Estimating the rate of Web Page Updates

Web/Data
Web mining, X-other-Web

Abstract

Estimating the rate of Web page updates helps in improving Web crawler's scheduling policy. But, most of the Web sources are autonomous and updated independently. Clients like Web crawlers are not aware of when and how often the sources change. Unlike other studies, we model the process of Web page updates as non-homogeneous Poisson process and focus on determining localized rate of updates. Then we discuss various rate estimators, showing experimentally how precise they are. This paper explores two classes of problems. Firstly we estimate localized rate of updates by dividing the given sequence of independent and inconsistent update points into consistent windows. From various experimental comparisons, the proposed Weibull estimator outperforms both Intuitive and Duane plot(another proposed estimator) in 91.5%(96.1%) of the whole windows for synthetic(real Web) datasets. Secondly, we predict future update points based on most recent window, where Duane plot estimator outperforms the rest in terms of repository freshness.

1 Introduction

Most of the information available on Web are autonomous and updated independently of the clients. Estimating the rate at which Web pages are updated, is important for Web clients like Web crawlers, Web caching etc. to improve their performance. Since the sources are updated on their own, such a client does not know exactly when and how often the sources change. Web crawlers need to revisit the sources repeatedly to keep track of the changes and maintain the repository uptodate. Further, if a source is revisited before it is changed, then it will be a wastage of resources such as CPU time, file system and network bandwidth for both client and server. We call such a visit as a *premature* visit. For a client like Web crawler with a collection of thousands of millions of such sources, it is important to schedule its visits optimally such that number of unobserved changes between the subsequent visits and the number of premature visits are minimum. Therefore, study of the changing behavior of web pages and estimating their changing rate can benefit in improving Web crawler's scheduling policy, Web page popularity measure, preprocessing the accessed schedules etc. This is the main motivation of this work. Another question that we try to answer in this paper is the localized rate of changes of the sources. For example, *At what rate was a Web page updated during the month of January last year? How does the rate of updates change during the month of January for the last five years?* This paper focuses mainly on estimating localized rate of updates rather than global update rates. Since the rate of updates of the sources may vary with time, localized estimation provides more detail, useful and accurate information.

Contrast to the previous studies [Cho and Garcia-Molina, 2003; Matloff, 2005], we model the process of Web page updates as *time-varying Poisson process*. It is because of the fact that sources are updated by its owner autonomously and independently whenever there are new developments and the rate of updates may vary with time. Therefore, it is more appropriate to assume the update process as a *time varying Poisson process* rather than *time homogeneous Poisson process*. The advantage of modeling the problem as *non-homogeneous Poisson process* is its wide range of flexibilities. It can be noted that *homogeneous Poisson process* is a special case of *non-homogeneous Poisson process*. A major difficulty in modeling the problem as non-homogeneous Poisson process is that it has infinitely many parameters. In particular, it is parameterized by its arrival-rate $\lambda(t)$, which varies with time. To simplify the problem, we restrict our attention to certain assumptions of $\lambda(t)$ i.e., *constant*, *increasing* or *decreasing*. Such assumption help in reducing the number of parameters. This assumption may not be correct over a long period of observations (say over years of page update records), but reasonable over appropriate subintervals (say over few weeks). Therefore, we divide the whole observation space into a sequence of windows of consistent behaviour (see section 4). We are interested more in *piecewise page updates rate*, therefore the above assumption is reasonable for our purpose.

1.1 Accessing the Web pages

One major problem in modeling the changing behavior of Web page is that, we do not have the complete information of the sequence of updates. Web pages are accessed repeatedly through normal activities such as periodic crawling of web pages. We can only determine, whether a page had changed, but not possible to determine the number of changes between

two subsequent accesses, which results in loss of information. The accuracy of the estimation of the rate of changes depends on how small the number of unobserved updates is.

Assuming that we access the Web pages periodically with a small interval τ, let $m(i)$ be the number of page updates between the $(i-1)^{th}$ and i^{th} accesses and $\{t_{(i,1)}, t_{(i,2)}, t_{(i,3)}, ..., t_{(i,m(i))}\}$ be the time sequence of corresponding updates. We have the following two ways of observing the changes:

-**Last date of change:** Some Web servers provide the information about when the page was last modified. It is not possible to obtain the complete update information between accesses. In such case, we only observe $t_{(i,m(i))}$ and $\{t_{(i,1)}, t_{(i,2)}, t_{(i,3)}, ..., t_{(i,m(i)-1)}\}$ are left unobserved. There is no way to obtain the unobserved information. Assuming that $m(i)$ is small, we ignore the unobserved updates and count only one update during $((i-1)\tau, i\tau]$ if changes occur and consider $t_{(i,m(i))}$ as the renewal point.

-**Existence of change:** Some Web servers do not provide such information *i.e., last date of change*. In such case, we can only determine whether the page has changed between accesses comparing with the old image of the page. We can not even determine $t_{(i,m(i))}$. It could have changed any number of times anywhere between $(i-1)^{th}$ access and i^{th} access, $i > 0$. If changes occur, we count it as one.

1.2 Formulating the Process

In the studies [Cho and Garcia-Molina, 2003; Matloff, 2005], the two classes of observing updates are handled separately. Unlike these studies, we preprocess the observed information and formulate a model which can handle both the cases identically. The advantage of formulating a single process is that, we can treat both the cases *last date of change* and *existence of change* identically simplifying the problem. For the case of *last date of change*, we do not need to reprocess the observed information. We simply consider each $t_{(k,m(k))}$ of k^{th} interval as update points and formulate the process of update time as $\{t(i) \mid t(i) = t_{(k,m(k))}, i = N(k\tau), i, k > 0\}$, where $N(t)$ is the number of updates during the period $(0, t]$.

For the case of *existence of change*, we do not have $t_{(k,m(k))}$. Assuming that sources are accessed periodically with an optimal interval, we can approximate $t(i)$, say the middle point or the end point of the period. Here, we approximate $t(i)$ as the middle point and formulate the process of update time as $\{t(i) \mid t(i) = (k-1)\tau + \tau/2, i = N(k\tau), i, k > 0\}$, where $N(t)$ is the number of observed updates during the period $(0, t]$.

Thus $\{t(i) : i = 0, 1, 2, ...\}$ represents the sequence of update points for both the cases satisfying the properties $t(0) = 0$ and $t(i) < t(i+1)$, where $t(i)$ is the time of i^{th} update. Since the client does not know exactly when and how often the Web page updates, the sequence $\{t(i) : i = 0, 1, 2, ...\}$ is an independent and identically distributed random process, which is often modeled as a Poisson process in several studies [Brewington and Cybenko, 2000; Cho and Garcia-Molina, 2003; Matloff, 2005]. We also assume the model as a Poisson process, but as a non-homogeneous Poisson process. One of the advantages of modeling the process as a non-homogeneous Poisson process is that we are able to determine instantaneous update rate i.e., $\lambda(t)$. In summary, our paper makes the following contributions:

-handles *last date of change* and *existence of change* cases identically.

-models the system as a *non-homogeneous Poisson process*.

-proposes two $\lambda(t)$ estimators namely Weibull and Duane plot estimators and compares the estimates using various datasets.

The rest of the paper is organized as follows. Section 2 discusses non-homogeneous Poisson distribution. In section 3, we discuss different estimators of rate of updates. In section 4, we divide the whole observation space into windows of consistent behavior. Experimental verifications are discussed in section 5. Then we conclude the paper in section 6.

2 Non-homogeneous Poisson Process

Non-homogeneous Poisson process over $[0, \infty]$ is determined by a rate function $\lambda(t) \geq 0$, known as *intensity density* of the Poisson process (see Chapter 6, Trivedi [Trivedi, 1982]). Let $N(T)$ be the number of updates during the time period $(0, T]$. Then, the *intensity function* of the process, i.e. mean number of updates during the period $(0, T]$ is defined as follows.

$$\mu(T) = E(N(T)) = \int_0^T \lambda(t)dt, \quad \text{for} \int_0^\infty \lambda(t)dt = \infty$$

Now, the probability that the number of updates during the period $(0, T]$ equals to k, is given by

$$Pr[N(T) = k] = \frac{[\mu(T)]^k}{k!} e^{-\mu(T)}$$

for $k = 0, 1, 2, ...$ Time homogeneous Poisson process is a special case, where $\lambda(t) = \lambda$ for all time instances $t \geq 0$. Now, the probability of $N(T)$ becoming zero is equal to $Pr[N(T) = 0] = e^{-\mu(T)}$. Thus we can write its cumulative distribution function (*cdf*) as $F(T) = 1 - e^{-\mu(T)}$ and its probability density function (*pdf*) as $f(T) = \mu'(T)e^{-\mu(T)} = \lambda(t)e^{-\int_0^T \lambda(t)dt}$. Using expression of $F(t)$ and $f(t)$ above, we can express instantaneous $\lambda(t)$ as

$$\lambda(t) = \frac{f(t)}{1 - F(t)} \tag{1}$$

3 Estimating $\lambda(t)$

As stated in section 1, we divide the whole observation space into a sequence of windows of consistent behavior. A window is a sequence of update points satisfying the property that $t(i) > t(i-1)$, where $t(i)$ is the time of i^{th} update. Since the sequence of update points are independent and identically distributed and satisfy the memoryless property, we can process each window independently to determine $\lambda(t)$.

Definition 1 *A window is said to be consistent, iff either one of the followings is true.*
1. $\lambda(t(i)) < \lambda(t(j)), \forall i < j, j > 0$ *i.e., increased*
2. $\lambda(t(i)) > \lambda(t(j)), \forall i < j, j > 0$ *i.e., decreased*
3. $\lambda(t(i)) = \lambda(t(j)), \forall i < j, j > 0$ *i.e., constant*

3.1 Intuitive Measure: $N(t)/t$

If $N(t)$ is the number of updates during the period $(0, t]$, the rate of updates at the time t is $\lambda(t) = N(t)/t$ and instantaneous *mean time between updates (MTBU)* at time t, $MTBU(t) = t/N(t)$. This estimation is suitable when the window satisfies the third consistency property i.e., the rate of update is *constant*. It is the biased estimation of $\lambda(t)$ in a homogeneous Poisson process [Cho and Garcia-Molina, 2003].

3.2 Estimation using Weibull Distribution

If the rate of update is either *increasing* or *decreasing* or *constant*, then Weibull distribution is a good mechanism to model the process. The two parameter Weibull *pdf* is given by

$$f(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} e^{-(t/\eta)^{\beta}}, \tag{2}$$

where β is known as *shape parameter* and η is known as *scale parameter* and $\beta, \eta > 0$ (see Chapter 2, Patrick [O'Connor, 2002], [Lieblein, 1955]). Then its *cdf* is given by $F(t) = 1 - e^{-(t/\eta)^{\beta}}$. Substituting equation 2 and $F(t)$ in equation 1, we get

$$\lambda(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} \tag{3}$$

From the study [Lieblein, 1955], we have (1) if $\beta = 1$, then $\lambda(t) = 1/\eta$, a constant (*the process reduces to the homogeneous Poisson process*), (2) if $\beta < 1$, then $\lambda(t)$ decreases as t increases and (3) if $\beta > 1$, then $\lambda(t)$ increases as t increases. A window is always in one of the above three states. Therefore, Weibull's distribution is obviously a good choice of estimating $\lambda(t)$.

In [Tsokos, 1995], the author found the relationship between $MTBU(t)$ and $\lambda(t)$ as follows.

$$MTBU(t_i) = \begin{cases} = \frac{1}{\lambda(t_i)} & \text{for } \beta = 1 \\ > \frac{1}{\lambda(t_i)} & \text{for } \beta < 1 \\ < \frac{1}{\lambda(t_i)} & \text{for } \beta > 1 \end{cases}$$

Thus, $1/\lambda(t)$ defines the bounds of $MTBU(t)$. Now our task is to estimate the parameters β and η. We estimate these two parameters using *maximum likelihood estimator* as given below. The conditional *pdf* of the i^{th} event given that the $(i-1)^{th}$ event occurred at t_{i-1} is given by $f(t_i \mid t_{i-1}) = \frac{\beta}{\eta}\left(\frac{t_i}{\eta}\right)^{\beta-1} e^{-\frac{1}{\eta^{\beta}}(t_i^{\beta} - t_{i-1}^{\beta})}, t_{i-1} < t_i$. Now, the likelihood function can be defined as

$$L(\beta, \eta) = \prod_{i=1}^{n} f(t_i \mid t_{i-1}) = \left(\frac{1}{\eta}\right)^{n\beta} \beta^n e^{-(\frac{t_n}{\eta})^{\beta}} \prod_{i=1}^{n} t_i^{\beta-1}$$

Taking log on both sides we get

$$\log L(\beta, \eta) = n\log(1/\eta^{\beta}) + n\log\beta - (\frac{t_n}{\eta})^{\beta} + (\beta-1)\sum_{i=1}^{n} t_i$$

Assuming $c = 1/\eta^{\beta}$, we can write

$$\log L(\beta, c) = n\log(c) + n\log\beta - ct_n^{\beta} + (\beta-1)\sum_{i=1}^{n} t_i$$

Taking partial derivative w.r.t. c and equating to zero, we get

$$\hat{\eta} = \frac{t_n}{n^{1/\beta}}. \tag{4}$$

Again, taking partial derivative w.r.t. β and equating to zero, we get

$$\hat{\beta} = \frac{n}{n\ln t_n - \sum_{i=1}^{n}\ln t_i} = \frac{n}{\sum_{i=1}^{n}\ln\frac{t_n}{t_i}} \tag{5}$$

Lemma 1 *$\hat{\beta}$ is biased for small value of n and equal to $\frac{n}{n-2}\beta$*

Proof As reported in [Calabria *et al.*, 1988], $2n\beta/\hat{\beta} \sim Z$, where Z is a chi-square random variable with $2(n-1)$ degree of freedom. Then we can write $\hat{\beta} = \frac{2n\beta}{Z} = \frac{2n\beta}{\Gamma(n-1)2^{n-1}} t^{n-2}e^{-t/2}$. Now, the expectation of $\hat{\beta}$ can be derived as $E[\hat{\beta}] = \frac{2n\beta}{\Gamma(n-1)2^{n-1}}\int_0^{\infty} t^{n-3}e^{-t/2}dt = \frac{2n\beta}{(n-2)\Gamma(n-2)2^{n-1}}\Gamma(n-2)2^{n-2} = \frac{n}{n-2}\beta$. If n is small, then $\hat{\beta}$ is biased. □

Now, Lemma 1 suggests correcting the ML by a factor of $(n-2)/n$ and thus obtain

$$\tilde{\beta} = \frac{n-2}{n}\hat{\beta} = \frac{n-2}{\sum_{i=1}^{n}\ln\frac{t_n}{t_i}}.$$

We can easily prove that $E[\tilde{\beta}]$ is unbiased i.e., $E[\tilde{\beta}] = E[\frac{n-2}{n}\hat{\beta}] = \frac{n-2}{n}E[\hat{\beta}] = \beta$. Thus we get $\tilde{\eta} = \frac{t_n}{n^{1/\beta}}$. Substituting the value of $\tilde{\beta}$ and $\tilde{\eta}$ in equation 3, we determine the value of $\tilde{\lambda}(t) = n\tilde{\beta}t^{\tilde{\beta}-1}/t_n^{\tilde{\beta}}$. In [Calabria *et al.*, 1988], it is proved that $\tilde{\lambda}(t)$ is less biased i.e., $E[\tilde{\lambda}(t)] = \lambda(t)(n-2)/(n-3)$. Again, we correct $\tilde{\lambda}(t)$ by a factor $(n-3)/(n-2)$ to get an unbiased estimation and thus obtain

$$\check{\lambda}(t) = \frac{n-3}{n-2}\tilde{\lambda}(t) = \frac{(n-3)n}{n-2}\tilde{\beta}t^{\tilde{\beta}-1}/t_n^{\tilde{\beta}}.$$

It can easily be proved that $E[\check{\lambda}(t)]$ is unbiased i.e., $E[\check{\lambda}] = E[\frac{n-3}{n-2}\tilde{\lambda(t)}] = \frac{n-3}{n-2}E[\tilde{\lambda}(t)] = \lambda(t)$. We use this estimator in our experiment. Another unbiased estimate of $\lambda(t)$ is $\bar{\lambda}(t) = (n-3)\hat{\beta}t^{\hat{\beta}-1}/t_n^{\hat{\beta}}$, which is obtained by correcting $\hat{\lambda}(t)$ by a factor of $(n-3)/2$ [Calabria *et al.*, 1988]. Once we get windows of consistent behavior (see section 4), we can thus estimate $\lambda(t)$.

3.3 Estimation using Duane Plots

In this subsection, we discuss another $\lambda(t)$ estimator using Duane plot [Duane, 1964].

Property 1 *Let $MTBU(t_i) = \frac{t_i}{i}, i > 0$ be the cumulative mean time between updates at i^{th} update. If the sequence of $MTBU(t_i)$ is either increasing or decreasing or constant as i increases and if we plot a graph between $MTBU(t_i)$ along y-axis and t_i along x-axis on* log-log *graph paper, the points tend to line up following a straight line. This plot is known as Duane Plot.*

The slope of the straight line passing through points is known as *Duane plot slope*. From the straight line, we can write $\log MTBU(t) = \log \alpha + \gamma \log t$, where γ is the *Duane plot slope* and α is the intercept when $t = 1$. Equating $MTBU$ to its expected value, we get $MTBU(t) = \alpha t^{\gamma}$. Similarly, cumulative $\lambda(t)$ can be written as $\lambda(t) = \frac{1}{\alpha} t^{-\gamma}$. Now, we can calculate the slope γ as

$$\gamma = \frac{\log MTBU(t_i) - \log MTBU(t_j)}{\log t_i - \log t_j}. \quad (6)$$

As we consider $\lambda(t)$ estimation only within consistent windows, Duane plot can be applied to estimate $\lambda(t)$. This estimation works fine as long as both the points $(MTB(t_i), t_i)$ and $(MTBU(t_j), t_j)$ are on the best straight line through the points. But, it is not the case in reality. Therefore, we may need to find out the straight line which is closest to all the observed points and take γ as the slope of the obtained straight line. We use simple linear regression to find the best straight line closest to all the observed points.

3.4 Estimation using Linear Regression

If we consider a consistent window with either *constant* or *increasing* or *decreasing* $\lambda(t)$, linear regression is an effective mechanism to estimate $\lambda(t)$. Under such conditions, the relationship between $\lambda(t)$ and t can be defined by a straight line. Linear regression defines a straight line closest to the data points. We can define the equation of the closest straight line as $y' = a + bx$, where a and b are regression coefficients and defined as $a = \overline{y} - b\overline{x}$ and $b = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sum (x_i - \overline{x})^2}$. We apply simple linear regression to get the best straight line closest to all the observed points in the *Duane plot*. If we consider $x = \log t_i$ and $y = \log MTBU_c(t_i)$, then b is equal to the slope of the *Duane plots* γ and a is equal to the intercept α.

3.5 Measuring error

We define an error at a point i as the difference between the observed value and predicted value at that point i i.e., $\bigtriangledown(i) = f_{observed}(i) - f_{predicted}(i)$. The error in measuring $\lambda(t)$ is minimum if error in measuring $MTBU(t)$ is minimum and vice versa. Again, overall error in $MTBU(t)$ is minimum when $\sum_{i=1}^{N(t)} \bigtriangledown(i) = \sum_{i=1}^{N(t)} \big(MTBU(t(i)) - (t(i) - t(i-1))\big)$ is minimum. Thus, $\sum_{i=1}^{N(t)} \bigtriangledown(i)$ can be used as a measure to compare different estimations of $\lambda(t)$. The $\lambda(t)$ with the minimum $\sum_{i=1}^{N(t)} \bigtriangledown(i)$ is considered as the best estimate. But, in the case of linear regression mechanism $\sum_i \bigtriangledown(i) = 0$. To avoid this problem, we can use mean of square error $MSE = \sum_{i=1}^{n} \bigtriangledown(i)^2/n$ as a measure to compare different estimators.

4 Generating Consistent Window

In reality, most of the Web page update period sequence is inconsistent. It is found to be a sequence of *constant*, *increasing* and *decreasing* over subsequent period of times in any order. In this section, we discuss a procedure to generate consistent windows (as define in Definition 1) out of a sequence of update points. Before we discuss the procedure, we define the localized $\lambda(t)$. If W be the set of update points, then the average number of update points within the window W can be defined as $\mu(W) = \int_W \lambda(x)dx < \infty$, where $\lambda(x)$ is locally integrable. Now the localized $\lambda(x)$ can be defined as $\lambda(x) = \lim_{W \to x} \frac{Pr(N(W)=1)}{|W|}$, where $N(W)$ denotes the number of updates within W and x is an update point. Thus, it is necessary for the window W to be consistent so as to be able to predict $Pr(N(W) = 1)$.

Now, we divide the total observation space into windows W_i of consistent $MTBU$ i.e. either *constant* or *increasing* or *decreasing* in nature. We define the length of a window $\mid W_i(k) \mid$ by $(t_{j+k} - t_j), j > 0$, where $W_i(k)$ starts just after j^{th} update and k is the number of update points within the window. The accuracy of the estimation depends on the nature of the update points within the window. So, *how do we decide the window size?* The important factor in deciding the window size i.e., k is the number of consistent points within the window, not $\mid W_i(k) \mid$.

Consistent window: We consider the last (first) three update points as the initial window and move backward (forward). A window is always in one of the three states, either *constant* or *increasing* or *decreasing* state. The window extends if the next point at t_j is consistent with the state of the current window. We often do not find consistent windows of large size. To increase the size of the windows, we allow few inconsistent update points within the window with a limit on the amount of discrepancies. Even if an update point t_j is not exactly consistent with the current state, we consider t_j as a consistent point if the amount discrepancy is below some *threshold value*, otherwise we create the next window. We compare the accuracy of different estimators with different *Threshold value* in Section 5

5 Experimental Evaluation

The comparisons of different estimators are based on both synthetic datasets (*collected from the UCLA WebArchive project data available at http://webarchive.cs.ucla.edu/*) and real Web datasets. Synthetic datasets consist of 3,00,000 pages' records and each record contains 200 access units. The real Web datasets were collected locally. 27034 number of frequently access Web pages were identified from our local proxy server log file. These pages were crawled every day for two months (12^{th} *April, 2006 to* 12^{th} *June, 2006*) to keep track of the changes. Out of these pages, 16522 number of pages are updated with *Last-Modified* information. For the remaining pages, we have detected the change manually using a simple cross validation technique. For each page, we have extracted the set of words and corresponding number of occurrence, including set of *hyperlinks*. If the set of words or their occurrence values between two subsequent images of a page are different, then we consider it as update. We use update unit on a *per-day* basis. We consider both the cases *last date of change* and *existence of change* identically and mix the records to form real Web datasets.

Our experimental evaluations consists of two parts. In the first part, we estimate the localized $\lambda(t)$ using the estimators

DataSet	Estimator	Total	Const	Incr	Decr
Synthetic	Intuition	3.587	0.345	12.891	4.944
	Weibull	2.363	0.249	7.955	3.274
	Duane	2.956	0.339	10.307	4.059
Real	Intuition	3.82	0.265	3.598	4.692
	Weibull	2.2	0.175	2.020	2.704
	Duane	2.6	0.262	3.205	3.159

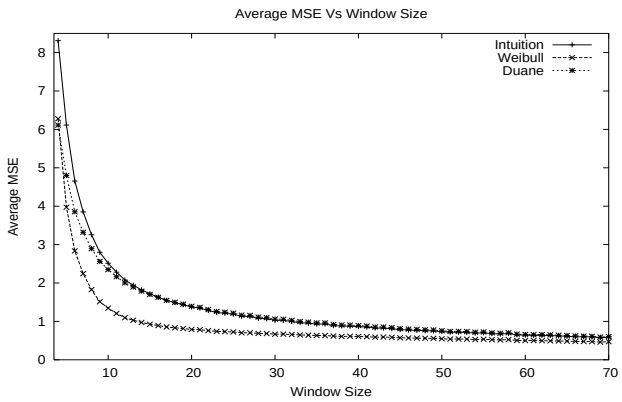
Table 1: Average MSE for different estimators over different window types. Windows were generated with a *Threshold* value of 3.

discussed in section 3 from the sequence of data points in the observation space. We generate the sequence of consistent windows using the procedure discussed in Section 4. We consider consistent windows with a minimum size of four update points. If the size of the window is less than four, then we merge one half with the left window and another half with the right window in the sequence introducing some discrepancies. For each window, we estimate $\lambda(t)$ at every update point t using different estimators. For simplicity we approximate $MTBU(t) \sim 1/\lambda(t)$ and then regenerate the whole observation space again using the estimated $MTBU(t)$. The difference between actual update point and estimated update point is considered as an error. We have recorded errors in every estimated update points and calculated average MSE for each window to compare the accuracy of different estimators.

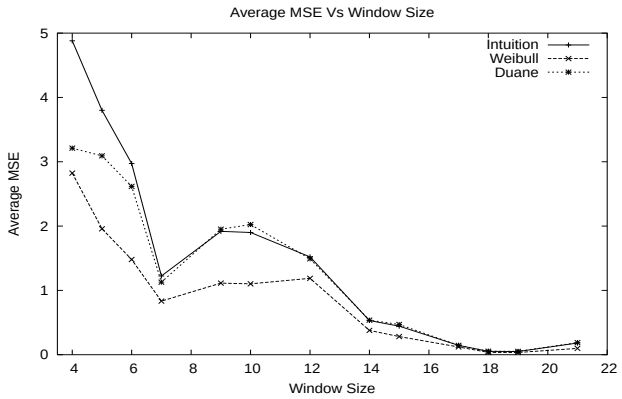
		Rest	Intuitive	Weibull	Duane
Syn	Intuition	0.5%		5%	50%
	Weibull	91.5%	95%		91.5%
	Duane	6.2%	47.2%	8.5%	
Real	Intuitive	0%		3.4%	17.44%
	Weibull	96.1%	96.5%		96.5%
	Duane	2.3%	81.44%	3.4%	

Table 2: Estimator in the row outperforms estimator in the column. Windows were generated with a *Threshold* value of 3. Syn :Synthetic

Table 1 compares the average MSE of different estimators in terms of different window classes. We find that Weibull estimator outperforms other estimators for all the cases. For the case of *constant state* windows, all these three estimators performs equally well with very small average MSE. But, Weibull estimator outperforms the rest with significantly smaller average MSE for both increasing and decreasing state windows. In Table 2, we show the percentage of the number of windows out of the whole windows in which row estimator outperforms the column estimator. It shows that Weibull estimator outperforms the rest in 91.5% of the whole windows for synthetic datasets and in 96.1% of the whole windows for real Web datasets. Whereas the intuitive estimator outperforms the rest only in 0.5% for synthetic datasets, which is negligible and 0% for our real datasets. The case of 0% may not be always true. Again we compare different estimators in terms of the average MSE for different window sizes as shown in Figure 1. It clearly shows that error decreases as we



(a) Synthetic Data



(b) Real Data

Figure 1: Window size versus MSE between different estimators. Windows were generated with a *Threshold* value of 3.

increase the size of window. For the larger size windows, all three estimators performs equally well (still Weibull outperforms by smaller extents), but for the smaller size windows, Weibull outperforms the other estimators by almost double.

We again investigate the affect of *threshold value* on average MSE. Larger *threshold value* means more inconsistent data points winthin the window. It can be clearly seen from Figure 2 that Weibull estimator significantly outperforms the rest for wider range of *threshold values*. It also shows that amount of errors for all the estimators increase, as we increase the threshold value. It is because, with the increase in threshold value, noise within the window increase. This part of the experimental evaluations show *how accurate is the estimator*.

The main motivation behind investigating localized $\lambda(t)$, rather than global λ is to use the piecewise information to improve the prediction of future update points. For example, *the trend in $\lambda(t)$ during the month of January for last five years can be used (or combined with recent trend) to predict future $\lambda(t)$ of next January*. Since we do not have enough datasets to make use of such information, we leave such study as our future work.

However in the second part of our experiment, we investigate *the effectiveness of different estimators* by predicting

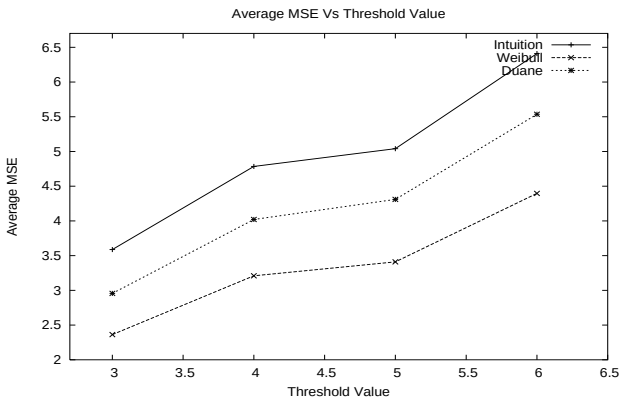


Figure 2: Threshold Value versus average overall MSE between different estimators over Synthetic datasets.

the future update points from the past information running a simulation program. We started with first consistent window in the observation space and based on this window, we predicted the next update point. If the new update point is not consistent with the current window, then we try to generate a new consistent window containing the new point, if possible, otherwise include it as a noise in the current window. From Table 3, we find that Duane plot estimator detects 88.6% of total updates, whereas Weibull estimator detects only 76%, but Duane plot is expensive i.e., around 36% of the total number accesses are comsumed in *premature visits*. The problem with Weibull estimator is that it overestimates (underestimates) the prediction during the transition from decreasing (increasing) state window to other state. We can further improve the performance Weibull estimator by carefully setting a bound on prediction. For example, $t_{next} = t_{cur} + \alpha MTBU(t_{cur}), \alpha \in (0, 1)$ if the window is in decreasing state. We have observed the detection of updates upto 86.7% by Weibull estimator, when $\alpha = 0.7$.

		%Unobs	%Obs	#Premature
Synthetic	Intuition	41.4%	58.6%	2889929
	Weibull	24%	76%	7196551
	Duane	11.7%	88.3%	11041074
Real	Intuition	34.8%	65.2%	110450
	Weibull	37.7%	72.3%	197257
	Duane	11.4%	88.6%	308924

Table 3: Number of unobserved updates, observed updates and premature visits experienced by each estimator. Synthetic (Real) datasets have 29435529 (609584) number of updates. Windows were generated with a *Threshold* value of 3

Comparison with homogeneous Poisson estimator: The *intuitive* estimator i.e., $N(t)/t$ is equivalent to biased homogeneous Poisson estimator [Cho and Garcia-Molina, 2003; Matloff, 2005]. However, comparison with unbiased homogeneous Poisson estimators is left as out of the scope of our work. It is because, the unbiased estimators in [Cho and Garcia-Molina, 2003; Matloff, 2005] are based on visiting the pages periodically with an interval τ, whereas our estimator is based on actual observed update points on the fly.

6 Conclusion

In this paper, we model the process of web page updates as non-homogeneous Poisson process. Since the update points are independent and identically distributed, we can not simply divide the number of observed updates by the observation time to estimate rate of updates. We divide the whole observation space into independent consistent windows to estimate localized rate of updates. In a scenario with inconsistent rate of updates, piecewise estimation provides more detail and useful information compared to global estimation. In this paper, we have discussed three estimation mechanisms namely *intuitive measure*, *Weibull* and *Duane plot*. Tests on both synthetic and real datasets confirm that Weibull's estimator outperforms both *intuitive measure* and *Duane plot* while estimating localized rate of updates even for the small size windows. We have also investigated the effectiveness of using these estimators to predict future update points and Duane plot estimator is found to outperform the rest in terms of repository freshness. We find that Weibull and Duane plot estimators perform equally if we reduce the affect of overestimation during state transistion by carefully setting a limit on overestimation.

References

[Brewington and Cybenko, 2000] Brian E. Brewington and George Cybenko. How dynamic is the web? In *Proceeding of WWW2000*, March 2000.

[Calabria *et al.*, 1988] R. Calabria, M. Guida, and G. Pulcini. Some modified maximum likelihood estimator for the weibull process. *Reliability Engineering and System Safety*, 23, 1988.

[Cho and Garcia-Molina, 2003] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Transaction on Internet Technology*, 3:256–290, August 2003.

[Duane, 1964] J. T. Duane. Learning curve approach to reliability monitoring. *IEEE Transactions on Aerospace*, 2, April 1964.

[Lieblein, 1955] Julius Lieblein. On moments of order statistics from the weibull distribution. *Annals of Mathematical Statistics*, 26, June 1955.

[Matloff, 2005] Norman Matloff. Estimation of internet file-access/modification rates from indirect data. *ACM Transactions on Modeling and Computer Simulation*, 15:233–253, July 2005.

[O'Connor, 2002] Patrick D. T. O'Connor. *Practical Reliability Engineering*. John Wiley, forth edition, 2002.

[Trivedi, 1982] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Application*. Prentice Hall, 1982.

[Tsokos, 1995] Chris P. Tsokos. *Reliability growth: nonhomogeneous poisson process*. CRC Press, 1995.