Clock estimator model
Working notes

Dmitriy Lyubimov

1 Online irregularly sampled exponentially weighed summarizer

General idea: given dataset of pairs $\{(x_i, t_i) : i = 1, 2, ...\}$ compute exponentially weighed average

$$\text{ewa}\,(t|\alpha) \triangleq \frac{\sum_i x_i \cdot e^{-(t-t_i)/\alpha}}{\sum_i e^{-(t-t_i)/\alpha}}.$$

Due to property $e^{t+\Delta t} = e^t \cdot e^{\Delta t}$ it is possible to create online update algorithm as follows.

static state (w, s, t) : initialized as $w = 0$, $s = 0$, $t = 0$.

update(x_i, t_i) :

if $t \le t_i$ then

$$\begin{cases} \pi \leftarrow e^{-(t_i - t)/\alpha}; \\ w \leftarrow \pi w + x_i; \\ s \leftarrow \pi s + 1; \\ t \leftarrow t_i. \end{cases}$$

else

$$\begin{cases} \pi \leftarrow e^{-(t - t_i)/\alpha}; \\ w \leftarrow w + \pi x_i; \\ s \leftarrow s + \pi. \end{cases}$$

Note that the state is 3 variables but can keep indefinite amount of history, down to precision.

The average itself is computed

$$\text{ewa}\,(\cdot) = \frac{w}{s}.$$

The scaling parameter is given more conveniently by "half period" $T_{0.5}$, i.e., time amount ago at which observations lose half of their weight comparing to the most current observation(s): $0.5 = e^{-T_{0.5.}/\alpha}$. It follows that $\alpha = -T_{0.5}/\log 0.5$. ■

2 Robust clock calibration with fast convergence.

2.1 Iterative step.

Requirements. The inputs are online updates in the form of $\{(a_i, t_i) : i = 1, 2, \dots\}$. a_i is human-made adjustment in terms of seconds at the time t_i. Positive adjustments imply added time and negative adjustments imply removed time. The goal is to advise on calibration number, c, which causes clock to go faster or slower. Seconds per day error d is assumed to be connected to calibration parameter c with monotonously increasing, quasi linear dependency (it doesn't have to be perfectly linear, but it is linear in the vicinity of any given point) $d = f\,(c)$.

The goal of the algorithm is to learn function f and c to minimize daily error d. Along with imperfectly linear assumptions, algorithm also must be able to handle reasonable errors in measurements of a_i. Also, calibration parameters c will have a significant rounding errors affecting numerical stability of the solution due to $c \in \mathbb{Z}$.

Implementation. First, we need to normalize current adjustment in terms of seconds per day:

$$d = \text{SPW} \cdot a_i / \left(t_{n+1} - t_n\right),$$

where SPW is seconds per week constant, and $t_{n+1} - t_n$ denotes time interval passed since the last known adjustment.

Within the vicinity of current assumption about c, c_i, we model $d = f\left(c | c \to c_i\right) = \alpha c$, $\alpha > 0$. The learning challenge is therefore to be aware of the rate α that translates calibration parameter into actual correction in terms of seconds per day, d, within the vicinity of current best known calibration setting c_i. During iterative update, we simply can work out daily error by setting calibration parameter increment

$$\Delta c = c_{n+1} - c_n = d / \hat{\alpha}, \tag{1}$$

where $\hat{\alpha}$ is our current estimate for rate α.

The first approximation for α updates works out trivially by correcting for the actual correction that happened, $d_n - d_{n+1}$:

$$\tilde{\alpha} \triangleq \frac{d_n - d_{n+1}}{\Delta c_n}, \tag{2}$$

that is, we were trying to correct reported error of d_n with a change of Δc_n but actually were still reported off by d_{n+1}. This formula discounts all imperfections in error measuring etc. and actually would lead to unstable process. To cope with this, we apply bayesian treatment in the form of obtaining a posterior estimate based on prior observation history:

$$\hat{\alpha} = \text{ewaUpdate}\left(\tilde{\alpha}, t\right), \tag{3}$$

where the Bayesian estimator has a reasonable half-period (1 week or so). That takes into account the entire history of observations, with weights being skewed heavily in favor of the rate observations made more recently.

The minimal necessary state of this online estimator is therefore $\{c, \Delta c, d\}$, and the ewa summarizer's state $\{w, s, t\}$.

We also track range and NaN states for $\hat{\alpha}$ computations as well as incoming adjustments d. If any of these falls out of allowed range (e.g., significant adjustment error, or timezone/initial clock change), then we simply automatically reset the model to initial state.

3 Experimental results

Simulation is set so that adjustments are made when error reaches 3 minutes, may have a human adjustment error uniformly distributed in the range -30 ... +30 seconds, initial rate guess 1.0 but actual rate is 0.33, and the daily error is -25 seconds. The system naturally initially doesn't know either necessary rate nor calibration settings to cancel out the error.

```
current calib: 0, adjusting for 25.0 sec per day after 7.2 days. Internal calib rate:1.0.

current calib: -26, adjusting for 16.42 sec per day after 10.962241169305724 days. Internal calib rate:1.0.

current calib: -51, adjusting for 8.169999999999998 sec per day after 22.03182374541004 days. Internal calib rate:0.6155189676323869.

current calib: -70, adjusting for 1.8999999999999986 sec per day after 94.73684210526324 days. Internal calib rate:0.37435203153916263.

current calib: -78, adjusting for -0.7400000000000002 sec per day after 243.2432432432426 days. Internal calib rate:0.26712168628017013.

current calib: -76, adjusting for -0.08000000000000185 sec per day after 2249.999999999948 days. Internal calib rate:0.34730555555277537.
```

```
current calib: -76, adjusting for -0.08000000000000185 sec per day after 2249.999999999948 days. Internal calib rate:0.34955555555555706.

current calib: -76, adjusting for -0.08000000000000185 sec per day after 2249.999999999948 days. Internal calib rate:0.34955555555555706.

current calib: -76, adjusting for -0.08000000000000185 sec per day after 2249.999999999948 days. Internal calib rate:0.34955555555555706.
current calib: -76, adjusting for -0.08000000000000185 sec per day after 2249.999999999948 days. Internal calib rate:0.34955555555555706.
```

So if we correct the system once error reaches 3 minute and did the aforementioned errors, in this simulation we'd have to correct it first time after 7.2 days after initial clock setting; then in 11 days again, then in 22, 94, and then it would be practically calibrated as good as it gets. If we applied more effort and made a reasonable attempt to synchronize seconds, and not just minutes, then the system would converge in 3 corrections to the stable calibration parameter (-76).

Note that this starts with 300% error on the calibration rate α. In practice, we could probably start with a better chip calibration rate estimate than this, and then simulations show it converges after second correction in most simulated cases .

In practice the error of the chip itself would probably fluctuate and even the perfectly converged model will require time adjustments from time to time.