat{Q_{n+1}}(q_j) &\leftarrow (1 + \lambda q_j)\widehat{Q_n}(q_j) && \text{for } j = 1, 2, \ldots, k \\ \widehat{Q_{n+1}}(q_j) &\leftarrow (1 - \lambda(1 - q_j))\widehat{Q_n}(q_j) && \text{for } j = k+1, \ldots, K \end{aligned} \tag{4}$$

which means that the estimates are increased for the quantiles with an estimate below x_n and decreased for the estimates above x_n. Consequently, the monotone property may be violated in this case. Next we present present two modifications of the DUMIQE such that the monotone property will be satisfied.

2.1 Sorting Based Approach

The first approach we propose in this paper is simple and intuitive. It is based on sorting the quantile estimates. Every time we receive a new sample x_n the procedure consists of the three following steps:

1. Update the quantile estimates according to (1) and get the estimates $\widehat{Q_{n+1}}(q_k), k = 1, 2, \ldots, K$
2. Sort the updated estimates and denote them $\widetilde{Q_{n+1}}(q_k), k = 1, 2, \ldots, K$. The estimates after sorting naturally will satisfy the monotone property, but will also contain less (or equal) estimation error than the original estimates. In other words, this is a win-win solution, but at the computational cost of sorting the quantiles, $O(K \log(K))$.

3. We have two alternatives for update at the subsequent time instant $n+1$. Upon receiving the sample (x_{n+1}), we may update according to Equation (1) using
   (a) the estimates from before the sorting, i.e. $\widehat{Q_{n+1}}(q_k), k = 1, 2, \ldots, K$
   (b) or the estimates after the sorting, i.e. $\widetilde{Q_{n+1}}(q_k), k = 1, 2, \ldots, K$

Alternative (a) means that we do *not* feed the information from the sorting back into the estimation process, while in (b) we do. Using alternative (a) means that we only use sorting to "repair" the estimates from the original estimation process based on Equation (1). The overall computational complexity of this approach thus is $O(K \log(K))$ in every iteration.

2.2 Adjusting the size λ

The next strategy is based on reducing the value of λ in a given iteration if the updates result in monotone property violation. Assume that we are in the situation where the sample x_n gets a value between $\widehat{Q_n}(q_k)$ and $\widehat{Q_n}(q_{k+1})$ as given by (3). The first observation is that after the update, the monotone property always will be satisfied on each side of x_n, i.e.

$$\begin{aligned}&\widehat{Q_{n+1}}(q_1) \leq \widehat{Q_{n+1}}(q_2) \leq \cdots \leq \widehat{Q_{n+1}}(q_k) \text{ and}\\ &\widehat{Q_{n+1}}(q_{k+1}) \leq \widehat{Q_{n+1}}(q_{k+2}) \leq \cdots \leq \widehat{Q_{n+1}}(q_K)\end{aligned}$$

This follows from Equation (4). Therefore a sufficient criterion to satisfy the monotone property is to make sure to use a sufficiently small λ such that permits to satisfy the following inequality: $\widehat{Q_{n+1}}(q_k) \leq \widehat{Q_{n+1}}(q_{k+1})$. We derive such a λ, denoted $\widetilde{\lambda}$, by making sure that the distance between $\widehat{Q_{n+1}}(q_k)$ and $\widehat{Q_{n+1}}(q_{k+1})$ is some portion, α, of the distance from the previous iteration, i.e.

$$\begin{aligned}\widehat{Q_{n+1}}(q_{k+1}) - \widehat{Q_{n+1}}(q_k) &= \alpha\left(\widehat{Q_n}(q_{k+1}) - \widehat{Q_n}(q_k)\right)\\ (1 - \widetilde{\lambda}(1 - q_{k+1}))\widehat{Q_n}(q_{k+1}) - (1 + \widetilde{\lambda}q_k)\widehat{Q_n}(q_k) &= \alpha\left(\widehat{Q_n}(q_{k+1}) - \widehat{Q_n}(q_k)\right)\end{aligned} \tag{5}$$

with $\alpha \in [0, 1)$. Solving (5) with respect to $\widetilde{\lambda}$ we get

$$\begin{aligned}\widetilde{\lambda} &= (1-\alpha)\frac{\widehat{Q_n}(q_{k+1}) - \widehat{Q_n}(q_k)}{(1 - q_{k+1})\widehat{Q_n}(q_{k+1}) + q_k\widehat{Q_n}(q_k)}\\ &= (1-\alpha)H\Big(\widehat{Q_n}(q_k), \widehat{Q_n}(q_{k+1})\Big)\end{aligned} \tag{6}$$

We substitute λ with $\widetilde{\lambda}$ in (1) if using the originally chosen λ results into the violation of the monotone property. We then obtain the following updates

$$\widehat{Q_{n+1}}(q_k) \leftarrow (1+\lambda q_k)\widehat{Q_n}(q_k) \quad \text{if } \widehat{Q_n}(q_k) < x_n \cap \widehat{Q_n}(q_{k+1}) < x_n \tag{7}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_k) \leftarrow{}& (1+\lambda q_k)\widehat{Q_n}(q_k) \\ & \text{if } \widehat{Q_n}(q_k) < x_n \cap \widehat{Q_n}(q_{k+1}) \geq x_n \cap \lambda < H\Big(\widehat{Q_n}(q_k), \widehat{Q_n}(q_{k+1})\Big)\end{aligned} \tag{8}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_k) \leftarrow{}& \Big(1+(1-\alpha)H\Big(\widehat{Q_n}(q_k), \widehat{Q_n}(q_{k+1})\Big)\, q_k\Big)\, \widehat{Q_n}(q_k) \\ & \text{if } \widehat{Q_n}(q_k) < x_n \cap \widehat{Q_n}(q_{k+1}) \geq x_n \cap \lambda > H\Big(\widehat{Q_n}(q_k), \widehat{Q_n}(q_{k+1})\Big)\end{aligned} \tag{9}$$

$$\widehat{Q_{n+1}}(q_k) \leftarrow (1-\lambda(1-q_k))\widehat{Q_n}(q_k) \quad \text{if } \widehat{Q_n}(q_k) \geq x_n \cap \widehat{Q_n}(q_{k-1}) \geq x_n \tag{10}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_k) \leftarrow{}& (1-\lambda(1-q_k))\widehat{Q_n}(q_k) \\ & \text{if } \widehat{Q_n}(q_k) \geq x_n \cap \widehat{Q_n}(q_{k-1}) < x_n \cap \lambda < H\Big(\widehat{Q_n}(q_{k-1}), \widehat{Q_n}(q_k)\Big)\end{aligned} \tag{11}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_k) \leftarrow{}& \Big(1-(1-\alpha)H\Big(\widehat{Q_n}(q_{k-1}), \widehat{Q_n}(q_k)\Big)\,(1-q_k)\Big)\, \widehat{Q_n}(q_k) \\ & \text{if } \widehat{Q_n}(q_k) \geq x_n \cap \widehat{Q_n}(q_{k-1}) < x_n \cap \lambda > H\Big(\widehat{Q_n}(q_{k-1}), \widehat{Q_n}(q_k)\Big)\end{aligned} \tag{12}$$

for $k = 2, \ldots, K-1$. The special cases for $k = 1$ and $k = K$ are shown below. Equation (7) shows the case when x_n takes a value above $\widehat{Q_n}(q_{k+1})$ and therefore is no risk of violation of the monotone property. The update therefore is as in (1). Equation (8) shows the case when x_n takes a value between $\widehat{Q_n}(q_{k-1})$ and $\widehat{Q_n}(q_k)$ and we may potentially get a monotone violation. But since $\lambda < H\Big(\widehat{Q_n}(q_k), \widehat{Q_n}(q_{k+1})\Big)$ we are able to maintain the monotone property using λ. Thus, this update is also as in (1). Equation (9) shows the case when x_n takes a value between $\widehat{Q_n}(q_{k-1})$ and $\widehat{Q_n}(q_k)$ and $\lambda > H\Big(\widehat{Q_n}(q_k), \widehat{Q_n}(q_{k+1})\Big)$ and therefore we get a monotone violation using λ and we need to use $\widetilde{\lambda}$ from (6) instead of λ in this update. Equations (10) to (12) show the similar updates when x_n takes a value below $\widehat{Q_n}(q_{k+1})$.

For the smallest and largest quantile estimates, we only get potential monotone violations upwards and downwards, respectively, resulting in the following

updates

$$\widehat{Q_{n+1}}(q_1) \leftarrow (1+\lambda q_1)\widehat{Q_n}(q_1) \quad \text{if } \widehat{Q_n}(q_1) < x_n \cap \widehat{Q_n}(q_2) < x_n \tag{13}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_1) \leftarrow (1+\lambda q_1)\widehat{Q_n}(q_1) \\ \text{if } \widehat{Q_n}(q_1) < x_n \cap \widehat{Q_n}(q_2) \geq x_n \cap \lambda < H\Big(\widehat{Q_n}(q_1), \widehat{Q_n}(q_2)\Big)\end{aligned} \tag{14}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_1) \leftarrow \Big(1+(1-\alpha)H\Big(\widehat{Q_n}(q_1), \widehat{Q_n}(q_2)\Big)\, q_1\Big)\, \widehat{Q_n}(q_1) \\ \text{if } \widehat{Q_n}(q_1) < x_n \cap \widehat{Q_n}(q_2) \geq x_n \cap \lambda > H\Big(\widehat{Q_n}(q_1), \widehat{Q_n}(q_2)\Big)\end{aligned} \tag{15}$$

$$\widehat{Q_{n+1}}(q_1) \leftarrow (1-\lambda(1-q_1))\widehat{Q_n}(q_1) \quad \text{if } \widehat{Q_n}(q_1) \geq x_n \tag{16}$$

and

$$\widehat{Q_{n+1}}(q_K) \leftarrow (1+\lambda q_K)\widehat{Q_n}(q_K) \quad \text{if } \widehat{Q_n}(q_K) < x_n \tag{17}$$

$$\widehat{Q_{n+1}}(q_K) \leftarrow (1-\lambda(1-q_K))\widehat{Q_n}(q_K) \quad \text{if } \widehat{Q_n}(q_K) \geq x_n \cap \widehat{Q_n}(q_{K-1}) \geq x_n \tag{18}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_K) \leftarrow (1-\lambda(1-q_K))\widehat{Q_n}(q_K) \\ \text{if } \widehat{Q_n}(q_K) \geq x_n \cap \widehat{Q_n}(q_{K-1}) < x_n \cap \lambda < H\Big(\widehat{Q_n}(q_{K-1}), \widehat{Q_n}(q_K)\Big)\end{aligned} \tag{19}$$

$$\begin{aligned}\widehat{Q_{n+1}}(q_K) \leftarrow \Big(1-(1-\alpha)H\Big(\widehat{Q_n}(q_{K-1}), \widehat{Q_n}(q_K)\Big)\,(1-q_K)\Big)\, \widehat{Q_n}(q_K) \\ \text{if } \widehat{Q_n}(q_K) \geq x_n \cap \widehat{Q_n}(q_{K-1}) < x_n \cap \lambda > H\Big(\widehat{Q_n}(q_{K-1}), \widehat{Q_n}(q_K)\Big)\end{aligned} \tag{20}$$

By estimating all the quantiles using the rules in (7) − (12), we ensure that the monotone property in (2) is satisfied in every iteration $n = 1, 2, 3, \ldots$.

The most expensive part of this algorithm is to find the k in (3) that can be computed in $O(\log(K))$ operations which is less expensive than updating every quantile ,i.e, $O(K)$. The overall computational complexity of this approach thus is $O(K)$ in every iteration.

3 Experiments

It is possible to prove that the DUMIQE approach in (1) converges to the true quantiles [5]. Unfortunately, it is hard (or impossible) to prove convergence for the methods described above. As described above the running estimation process is then simply the DUMIQE in (1). Since the theoretical proofs of the methods above are intrinsically hard, we instead resort to simulations to document the effectiveness of the approaches.

The experiments focus on the methods ability to track quantile estimates when the distribution of the data stream changes with time. We consider the

two different cases were we assume that the data correspond to outcomes from a normal distribution. Furthermore, we assume that the expectation of the distribution varies with time

$$\mu_n = a \sin\left(\frac{2\pi}{T} n\right), \;\; n = 1, 2, 3, \ldots$$

which is the sinus function with period T. Moreover, we assume that the standard deviation of the distribution do not vary with time but is equal to one.

Now, we turn to conducting a thorough analysis of how well the proposed methods in Section 2 estimate quantiles of data streams. We estimated quantiles of both the normally and chi^2 distributed data streams above using two different periods, namely $T = 800$ (rapid variation) and $T = 8000$ (slow variation), i.e. in total four different data streams. In addition, for each of the four data streams we estimated quantiles that were centered around the median or in the tail of the distribution, i.e. eight different cases. We chose the quantiles close enough to get a fair amount of monotone property violations. Naturally, if we choose the quantiles far from each other we will rarely or never get any violations. In greater details, we estimated to following quantiles for the different cases.

- For the normal distribution and the quantiles around the median, we estimated the quantiles related to the following probabilities $q_k = \Phi(-0.8 + 0.2(k-1)), \;\; k = 1, 2, \ldots, 9$ where $\Phi(\cdot)$ refers to the cumulative distribution function of the standard normal distribution. Recall that in dynamical systems, as in these experiments, the value of a quantile related to a specific probability varies with time.
- For the normal distribution and the quantiles in the tail of the distribution, we use $q_k = \Phi(0.8 + 0.2(k-1)), \;\; k = 1, 2, \ldots, 9$.

The probabilities related to quantiles in the median and around the tail of the distribution are centered around the probabilities 0.5 and 0.95, respectively. The choices above resulted in a monotone property violation in about every third iteration using a typical value $\lambda = 0.05$ in (1).

To measure estimation error, we use the average of the root mean squares error (RMSE) for each quantile

$$RMSE = \frac{1}{K} \sum_{k=1}^{K} \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left(Q_n(q_k) - \widehat{Q_n}(q_k) \right)^2}$$

where N is the total number of samples in the data stream. We investigate the estimation error for a large set of different values of the parameter λ. In the experiments we used $N = 10^7$ which efficiently removed any Monte Carlo errors in the experimental results.

The results for the normal are shown in Figure 1. In the figure the abbreviations SORT, PREV refer to the estimation approaches presented in Sections 2.1 and 2.2.For the sorting based approach in Section 2.1, bring = TRUE means that we fed the sorted quantiles back into the estimation procedure. were fed

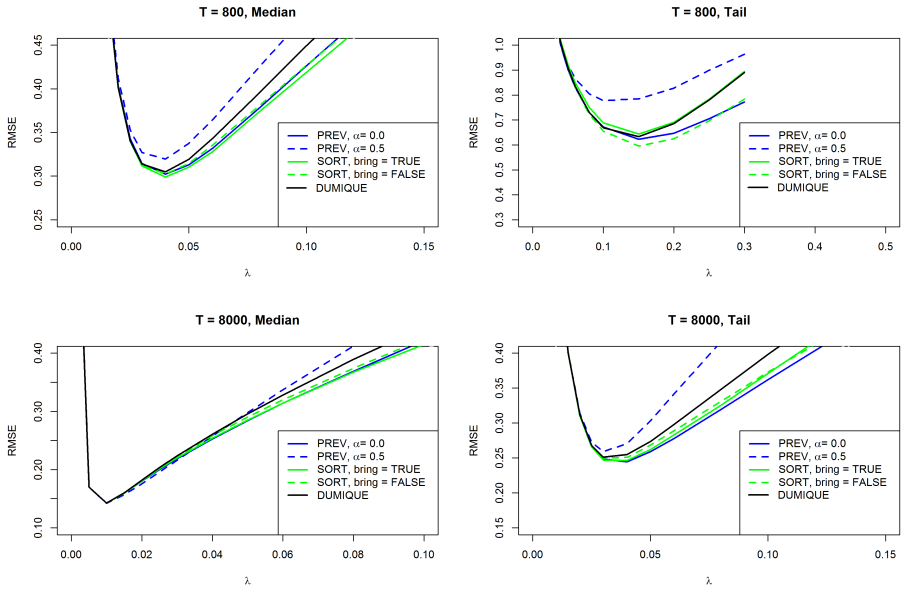

Fig. 1. Estimation error for data from the normal distribution.

back in to the estimation procedure. DUMIQE refers to updating the quantiles using (1) and ignoring that the monotone property may get violated.

For all the estimation methods, we observe that the estimation error increases when the period decreases or when estimating further into the tail of the distribution. It seems also that feeding the updated estimates $\widetilde{Q}_n(q_k)$ back into the estimation process further improves the estimation compared to not feeding them in. For the approach based on adjusting the size of λ in Section 2.2, it seems like using $\alpha = 0.5$ (making small updates) performs poor in all the experiments. Using $\alpha = 0$ we update as much as possible without violating the monotone property and performs about equally well to sorting the quantiles (Section 2.1). For the sorting approach, wether feeding the sorted estimates back in the estimation process or not has minimal effect on the estimation. An interesting observation is that almost all the approaches perform better than updating DUMIQE in isolation, i.e, without enforcing the monotone property. In other words, we are able to both satisfy the monotone property and improve estimation precision with a minimal extra computational costs.

For comparison we also tested the method in [3] for the eight estimation tasks described above. The latter method is the only method we have found in the literature that attempts to estimate multiple quantiles in a dynamical system. The method have two tuning parameters, a weight parameter similar to λ in the methods in this paper, and a parameter that controls the width of intervals to estimate the distribution of the data stream around a quantile. To achieve

as good results as possible we ran the method for a large set of values for the two parameters. The best estimation results are shown in Table 1. We remark

	$T = 800$, Median	$T = 800$, Tail	$T = 8000$, Median	$T = 8000$, Tail
Normal distribution	0.312	0.630	0.259	0.370
	$T = 800$, Median	$T = 800$, Tail	$T = 8000$, Median	$T = 8000$, Tail
χ^2 distribution	0.79	2.40	0.445	1.611

Table 1. Estimation error using the method in Cao et al. (2009) [3].

that for the normal distribution and $T = 800$ Cao et al. performs well. For the normal distribution and $T = 8000$, the methods in this paper outperforms Cao et al. (2009) [3]. For all the cases related to the χ^2 distribution, the methods in this paper outperforms Cao et al. (2009) [3] with a clear margin. Not only does the methods in this paper outperform Cao et al. (2009) [3], they are also far simpler to implement and only contain only one tuning parameters which makes it easier to tune the method to perform well. The experiments also showed that the methods in this paper are less sensitive to the choice of the tuning parameter compared to Cao et al. (2009) [3].

4 Closing remarks

In this paper, we have devised two methods that incrementally estimate multiple quantiles from a dynamic data stream while enjoying the ability to maintain the monotone property of the quantiles. Surprisingly, the work on this type of incremental quantile estimators is very sparse. The first proposed method is simple and is based on sorting the quantiles whenever the monotone property is violated. The second method suggests to adjust in a online manner the value of the parameter λ to ensure that the monotone property is never violated.

The results show that the suggested methods perform very well in estimating multiple quantiles. Most of the methods outperform the DUMIQE and at the same time satisfy the monotone property of quantiles. The method of adjusting the value of λ (Section 2.2) is of the same order of computational complexity as DUMIQE. In other words, we are able to both satisfy the monotone property and improve estimation precision at the cost of a minimal increase of the computational cost.

A research avenue worth investigating in the future is to deploy multiple parallel quantile estimators in order to improve the accuracy of tracking a single quantile estimate.

References

1. Tian Bu, Jin Cao, Aiyou Chen, and Li Li. Method and apparatus for incremental tracking of multiple quantiles, November 19 2013. US Patent 8,589,329.

2. Jin Cao, Li Li, Aiyou Chen, and Tian Bu. Tracking quantiles of network data streams with dynamic operations. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.
3. Jin Cao, Li Erran Li, Aiyou Chen, and Tian Bu. Incremental tracking of multiple quantiles for network monitoring in cellular networks. In *Proceedings of the 1st ACM workshop on Mobile internet through cellular networks*, pages 7–12. ACM, 2009.
4. Fei Chen, Diane Lambert, and José C Pinheiro. Incremental quantile estimation for massive tracking. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 516–522. ACM, 2000.
5. A. Yazidi and H. L. Hammer. Multiplicative Update Methods for Incremental Quantile Estimation. *Journal Article, Under review*, 2016.
6. Anis Yazidi and Hugo Hammer. Quantile estimation using the theory of stochastic learning. In *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems*, RACS, pages 7–14, New York, NY, USA, 2015. ACM.