Exponentially Weighted Metrics
With Irregular Sampling

Dmitriy Lyubimov
dlyubimov@inadco.com

September 26, 2012

Contents

1 Notations

Internal state of the summarizer

$$S_n = \{w_n, s_n, t_n\} .$$

Observation values are $(x_1, x_2, ...x_n)$ corresponding to times $(t_1, t_2, ...t_n)$. Hence, single observation could be denoted as (x_i, t_i) . In our specific case, for the map/reduce paradigm, we can assume that earlier observations may occur further in the sequence for the same summarizer, i.e it is possible that $t_n < t_{n-1}$.

2 Short summary

Exponentially weighted time series analysis. Weighing irregularly weighted facts using simple exponent or difference of exponents (the latter is almost as simple but has desired property of $f'(0) = 0$. With regular exponent, most recent observations get discounted at highest rate than any other observations $f'(e^{-x}) = -1$. This is suboptimal in some situations. Difference of exponents is as simple as single exponent weighting but has desired identity of $f'(x) = 0$. We provide irregularly sampled average and rate using this technique.

Binomial Bayesian estimators. Exponentially weighted time series estimators representing Bayesian treatment (right now, for Bernoulli events only). The idea is that given prior belief Beta (α, β) and some time series evidence (a bunch of Bernoulli trials) $\mathbf{X}$, we can derive posterior belief Beta (α', β'). We consider $\mathbf{X} = (\mathbf{x}, \mathbf{t})$, i.e. matrix consisting of bunch of observations $\left(x^{(1)}, t^{(2)}\right)$, $\left(x^{(2)}, t^{(2)}\right)$ ... $\left(x^{(n)}, t^{(n)}\right)$. Each individual observation in the evidence is therefore tied to a moment of time. Next, we marry exponential weights on observations with this conjugate prior analysis (i.e. form of Bayesian treatment) to produce posterior belief $p\left(\boldsymbol{\theta}'|\mathbf{X}, T, \boldsymbol{\theta}\right)$ for hyper-parameters (T is the amount of significant history under weighing assumptions). Finally, once we know shape of the posterior, we can produce posterior belief estimators for Bernoulli p (as an expectation of distribution density) and other interesting things such as credible intervals (in the open version, $\mathbb{E}\left(p\left(\boldsymbol{\theta}'|\mathbf{X}, T, \boldsymbol{\theta}\right)\right)$ only). Skipping math details, in this version we make some assumptions about prior variance to simplify things a bit but still need prior mean further denoted as P_0: $P_0 = \mathbb{E}\left(p\left(x|\boldsymbol{\theta}\right)\right)$ to derive reasonable prior parameters $\boldsymbol{\theta}$ so the actual posterior ends up with more human friendly parameters $p\left(\boldsymbol{\theta}'|\mathbf{X}, T, P_0\right)$. Even though an *uninformative prior* can be used with estimator, most subsequent sufficiently interesting applications arise from ability to apply *Informative Prior* approach (or even chain of informative priors) for estimating prior belief P_0 using various HBL slicing schemes including slicing for posteriors.

3 Exponentially weighted average with irregular sampling

3.1 Iterative state updates

Case $t_{n+1} \geq t_n$:

$$\begin{cases} \begin{cases} \pi_0 = 1, \\ \pi_{n+1} = e^{-(t_{n+1}-t_n)/\alpha}; \end{cases} \\ w_{n+1} = 1 + \pi_{n+1} w_n; \\ s_{n+1} = x_{n+1} + \pi_{n+1} s_n; \\ t_{n+1} = t_{n+1}. \end{cases}$$

Case $t_{n+1} < t_n$:

$$\begin{cases} \begin{cases} \pi_0 = 1, \\ \pi = e^{-(t_n - t_{n+1})/\alpha}, \end{cases} \\ w_{n+1} = w_n + \pi_{n+1}, \\ s_{n+1} = s_n + \pi_{n+1} x_{n+1}, \\ t_{n+1} = t_n. \end{cases}$$

Latest average value at any moment :

$$\mu(S) = \frac{s}{w}.$$

3.2 Combining two summarizers having observed two disjoint sets as subsets of original observation set

Combining two summarizer states S_1, S_2 having observed two disjoint sets of original observation superset:

Case $t_2 \geq t_1$:

$$\begin{cases} t = t_2; \\ \pi = e^{-(t_2 - t_1)/\alpha}; \\ s = s_2 + s_1 \pi; \\ w = w_2 + w_1 \pi. \end{cases} \tag{1}$$

Case $t_2 < t_1$ is symmetrical to the above w.r.t. to indices $(\cdot)_1, (\cdot)_2$.

The prerequisite for combining is $\alpha_1 = \alpha_2 = \alpha$ (history decay is the same).

3.3 Complement summarizer

Suppose we have results for two observation sets $S_1 = \{x_i, t_i\}$, $S_2 = \{x_j, t_j\}$ such that $S_1 \supseteq S_2$. We want to find summarizer state for a complement observation set

$$S = S_1 \backslash S_2.$$

It's not hard to see that finding complement summarizer is an inverse operation of combining. Indeed, since S and S_2 are disjoint sets such that $S_1 = S \cup S_2$ then $\text{sum}(S_1) = \text{combine}(\text{sum}(S), \text{sum}(S_2))$. Assuming inversion of the combine operation,

$$\text{case } t_1 \geq t_2 :$$

$$\begin{cases} t = t_1; \\ \pi = e^{-(t_1 - t_2)/\alpha}; \\ s = s_1 - s_2 \pi; \\ w = w_1 - w_2 \pi. \end{cases} \tag{2}$$

Note that case $t_1 < t_2$ is invalid, because $S_1 \supseteq S_2$, i.e. S_1 must include latest event in S_2. So we must assert $t_1 \geq t_2$.

Complement summarization comes up when optimizing slice hierarchies in the Time Series projections. E.g. suppose we have a summarizer for a month with UTC boundaries and we want to adjust it for timezone $z > 0$. Then what we want to do is find a result of

$$\text{complement}\left(\text{sum}\left(month + z_{fromMonthEnd}\right), z_{fromMonthStart}\right).$$

3.4 Initial parameter computation.

Given T is the desired history length that experiences phase out down to margin m in exponential scale ($m = 0.01$ default), history decay parameter

$$\alpha = \frac{-T}{\ln m}.$$

3.5 Implementation class

These formulas are implemented by class `OnlineExpWeightedIrregularSummarizer`.

4 Exponentially weighted biased binomial average with irregular sampling.

4.1 Binomial biased probability at time of the update $n + 1$:

$$P_{n+1} = \frac{b_+ + s_{n+1}}{b_+ + b_- + w_{n+1}},$$

where bias parameters are based on bias $P_0 = \lim_{n\to 0} P = \frac{b_+}{b_+ + b_-}$ as*

$$\begin{aligned} k &= \frac{1}{\min\left(P_0, 1 - P_0\right)}; \\ b_+ &= kP_0 \frac{\varepsilon - 1}{\ln \varepsilon}; \\ b_- &= k\left(1 - P_0\right)\frac{\varepsilon - 1}{\ln \varepsilon}. \end{aligned}$$

$\varepsilon \in (0, 1)$ is margin excluded from history for the purposes of bias, default $\varepsilon = 0.5$.

*for compactness of notation. In fact, computation is simpler, e.g. it's not difficult to see that for the case $P_0 \geq 0.5$ it becomes

$$\begin{aligned} b_- &= \frac{\varepsilon - 1}{\ln \varepsilon}, \\ b_+ &= \frac{P_0}{1 - P_0} b_-. \end{aligned}$$

Similarly, for the case $P_0 < 0.5$ we get:

$$\begin{aligned} b_+ &= \frac{\varepsilon - 1}{\ln \varepsilon}, \\ b_- &= \frac{1 - P_0}{P_0} b_+ \end{aligned}$$

b_- and b_+ is something that sometimes in literature is being referred to as “pseudocounts” (in this case, of Bernoulli trials). The intuition reasoning for this that is that if prior belief of a (binomial) event is not exactly 1 or 0, then there must have been at least one positive or negative event before. Corollary to this reasoning, we define the pseudocounts formula above.

Also, for the corner cases $P_0 = 1 \,\mathrm{or}\, 0$, one of the pseudocounts does not exist (approaches infinity) which probably amounts to no prior knowledge at all (our prior estimator always has seen either all 0 or 1s.). It is not difficult to see that no prior biased estimator should ever produce corner cases, and the very first estimator in the chain of inference (as in Beta-distribution mean estimator) should assume pseudo counts corresponding to 50% prior belief (no prior knowledge whatsoever).

So for the corner cases which should really never be motivated in reality we just correct prior knowledge to 0.5 automatically to avoid division by zero.

4.2 pNow()

Function that evaluates binomial prob from ’now’ point of view:

$$P_t = \frac{b_+ + s_n e^{-(t-t_n)/\alpha}}{b_+ + b_- + w_n e^{-(t-t_n)/\alpha}} \tag{3}$$

As we can see, this formula has many common elements with case $t_{n+1} > t_n$ of 3.1, so in practice these computations are combined into one.

4.3 Summing up pageload logs and positive event logs separately.

Typical case is when we have to analyze 2 different logs separately. One log typically contains event counts only (typically, page loads, so I will call it a “pageload” log) but no information of the user action on those. Second log contains information of the user action but most typically only if user had any action at all (i.e. only positives). At the end we want an estimate of the user action rate. First intuitive impulse is just to outer-join the logs and then use a summarizer to deduce the final inference.

But it appears that it may not always be practical. Specifically, for real time incremental compilations such as what we have in HBase-Lattice.

For once, we are not even guaranteed to see a complete user session in incremental fact streams and therefore we will not be able to guarantee we truly know whether event is negative or it is yet to become positive, even if we joined them. This strategy would probably be practical (although still not 100% accurate) on larger chunks of logs (such as daily chunks) but we don’t want to wait a day to derive what we already can tell right now.

And of course, joining is considerably more expensive than considering these log streams separately in a completely independent ETL fashion.

To weasel my way out of this real-time statistics predicament, I decided to use 2 summarizers separately and then combine them in a specific way to deduce final summarizer state at the request time. With this approach I still may account for some yet-to-be-completed user sessions as negatives at the request time, but it will self-correct if they become positives and i poll the summarizer state later. Also, I can decrease amount of false positives in the most recent statistics if I query HBL to exclude most recent counts (because that’s where my false negatives will likely be).

We will denote the summarizer that runs over pageload log as $S^{(0)}$ (assuming every pageload as a negative event at that time, but some of them actually may turn later to be positives) and the one that runs over positive event log as summarizer $S^{(1)}$.

After we computed those sums, we can't just combine those two summarizers per 1 above as our summarizers counted all positive events twice: once as $(0, t_i)$ in pageload log and once as $(1, t_i)$ in the positive event log [†]. The formulas to combine positive summarizers thus should not account for weights of 1's when merging them into 0. We actually could merge summarizers counting positive events onto base summarizer multiple times:

assuming case $t^{(1)} \geq t^{(0)}$:

$$\begin{cases} t = t^{(1)}; \\ s = s^{(1)} + s^{(0)} e^{-(t_2 - t_1)/\alpha}; \\ w = w^{(0)} e^{-(t_2 - t_1)/\alpha}. \end{cases}$$

Case $t^{(1)} < t^{(0)}$:

$$\begin{cases} t = t^{(0)}; & (no\, change) \\ s = s^{(0)} + s^{(1)} e^{-(t_2 - t_1)/\alpha}; \\ w = w^{(0)}. & (no\, change) \end{cases}$$

These formulas are implemented by `OnlineExpWeightedIrregularSummarizer#combineBinomialOnes()` method. Thus, to combine the impression binomial sum $S^{(0)}$ and click binomial sum $S^{(1)}$, we need to do `pageloadSum.combineBinomialOnes(positiveLogSum)`.

4.4 Implementation class

`OnlineBinomialBiasedExpWeightedIrregularSummarizer`.

5 Exponentially weighted rates

5.1 Rate updates

Exponentially weighted rates is essentially the same as exponentially weighted average except we rate it by count/time, so every event accounts for $x_n / (t_n - t_{n-1})$, where x_n plays role of **number of occurrences** of the event at time t_n:

$$r = \frac{\sum_i x_i e^{-(t_i - t_0)/\alpha}}{\sum_i \left[(t_i - t_{i-1}) \cdot \operatorname{avg} \left(e^{-x} |_{-(t_i - t_{i-1})/\alpha}^{0} \right) \right]},$$

which translates again into (assuming $\pi_0 = 1$, $w_0 = 0$, $s_0 = 0$):

Case $t_{n+1} > t_n$:

$$\begin{cases} \Delta t = t_{n+1} - t_n, \\ \begin{cases} \pi_0 = 1; \\ \pi_{n+1} = e^{-\Delta t/\alpha}; \end{cases} \\ w_{n+1} = \pi_{n+1} w_n + \alpha \cdot (1 - \pi_{n+1}); \\ s_{n+1} = x_{n+1} + \pi_{n+1} s_n; \\ t_{n+1} = t_{n+1}. \end{cases}$$

[†] Perhaps error due to simple history merge would be small enough and acceptable if there's huge skew towards negatives; but it doesn't cost much extra to compensate for this, so let's do a little more work here.

Case $t_{n+1} < t_n$:

$$\begin{cases} \Delta t = t_n - t_{n+1}, \\ \begin{cases} \pi_0 = 1, \\ \pi = e^{-\Delta t/\alpha}, \end{cases} \\ w_{n+1} = \max\left[w_n, \alpha \cdot (1 - \pi_{n+1})\right], \\ s_{n+1} = s_n + \pi_{n+1} x_{n+1}, \\ t_{n+1} = t_n. & \text{(no change)} \end{cases}$$

The term $\alpha\,(1 - \pi_{n+1})$ really corresponds to the function average of the exponent $f\,(x) = e^x$ multiplied by Δt which boils down to integrating area under exponent[‡]

$$\Delta t \cdot \bar{f}\left(x \middle| \left[-\frac{\Delta t}{\alpha}, 0\right]\right) = \int_0^{\Delta t} e^{-x/\alpha} dx =$$

$$= \alpha \cdot \left(e^0 - e^{-\Delta t/\alpha}\right) =$$

$$= \alpha\,(1 - \pi)\,.$$

5.2 rNow()

Similarly to `pNow()`, it is possible to pull rate from any future point of time and incorporate the gap since last even into the rate:

$$r_t = \frac{s_n \pi_t}{\alpha\,(1 - \pi_t) + w_n \pi_t},$$

where $\pi_n = e^{-(t - t_n)/\alpha}$.

5.3 Combining rate summarizers

See implementation class for detail. In most general case, when both summarizers have non-zero history, and $t_1 \geq t_2$,

$$\pi = e^{-\Delta t/\alpha},$$

$$w = \max\left(w_1, \pi w_2 + \alpha\,(1 - \pi)\right),$$

$$s = s_1 + \pi s_2,$$

$$t = \max\left(t_{1,} t_2\right).$$

Case $t_1 < t_2$ is symmetrical w.r.t. indices $(\cdot)_1$, $(\cdot)_2$.

[‡] Using this useful identity: $\int e^{ax} = \frac{1}{a} e^{ax}$.

5.4 Complementing rate summarizers

Again, per above 2, we assert $t_1 \geq t_2$. Strictly speaking, inverse operation for rates is not possible (at least, i don't see how). The only thing that is possible is to compute a complement set of events with assumption that S_2 includes all measurements in the correspondent continuous period (i.e. S1 is all events for April and S_2 is all events for 3-5 of April and we want to find rate during 1-2 , 6-30 of April.)

For that corner case, corrections for rates are:

$$s = s_1 - s_2\pi;$$

$$w = w_1 - w_2\pi.$$

Also, another invariant to assert is

$$w_1 - \alpha\left(1 - \pi\right) \geq \pi w_2,$$

which asserts that first observation of S_2 was not earlier than first observation of S_1.

Warning1: As I said, the above formula only works for 100% continuous intervals S_1 and S_2.

Warning2: Note that if you create summarizer for $\{x_1, t_1 ... x_k, t_k\}$ then it relates to time period $[t_1, t_k]$. Subsequently, if you combine two subsequnces of the above, it will correspond to times $[t_1, t_i]$ and $[t_{i+1}, t_k]$ which does not add up to a continues interval. Instead, it will have a time gap (t_i, t_{i+1}) that is unaccounted for. This will create errors for both combining rates and complementing continues rate intervals. To fix that, the first sequence S_1 needs to include additional sample $(0, t_{i+1})$ to connect correctly to the next slice.

In practice, when building projections, we just need to explicitly set boundaries of projections by adding samples $(0, t_{left})$ and $(0, t_{right})$.

5.5 Implementation class

`OnlineExpWeightedRateSummarizer`.

6 Canny filters

6.1 Adaptation

$$s\left(t\right) = ke^{-t} - \left(k-1\right)e^{-\frac{k}{k-1}t},\ k > 1.$$

This $s\left(t\right)$ (which Ted Dunning calls 'Canny's filter', Figure 1 on page 10) has desirable properties of $s'\left(0\right) = 0$ and $s\left(0\right) = 1$.

In our case, it takes form of

$$y\left(x\right) = ke^{-x/\alpha} - \left(k-1\right)e^{-kx/\alpha(k-1)}.$$

Given margin m and decay period T, so that

$$y\left(T\right) = m, \tag{4}$$

solving (4) for alpha, we get estimate for α assuming margin $m \ll 1$ as

$$\alpha \approx -\frac{T}{\ln\left(m/k\right)}.$$

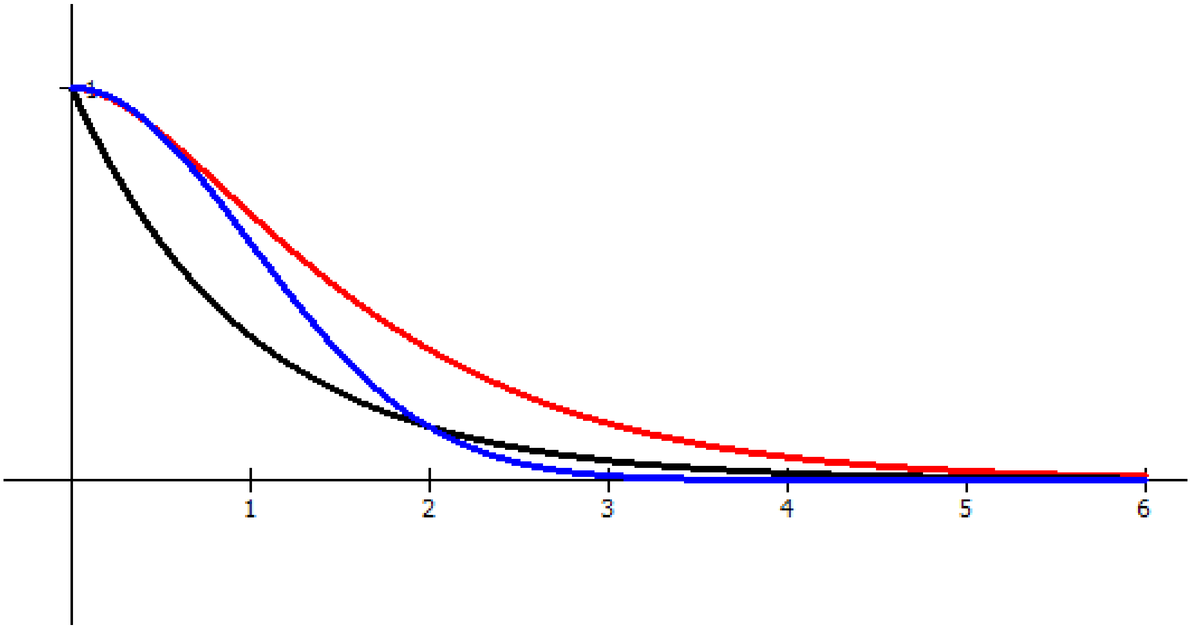

Figure 1: Canny filter with $k = 4$ (red) vs. $y(x) = e^{-x}$ (black) vs. bell-shape $y(x) = \sqrt{e^{-x^2}}$(blue).

6.2 Iterative update $t_{n+1} \geq t_n$

$$\begin{cases} \begin{cases} \pi_0 = 1,\, \nu_0 = 1, \\ \pi = e^{-(t_{n+1}-t_n)/\alpha}; \\ \nu = e^{-k(t_{n+1}-t_n)/\alpha(k-1)}; \end{cases} \\ w_{n+1} = 1 + \pi w_n; \\ v_{n+1} = 1 + \nu v_n; \\ s_{n+1} = x_{n+1} + \pi s_n; \\ u_{n+1} = x_{n+1} + \nu u_n; \\ t_{n+1} = t_{n+1}. \end{cases}$$

Then average

$$\mu = \frac{ks - (k-1)\, u}{kw - (k-1)\, v}.$$

Persisted summarizer state thus consists of parameters $\{w, v, s, u, t, k, \alpha\}$. Parameters k and α are set in constructor and thus don't evolve.

6.3 Iterative update in-the-past $t_{n+1} < t_n$

$$\begin{cases} \begin{cases} \pi_0 = 1,\, \nu_0 = 1; \\ \pi = e^{-(t_n - t_{n+1})/\alpha}, \\ \nu = e^{-k(t_n - t_{n+1})/\alpha(k-1)}; \end{cases} \\ w_{n+1} = w_n + \pi, \\ v_{n+1} = v_n + \nu, \\ s_{n+1} = s_n + \pi x_{n+1}, \\ u_{n+1} = u_n + \nu x_{n+1}, \\ t_{n+1} = t_n. \end{cases}$$

6.4 Combining Canny summarizers

Combining two summarizer states S_1, S_2 having observed two disjoint sets of original observation superset: Case $t_2 \geq t_1$:

$$\begin{cases} t = t_2; \\ \pi = e^{-(t_2 - t_1)/\alpha}; \\ \nu = e^{-k(t_2 - t_1)/\alpha(k-1)}; \\ s = s_2 + s_1\pi; \\ u = u_2 + u_1\nu; \\ w = w_2 + w_1\pi; \\ v = v_2 + v_1\nu. \end{cases} \tag{5}$$

6.5 Complementing Canny summarizers

Assuming $S = S_1 \backslash S_2$,

$$\text{case } t_1 \geq t_2 :$$

$$\begin{cases} t = t_1; \\ \pi = e^{-(t_1 - t_2)/\alpha}; \\ \nu = e^{-k(t_1 - t_2)/\alpha(k-1)}; \\ s = s_1 - s_2\pi; \\ u = u_1 - u_2\nu; \\ w = w_1 - w_2\pi; \\ v = v_1 - v_2\nu. \end{cases}$$

Case $t_1 < t_2$ is illegal (since S_1 must include all events in S_2, including the latest one).

Note1: Surprisingly, this intuition doesn't work in case of Canny sums, apparently, because it keeps ratios s/w and u/v identical but not $\frac{ks-(k-1)u}{kw-(k-1)v}$.

Note2: as additional unit testing shows, it apparently works, just not terribly numerically stable over significant amount of iterations as opposed to combining. So we should be avoiding a lot of nested complement operations with these functions (but 1 or 2 are o.k.)

6.6 Merging in positive samples only for binomial cases

Case $t^{(1)} \geq t^{(0)}$:

$$\begin{cases} t = t^{(1)}; \\ \pi = e^{-\left(t^{(1)} - t^{(0)}\right)/\alpha}; \\ \nu = e^{-k\left(t^{(1)} - t^{(0)}\right)/\alpha(k-1)}; \\ s = s^{(1)} + s^{(0)}\pi; \\ u = u^{(1)} + u^{(0)}; \\ w = w^{(0)}\pi; \\ v = v^{(0)}\nu. \end{cases}$$

Case $t^{(1)} < t^{(0)}$:

$$\begin{cases} t = t^{(0)}; & (no\,change) \\ \pi = e^{-\left(t^{(0)} - t^{(1)}\right)/\alpha}; \\ \nu = e^{-k\left(t^{(0)} - t^{(1)}\right)/\alpha(k-1)}; \\ s = s^{(0)} + s^{(1)}\pi; \\ u = u^{(0)} + u^{(1)}\nu; \\ w = w^{(0)}; & (no\,change) \\ v = v^{(0)} & (no\,change). \end{cases}$$

6.7 Peculiarities of Canny rate summarizer implementation

As with regular exponential rate summarizer, combining and complementing state operations work much smoother if for time increments, we take area under curve, instead of just time increment Δt, i.e. we use general approach of incremental build of time-related metric

$$\hat{w}_{n+1} = s(t_n)\hat{w}_{n-1} + \int_0^{\Delta t} s\left(t\right) dt. \tag{6}$$

The term $\int_0^{\Delta t} s\left(t\right) dt$ is essentially the same time, Δt, "continuously compressed" by the negative exponent decay, or, in this case, by Canny function decay.

The 6 comes apart as difference of two exponent related terms, which we previously denoted w and v.

Integrating time for the first exponent gives already familiar result

$$\begin{gathered} \int_0^{\Delta t} \exp\left(-\frac{x}{\alpha}\right) dx = \\ = \alpha \left[\exp\left(-\frac{x}{\alpha}\right) |_{\Delta t}^{0}\right] = \\ = \alpha \left(1 - \pi\right) \end{gathered}$$

Integrating time for second exponent

$$\int_0^{\Delta t} \exp\left(-\frac{k}{\left(k-1\right)\alpha} x\right) dx =$$

$$= \frac{(k-1)\,\alpha}{k} \exp\left(-\frac{k}{(k-1)\,\alpha} x\right) |_{\Delta t}^{0} =$$

$$= \frac{\alpha\,(k-1)}{k}\,(1-\nu)$$

(of course in the latter case if $w_n \geq \alpha\,(1-\pi) \equiv v_n \geq \frac{(k-1)}{k}\alpha\,(1-\nu)$, so we can optimize a little based on that).

Assuming $\pi_0 = 1$, $\nu_0 = 1$, but otherwise

$$\pi = \exp\left(-\frac{|t_{n+1} - t_n|}{\alpha}\right),$$

$$\nu = \exp\left(-\frac{k \cdot |t_{n+1} - t_n|}{\alpha\,(k-1)}\right),$$

for the case of $t_{n+1} \geq t_n$:

$$\begin{cases} w_{n+1} = \pi w_n + k\alpha\,(1-\pi)\,; \\ v_{n+1} = \nu v_n + \frac{(k-1)^2}{k}\alpha\,(1-\nu)\,; \end{cases}$$

and for the case of $t_{n+1} < t_n$:

$$\begin{cases} w_{n+1} = \max\left(w_n, \alpha\,(1-\pi)\right); \\ v_{n+1} = \max\left(v_n, \frac{(k-1)}{k}\alpha\,(1-\nu)\right). \end{cases}$$

The logic of advancing s_{n+1} and u_{n+1} is the same as in Canny average summarizer.

6.8 Peculiarities of combining Canny rate summarizers

Assuming both summarizers have nonzero history and $\alpha^{(1)} = \alpha^{(2)} = \alpha$, and $k^{(1)} = k^{(2)} = k$ as a precondition, combining routine looks like

$$\pi = \exp\left(-\frac{\left|t^{(2)} - t^{(1)}\right|}{\alpha}\right),$$

$$\nu = \exp\left(-\frac{k \cdot \left|t^{(2)} - t_n^{(1)}\right|}{\alpha\,(k-1)}\right),$$

$$\text{case } t^{(1)} \geq t^{(2)}:$$

$$\begin{cases} t = t^{(1)} \\ s = s^{(1)} + \pi s^{(2)} \\ u = u^{(1)} + \nu u^{(2)} \\ w = \max\left(w^{(1)}, \pi w^{(2)} + \alpha\,(1-\pi)\right) \\ v = \max\left(v^{(1)}, \nu v^{(2)} + \frac{(k-1)}{k}\alpha\,(1-\nu)\right) \end{cases}$$

case $t^{(1)} < t^{(2)}$ is symmetric w.r.t. indices $(\cdot)^{(1)}$ and $(\cdot)^{(2)}$. Also note that choices under both max() functions are identical, so we can save on comparisons a little bit.

6.9 Peculiarities of biased binomial Canny estimator

I haven't figured a good way to solve this exactly, so i am using approximations as per 1 exponent.

7 OnlineMeanSummarizer

This is a simple Welford summarizer computing mean and variance with combine method provided and supported `Writable` serialization.

8 Test classes

Test class for summarizers is `OnlineSummarizerTest`.

9 Pig functions

Each pig function accepts initialization string determining type of summarizer and (optionally) its parameters. The initialization spring is space-separated <name>=<value> pairs:

```
sep = 1*( SPACE / TAB )
init-str = nvp *( sep nvp )
nvp = name *sep '=' *sep value    ; name-value pair
```

One common parameter is `type`. Currently available values are :

type=EXP_AVG exponential average with irregular sampling. Valid additional parameters are: `dt` (mandatory, history length), `m` (optional, default 0.01, history length margin).

type=EXP_BIASED_BINOMIAL exponentially weighted biased binomial estimator. Valid additional parameters are: `p0` (mandatory, prior value), `dt` (mandatory, history length), `m` (margin, optional, default 0.01), `epsilon` (bias significance margin, optional, 0.5 default).

type=EXP_RATE exponentially weighted rate of events (x is number of events at time t). Valid additional parameters are: `dt`, `m` (same as in exponential average).

type=CANNY_AVG Canny's filter weighted exponential average with irregular sampling. Valid additional parameters are: `dt` (mandatory, history length), `m` (optional, default 0.01, approximate history margin), `k` (optional, default 4, k-factor in Canny's filter function).

type=CANNY_AVG Canny's rate summarizer. Parameters are the same as for CANNY_AVG.

com.inadco.math.pig.ExpAvgUpdate(bag:{(x:double,t:double)}) Create & summarize a bag given as a parameter. Bag schema must contain a tuple (x_i, t_i). See use example below.
Function's constructor accepts initialization script that should contain `type` parameter and all mandatory parameters for that type per above.

com.inadco.math.pig.ExpAvgCombine(s1:bytearray,...,sN:bytearray) Combine an arbitrary number of summarizers.
The constructor accepts initialization string which can have only `type` parameter per above since this function never creates an 'empty' serializer but rather only deserializes existing ones.

com.inadco.math.pig.ExpAvgGet(s:bytearray [, tNow:double]) Get most recent summarizer value. Some summarizers can account for the time that has passed since the last observation (pNow parameter). If it is supplied, then that version of summarizer value is called. Otherwise, "now" is considered to be time of the last (most recent time-wise) observation.
The constructor accepts initialization string which can have only `type` parameter per above since this function never creates an 'empty' serializer but rather only deserializes existing ones.

com.inadco.math.pig.ExpAvgNew() Create an empty summarizer with parameters supplied in the init string.

Example. The example that uses the functions is found in `ExpAvgTest.java`. The pig script that is run is found in `expavgtest.pig`.

expavgtest.pig

```
DEFINE CannyUpdate6days com.inadco.math.pig.ExpAvgUpdate('type=CANNY_AVG dt=6.0');
DEFINE CannyCombine com.inadco.math.pig.ExpAvgCombine('type=CANNY_AVG');
DEFINE CannyGet com.inadco.math.pig.ExpAvgGet('type=CANNY_AVG');
DEFINE CannyNew6days com.inadco.math.pig.ExpAvgNew('type=CANNY_AVG dt=6.0');
data = load 'input.txt'  using PigStorage(',') AS (name:chararray,x:double,t:double);
describe data;
D10 = foreach data generate name, x,t;
describe D10;
D20 = GROUP D10 by name;
describe D20;
D30 = foreach D20 generate group as name, CannyUpdate6days(D10.(x,t)) as avg;
describe D30;
-- add additional empty compatible summarizer as avg1
D40 = foreach D30 generate name, avg, CannyNew6days() as avg1;
describe D40;
-- combine use demo D50 = foreach D40 generate name, CannyCombine(avg,avg1) as avg;
describe D50;
-- now get the final average and report it D60 = foreach D50 generate name, CannyGet(avg) as avg;
TFin = foreach D60 generate *;
describe TFin;
store  TFin into 'output';
```

References

[1] Credit is due to Ted Dunning's blog http://tdunning.blogspot.com/2011/03/exponentially-weighted-averaging-for.html which touched off the thought process on online summarization and exponent difference functions.

[2] http://weatheringthrutechdays.blogspot.com/search/label/biased%20summarizer