Probabilistic Progress Bars

Martin Kiefel, Christian Schuler, Philipp Hennig

Max Planck Institute for Intelligent Systems, Tübingen, Germany

Abstract. Predicting the time at which the integral over a stochastic process reaches a target level is a value of interest in many applications. Often, such computations have to be made at low cost, in real time. As an intuitive example that captures many features of this problem class, we choose progress bars, a ubiquitous element of computer user interfaces. These predictors are usually based on simple point estimators, with no error modelling. This leads to fluctuating behaviour confusing to the user. It also does not provide a distribution prediction (risk values), which are crucial for many other application areas. We construct and empirically evaluate a fast, constant cost algorithm using a Gauss-Markov process model which provides more information to the user.

1 Introduction

The problem of predicting when the integral over a random rate will reach a certain level, i.e. to solve

$$R = \int_0^T r(t)\, dt \qquad (1)$$

for T, where $r(t)$ is a draw from a random process (below we will focus on specific Gaussian processes), is at the heart of a number of applied problems. For motivation, consider the following examples:

– Retailers need to predict the point at which their reserves of individual products run out, an event that incurs a big loss. For products which are sold at sufficiently fine increments such that stocks are essentially a continuous quantity, a stochastic rate of sale is a realistic model.
– The onboard computer of a car needs to predict the remaining range while the fuel tank is slowly emptying. Changes in the drive speed, steepness of the road and wind conditions continuously change the fuel efficiency of the car, making prediction nontrivial.
– A startup company that is trying to attract crowd funding needs to predict the point at which funding goals will likely be reached, to efficiently prepare production (Section 4.2).

In this manuscript, we will focus on a fourth example application which shares many of the fundamental properties of the ones above, and adds the complication of a particularly restrained computational budget: Plotting a progress bar to

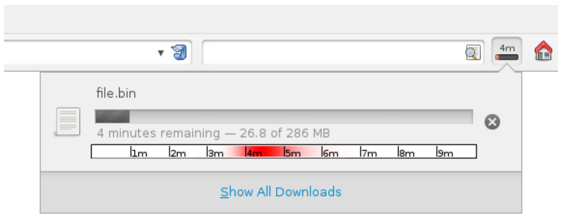

Fig. 1. A graphical representation of uncertainty about the remaining runtime of a download, resulting from the algorithm described below. The top half of the interface shows a classic progress bar, while the bottom half shows, in red, a cloud of uncertainty representing the weighted set of hypotheses over the remaining download time. The shown implementation is available as an open-source plugin. (Note that the plugin currently only works in the pull-down dialogue from the address bar, as shown in the figure, not in the separate download window also provided by Firefox).

predict the completion time of a file download on a desktop computer, under a stochastically varying connection bandwidth.

Although progress bars are ubiquitous elements of user interfaces, they are often treated as 'eye candy', at the margins of both the users' and the designers' attention. Visual design aspects of these pieces of graphical user interface have sometimes been studied [8], but the algorithms behind them are usually simplistic, a combination of simple moving averages, with ad-hoc backstops to avoid pathological values. The resulting predictions are not just unreliable, they are also qualitatively, visually unsatisfying: They give quickly varying predictions, which are confusing to the user, who has to mentally compute the averages.

Instead, we will develop a probabilistic progress bar algorithm, which returns a nonparametric probability distribution over the completion-time of a file download. This distribution can be shown graphically to the user (Figure 1), providing a less volatile, and more informative interface.

The mathematical 'lever' central to our derivations of a low-cost algorithm is the fact that Gaussian distributions, including nonparametric Gaussian process models, are closed under linear projections, and thus in particular also under integration: The integral over a Gaussian process is itself a Gaussian process. This observation is not new, and has been used repeatedly in the past to establish connections between numerical quadrature rules and Gaussian process regression [1, 3, 6, 7]. The twist in the progress bar setting (and related problems) is that the problem is inverted: Instead of predicting the *value* of the integral over a stochastic process at a particular *time*, one needs to predict the *time* at which the integral over the process reaches a particular *value*. This complicates the computation and typically gives non-Gaussian predictions (Figure 2).

In addition, our application requires a very low computational cost profile: Progress bars, as elements of the graphical user interface, must not cause serious computational overhead. We thus strive here to develop a algorithm of radically low, constant cost both in memory and computation time. This is achieved

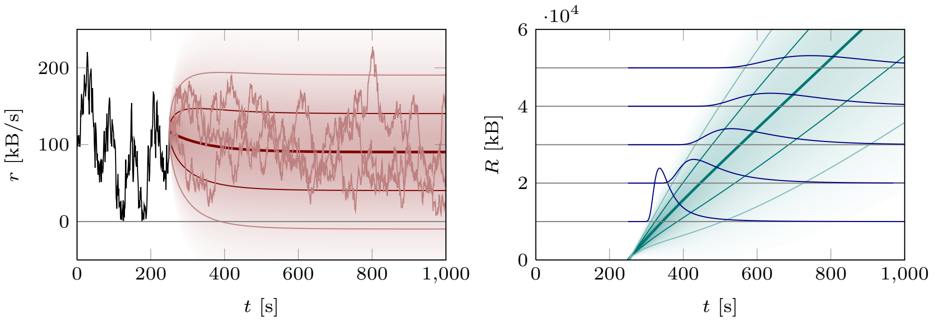

Fig. 2. Gauss-Markov process rate model. Left: an observed download rate $r(t)$ (black) gives rise to a Gauss-Markov process posterior belief about the future $r(t)$ (red, thick mean function, one and two standard deviations in decreasing red intensity, three samples from the belief in faint red). Right: The belief over the integral $R(t) = \int_{t_0}^{t} r(\tilde{t})d\tilde{t}$ is also a Gaussian process (green, same scheme as left). The belief for when a particular amount R_{target} is reached is a (normalised) cut through this Gaussian belief along a particular output value. In particular, it is *not* a Gaussian distribution (blue lines).

by using a nonparametric Gauss-Markov process prior for the download rate, with a parametric mean function, which leads to a filtering method [11] that has constant, very low cost, while capturing the important features of the rate distribution. Our algorithm has been implemented as a plugin for the open-source Firefox browser, available on the project page[1].

To simplify exposition and ease intuition, the remainder of this text will use concepts like the 'download rate' and 'data volume' specific to the example application of web browser's download progress bar. The results translate easily to other applications (e.g. the download rate could be replaced with the rate of products being sold, the rate at which funding pledges arrive, etc.).

2 Model

We assume that the algorithm has access to a rate function $r(t) : \mathbb{R} \to \mathbb{R}$ at recurring (not necessarily regularly spaced) time locations, describing the rate at which data accumulates. Crucially, observations $\boldsymbol{y} = [y(t_1), \dots, y(t_N)]$ of $r(t)$ at times $T = [t_1, \dots, t_N]$ can be made without observation noise, i.e. with Dirac likelihood $p(\boldsymbol{y} \mid r) = \delta(\boldsymbol{r}(T) - \boldsymbol{y})$. This is a realistic assumption for a web browser, and reasonable for many other applications (e.g. retailers have, of course, noiseless access to individual orders arriving).

We consider a Gaussian process prior over r, with constant mean β and Ornstein-Uhlenbeck [13] covariance function

$$p(r \mid \beta, \theta, \lambda) = \mathcal{GP}(r; \beta, k); \qquad \text{with} \qquad k(t, t'; \theta, \lambda) = \theta^2 \exp\left(-\frac{|t - t'|}{\lambda}\right). \quad (2)$$

[1] `http://people.tuebingen.mpg.de/mkiefel/projects/mlprogressbar/`

Ornstein-Uhlenbeck processes are a meaningful and relatively conservative prior for rate functions: Draws from OU processes are stationary (there is no reason to assume, a priori, that download rates change qualitatively over time), and continuous, but almost surely not differentiable [9, §4.2.1], allowing for relatively drastic variations in rate while retaining a degree of extrapolation ability. They also are first-order Markov processes [10, §I.23], a property that will become crucial for fast inference.

Further, we assign Gaussian uncertainty $p(\beta) = \mathcal{N}(\beta; 0, b)$ to the mean, and allow for arbitrarily large uncertainty on β, by taking $b \to \infty$. These priors and the Dirac likelihood give a Gaussian process posterior on r with mean and covariance functions [9, §2.7]

$$\mu(t) = k_{tT} K_{TT}^{-1} \boldsymbol{y} + R_t^{\intercal} \bar{\beta} \qquad \text{and} \tag{3}$$

$$\mathbb{V}(t, t') = k_{tt'} - k_{tT} K_{TT}^{-1} k_{Tt'} + R_t^{\intercal} A^{-1} R_{t'}. \tag{4}$$

This is using the widely adopted notation $k_{ab} = k(a, b)$ for kernel Gram matrices: if a has n_a elements and b has n_b elements, then k_{ab} is a matrix of size $n_a \times n_b$. We will shorten $K_{TT} \equiv K$ from now on. The other objects in the equation are the residual projection $R_t \equiv 1 - \mathbf{1}^{\intercal} K^{-1} k_{Tt}$, the mean of the belief over the empirical mean $\bar{\beta} \equiv A^{-1} \mathbf{1}^{\intercal} K^{-1} \boldsymbol{y}$ and its precision $A \equiv \mathbf{1}^{\intercal} K^{-1} \mathbf{1}$. The marginal likelihood, the evidence $p(\boldsymbol{y} \,|\, T, \theta, \lambda)$ for the data, is also Gaussian. Its logarithm is

$$\begin{aligned} \log p(\boldsymbol{y} \,|\, T, \theta, \lambda) = -\frac{1}{2} \big[& \boldsymbol{y}^{\intercal} K^{-1} \boldsymbol{y} - A^{-1} (\boldsymbol{y}^{\intercal} K^{-1} \mathbf{1})^2 \\ & + \log |K| + \log |A| + N \log 2\pi \big]. \end{aligned} \tag{5}$$

For efficient inference on the signal variance θ^2, we separate it from the kernel matrices, defining $\tilde{K} \equiv \theta^{-2} K$ and $\tilde{A} \equiv \theta^2 A$. The log evidence, as far as it relates to θ, can then be written up to additive terms as

$$\log p(\boldsymbol{y} \,|\, T, \theta, \lambda) = -\frac{1}{2\theta^2} \left[\boldsymbol{y}^{\intercal} \tilde{K}^{-1} \boldsymbol{y} - \tilde{A}^{-1} (\boldsymbol{y}^{\intercal} \tilde{K}^{-1} \mathbf{1})^2 \right] + \frac{N}{2} \log \theta^{-2} + \text{const.} \tag{6}$$

For hierarchical inference, we introduce a Gamma prior on θ^{-2}:

$$\log p(\theta^{-2} \,|\, a, b) = a \log b - \log \Gamma(a) + (a - 1) \log \theta^{-2} - \frac{b}{\theta^2}, \tag{7}$$

to get a Gamma posterior on θ^{-2} with

$$a' = a + \frac{N}{2}; \qquad b' = b + \frac{1}{2} \left[\boldsymbol{y}^{\intercal} \tilde{K}^{-1} \boldsymbol{y} - \tilde{A}^{-1} (\boldsymbol{y}^{\intercal} \tilde{K}^{-1} \mathbf{1})^2 \right], \tag{8}$$

which has its mean at a/b, its maximum at $(a - 1)/b$, and has variance a/b^2.

2.1 Fast Gauss-Markov Inference

General matrix inversion is computationally expensive if the size of the matrix grows with the number of datapoints. Furtunately, because the Ornstein-Uhlenbeck process is first-order Markov and the input domain (time) is scalar, the Gram matrix K can be inverted analytically, which gives rise to a light-weight, constant cost filtering algorithm.[2] Let there be N observations at sorted locations $t_1 < t_2 < \cdots < t_N$, and let the scaled distance between locations be $\delta_i = (t_{i+1} - t_i)/\lambda$, $i = 1, \dots, N-1$. Then a straightforward but tedious inductive argument shows that the inverse of the Gram matrix K with $K_{ij} = \theta^2 e^{-\sum_{k=\min(i,j)}^{\max(i,j)-1} \delta_k}$ is given by the symmetric tri-diagonal matrix

$$K^{-1} = \theta^{-2} \begin{pmatrix} c_1 & b_1 & 0 & \cdots & 0 \\ b_1 & c_2 & b_2 & & \vdots \\ 0 & b_2 & c_3 & \ddots & 0 \\ 0 & 0 & \ddots & c_{N-1} & b_{N-1} \\ 0 & \cdots & 0 & b_{N-1} & c_N \end{pmatrix} \qquad \text{with} \qquad \begin{aligned} b_i &= \tfrac{-e^{-\delta_i}}{1-e^{-2\delta_i}} \\ c_1 &= \tfrac{1}{1-e^{-2\delta_1}} \\ c_N &= \tfrac{1}{1-e^{-2\delta_{N-1}}} \\ c_{i\neq 1,N} &= \tfrac{1-e^{-2(\delta_i+\delta_{i+1})}}{(1-e^{-2\delta_i})(1-e^{-2\delta_{i+1}})}. \end{aligned} \tag{9}$$

To simplify notation for the following derivations, we use the shortcut $\varDelta_i = \exp(-\delta_i)$. Neatly, the starting case ($N = 0$) can be incorporated by considering an effective additional datapoint at $-\infty$ without changing the results, showing that the matrix K^{-1} is actually circular. We also find that $\log|\tilde{K}^{-1}| = -\sum_{i=1}^{N-1} \log(1 - e^{-2\delta_i})$. From the derivations in the preceding section, we observe that the sufficient statistics for inference are the scalar objects $\mathbf{1}\tilde{K}^{-1}\mathbf{1}$, $\mathbf{1}\tilde{K}^{-1}\boldsymbol{y}$ and $\boldsymbol{y}\tilde{K}^{-1}\boldsymbol{y}$. Using the aforementioned datapoint at $-\infty$, which amounts to $\delta_0 = \delta_N = \infty$ and thus $b_N = 0$, we get the compact form

$$\alpha \equiv \mathbf{1}_i \tilde{K}_{ij}^{-1} y_j = \sum_{i=1}^{N} c_i y_i + b_i(y_i + y_{i+1}) \qquad \tilde{A} = \mathbf{1}_i \tilde{K}_{ij}^{-1} \mathbf{1}_j = \sum_{i=1}^{N} c_i + 2b_i \tag{10}$$

$$\gamma \equiv y_i \tilde{K}_{ij}^{-1} y_j = \sum_{i=1}^{N} c_i y_i^2 + 2b_i(y_i y_{i+1}). \tag{11}$$

Up to this point, calculation of the different parts of the inference still is linear in the number of observables. For a constant-cost filtering rule, we treat the update from N to $N+1$ data points explicitly. A few lines of algebra show that updates for the three sufficient statistics, after collecting a new observation y_{new} at time step t_{new}, and given the observation y_{old} and statistics from the previous time step t_{old}, at distance $\varDelta \leftarrow \exp[-(t_{\text{new}} - t_{\text{old}})/\lambda]$, are

$$\tilde{A} \leftarrow \tilde{A} + \frac{1-\varDelta}{1+\varDelta} \qquad \alpha \leftarrow \alpha + \frac{y_{\text{new}} - y_{\text{old}}\varDelta}{1+\varDelta} \tag{12}$$

$$\gamma \leftarrow \gamma + \frac{(y_{\text{new}} - y_{\text{old}}\varDelta)^2}{1-\varDelta^2} \qquad t_{\text{old}} \leftarrow t_{\text{new}} \qquad y_{\text{old}} \leftarrow y_{\text{new}} \tag{13}$$

[2] The expositions in this section could also be formulated more generally (for Matérn-class covariances) in the framework of state-space models and associated filters [11]. The derivations here only work for our specific choice of the Ornstein-Uhlenbeck kernel (the first member of the Matérn class), but they allow a more straightforward treatment of the uncertainty on the parametric mean.

To perform this computation at any point in time, we only need to keep the variables t_{old}, y_{old}, $\tilde{A}$, α_{old}, γ_{old}, a, and b in memory. Due to the first order Markovianity of the Ornstein Uhlenbeck process, mean and covariance of the Gaussian posterior can then be found using the simple forms [11]

$$\Delta_* \equiv \exp[-(t_* - t_{\text{old}})/\lambda] \qquad R = (1 - \Delta_*) \tag{14}$$

$$\mu(t_*) = \Delta_* y_{\text{old}} + R^{\intercal}\bar{\beta} \qquad \mathbb{V}(t_*) = \theta^2[(1-\Delta_*^2) + R^{\intercal}\tilde{A}^{-1}R]. \tag{15}$$

Using $\bar{\beta} = \frac{\alpha}{\tilde{A}}$. Because of the conjugacy of the Gamma prior to precisions of Gaussians, the posterior over θ is also inverse Gamma, with

$$a = a_0 + \frac{N}{2} \qquad \text{and} \qquad b = b_0 + \frac{1}{2}\left(\gamma - \frac{\alpha^2}{\tilde{A}}\right). \tag{16}$$

which can be marginalized to give a Student-t prediction for the rate (see Equation (23) and following below). We initialise the set of variables to

$$t_{\text{old}} \leftarrow -\infty \qquad y_{\text{old}} \leftarrow 0 \qquad \tilde{A} \leftarrow \tilde{A}_0 \tag{17}$$

$$\alpha_{\text{old}} \leftarrow \alpha_0 \qquad \gamma_{\text{old}} \leftarrow 0 \qquad a \leftarrow a_0 \qquad b \leftarrow b_0 \tag{18}$$

with sensible values for $\tilde{A}_0, \alpha_0, a_0, b_0$, which can be used to propagate experience from past runs of the algorithm. For example, if the average download rate in previous runs was $\bar{r}$ with an empirical variance of σ_r^2, we set $\alpha_0 = \bar{r}$, $\tilde{A}_0 = 1$, $b_0 = 1e-2$, $a_0 = \sigma_r^2 b_0$ to get a broad Gauss-Gamma prior. For the very first download on a particular network connection, Algorithm 1 below contains sensible standard values for the progress bar setting, with rates measured in kB/s.

3 Constructing Predictions

The Gaussian family is closed under linear operations L:

$$p(t) = \mathcal{N}(t; m, V) \qquad \Rightarrow \qquad p(Lt) = \mathcal{N}(Lt; Lm, LVL^{\intercal}). \tag{19}$$

Since integration is a linear operation, a belief over the integral over the rate, the accumulated data at time t_*, $d(t_*) = \int_{t_0}^{t_*} r(t)\,dt$, can be constructed from the Gaussian process posterior mean $\mu(t)$ and covariance $\mathbb{V}(t,t')$ as

$$\begin{aligned} p(d(t_*)\,|\,\theta) &= \mathcal{N}\left(d(t_*); \int_{t_0}^{t_*} \mu(t)\,dt, \iint_{t_0}^{t_*} \mathbb{V}(t,t')\,dt\,dt'\right) \qquad (20) \\ &= \mathcal{N}\left(d(t_*); \mathfrak{k}_{t_*T}K^{-1}\boldsymbol{y} + \mathfrak{R}_{t_*}^{\intercal}\bar{\beta}, \kappa_{t_*t_*} - \mathfrak{k}_{t_*T}K^{-1}\mathfrak{k}_{Tt_*} + \mathfrak{R}_{t_*}^{\intercal}A^{-1}\mathfrak{R}_{t_*}\right), \end{aligned}$$

with the integrated projection operators (assuming the predictive distribution is only evaluated for future time points $t > t_N$)

$$\mathfrak{k}_{t_*T} = \int_{t_0}^{t_*} k(t,T)\,dt, \quad \mathfrak{R}_{t_*} = \int_{t_0}^{t_*} R_t\,dt, \quad \text{and} \quad \kappa_{t_*t_*} = \iint_{t_0}^{t_*} k(t,t')\,dt\,dt'. \tag{21}$$

In fact, as pointed out above, due to the first-order Markovianity of the Ornstein Uhlenbeck process, $\mu(t)$ and marginal variance $\mathbb{V}(t,t)$ only depend on the last observed function value, $y_{\text{old}} = r(t_{\text{old}})$. The integral prediction is simply given by

$$\begin{aligned} p(d(t_*) \mid \theta) &= \mathcal{N}\left[d(t_*); \mathfrak{k}_{t_* t_{\text{old}}} y_{t_{\text{old}}} + \mathfrak{R}_{t_* t_{\text{old}}} \bar{\beta}, \theta^2 [\kappa_{t_* t_*} - \mathfrak{k}^2_{t_* t_{\text{old}}} + \mathfrak{R}^\intercal_{t_* t_{\text{old}}} \tilde{A}^{-1} \mathfrak{R}_{t_* t_{\text{old}}}]\right] \\ &\equiv \mathcal{N}\left[d(t_*); \mu_\int(t_*), \theta^2 \sigma^2_\int(t_*)\right] \end{aligned} \tag{22}$$

$$\text{with} \quad \mathfrak{k}_{t_* t_{\text{old}}} = \lambda \left(1 - \exp\left(-\frac{t_* - t_{\text{old}}}{\lambda}\right)\right)$$

$$\text{and} \quad \mathfrak{R}_{t_* t_{\text{old}}} = (t_* - t_{\text{old}}) - \mathfrak{k}_{t_* f} \quad \text{and} \quad \kappa_{t_* t_*} = 2\lambda\left[(t_* - t_{\text{old}}) - \mathfrak{k}_{t t_{\text{old}}}\right]$$

The uncertainty in θ is incorporated by marginalisation, using the Gamma posterior from Equation (8). The resulting marginal over $d(t_*)$ is a Student t-distribution [e.g. 2, §2.3]

$$\begin{aligned} p(d(t_*)) &= \int_0^\infty p(d(t_*) \mid \theta) \mathcal{G}(\theta^{-2} \mid a, b)\, d\theta^{-2} \\ &= \int_0^\infty \frac{b^a \exp(-b/\theta^2) \theta^{-2(a-1)}}{\Gamma(a) \sqrt{2\pi\theta^2 \sigma^2_\int(t_*)}} \exp\left(-\frac{\theta^{-2}}{2}\left(\frac{d(t_*) - \mu_\int(t_*)}{\sigma_\int(t_*)}\right)^2\right) d\theta^{-2} \\ &= \frac{b^a}{\sqrt{2\pi}} \frac{\Gamma\left(a + \frac{1}{2}\right)}{\Gamma(a)} \left[b + \frac{(d(t_*) - \mu_\int(t_*))^2}{2\sigma^2_\int}\right]^{-a-1/2} \\ &= \mathrm{St}(d(t_*)/\sigma^2_\int; \mu/\sigma^2_\int, a/b, 2a). \end{aligned} \tag{23}$$

To construct a density for the probability of target D being reached at time t_*, we interpret the density of Eq. (23) as $p(d(t_*) = D)$ for every t_*, and normalise. Doing so causes a small error: The physical rate is strictly positive $r(t) \geq 0$, so there is one and only one correct D. But our Gaussian model puts a small amount of probability mass on negative rates. Hence interpreting a normalised $p(d(t_*) = D)$ as a density on D puts too much mass on "late" times, which are only possible under negative rates. The exact correction — enforcing strictly positive rates everywhere — involves an intractable integral. One could consider constructing a correction through an additional term multiplied with $p(D = d(t_*))$, decaying for large t_*. Another, simpler option is to use a "warped GP" strictly positive prior [12]. Inference in such models can be performed approximately by linearization [5]. However, our empirical studies suggest that the error is small overall, and can simply be ignored. The resulting overall procedure is given in Algorithm 1. It requires the storage of eight floating point numbers, whose update involves two exponential functions, one logarithm, and a handful of sums and products of floats. The procedure `Predict` returns the (logarithm of) the probability $p_{d(t_*)=D}$ that the download volume D will accumulate form the current time $t_i > t_{\text{new}}$ to

Algorithm 1 Probabilistic progress bar. Every time t_{new} a new rate y_{new} is observed `Inference` updates the posterior belief. Using its results, `Predict`(D, t_i, t_*) returns the likelihood that data volume D will accumulate from time t_i to time t_*. The routine `Initialise` sets prior assumptions.

```
1: procedure INITIALISE
2:     $t_{\texttt{old}} \leftarrow -\infty$, $y_{\texttt{old}} \leftarrow 0$, $A \leftarrow 0$, $\alpha \leftarrow 0$, $\gamma \leftarrow 0$, $\lambda \leftarrow 30[s], a_0 \leftarrow 0.1, b_0 \leftarrow 10^6$
3: end procedure
1: procedure INFERENCE($t_{\texttt{new}}, y_{\texttt{new}}$)
2:     $\Delta \leftarrow \exp[-(t_{\texttt{new}} - t_{\texttt{old}})/\lambda]$                ▷ scaled distance
3:     $N \leftarrow N + 1$                ▷ observation count
4:     $A \leftarrow A + \frac{1-\Delta}{1+\Delta}$                ▷ residual uncertainty on mean
5:     $\alpha \leftarrow \alpha + \frac{y_{\texttt{new}} - y_{\texttt{old}}\Delta}{1+\Delta}$                ▷ sufficient statistics for signal mean
6:     $\gamma \leftarrow \gamma + \frac{(y_{\texttt{new}} - y_{\texttt{old}}\Delta)^2}{1-\Delta^2}$                ▷ sufficient statistics for signal variance
7:     $t_{\texttt{old}} \leftarrow t_{\texttt{new}}, y_{\texttt{old}} \leftarrow y_{\texttt{new}}$
8:     $\beta \leftarrow \frac{\alpha}{A}$, $a \leftarrow a_0 + \frac{N-1}{2}$, $b \leftarrow b_0 + \frac{1}{2}\left(\gamma - \frac{\alpha^2}{A}\right)$
9: end procedure
1: procedure PREDICT($D, t_i, t_*; t_{\texttt{old}}, y_{\texttt{old}}, A, \beta, a, b$)
Ensure: $t_i > t_{\texttt{old}} \wedge t_* > t_i$                ▷ assumptions valid?
2:     $\delta_* \leftarrow t_* - t_i$
3:     $k_* \leftarrow \lambda[e^{-(t_i - t_{\texttt{old}})/\lambda} - e^{-(t_* - t_{\texttt{old}})/\lambda}]$
4:     $\kappa \leftarrow 2\lambda(\delta_* - k_*)$                ▷ prior variance of integral
5:     $\mu_* \leftarrow k_*(y_{\texttt{old}} - \beta) + \delta_*\beta$                ▷ post. mean
6:     $\sigma_*^2 \leftarrow \kappa - k_*^2 + (\delta_* - k_*)^2/A$                ▷ post. variance
7:     $\log p_{d(t_*)=D} \leftarrow -\left(a + \frac{1}{2}\right)\log\left[b + \frac{(d(t_*) - \mu_{t_*})^2}{2\sigma_{t_*}^2}\right] + \texttt{const.}$
8:     return $\log p_{d(t_*)=D}$                ▷ log Student-$t$ likelihood
9: end procedure
```

the target time t_*. This output can be used in two different ways, depending on the task and setting: To construct a graphical output for a *probabilistic progress bar*, as in Figures 1 and 2, we evaluate $p_{d(t_*)=D}$ over a set grid of values for t_* (doing so is less expensive than constructing the graphical output itself). If the grid is fine enough, this probability (multiplied with a regulariser, if needed) can also be used to compute mean, variance, and other moments of the distribution over run times. For applications requiring the *most likely* time of completion, this time can be found by an efficient optimization. This can be done very efficiently, because the domain is one-dimensional, and derivatives of $p_{d(t_*)=D}$ can be computed to high order, at diminishing cost. So very efficient numerical optimization techniques, such as Halley's method, are applicable, which we have empirically found to converge within one or two steps in this setting.

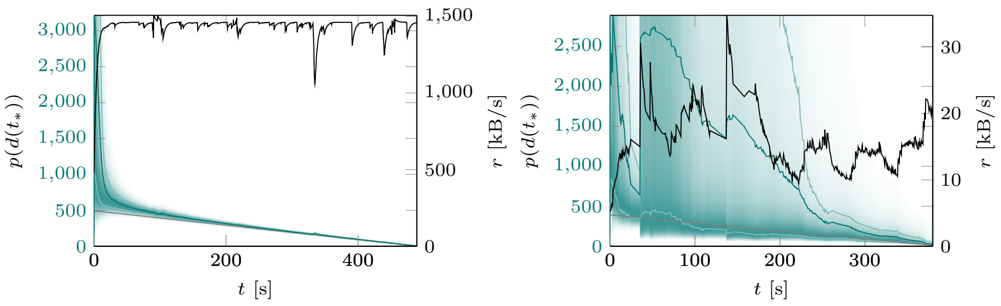

Fig. 3. Behaviour of the probabilistic error bar for two downloads. **Left:** a fast, consistent dial up connection for a 605Mb file. **Right:** a shaky cell phone connection for a 5Mb file. Download rates in black, scale on right ordinate. True remaining time of download as linear gray line (scale on left ordinate), probabilistic prediction shaded in green. The dark green line is the mean of the progress bar algorithm's posterior predictive distribution. The edges of the shaded regions mark the 10% and 90% quantiles of the predictive distribution.

4 Experiments

We tested our model both in the progress bar setting on several pre-recorded downloads (§4.1), and in a related task (see §1), predicting the time of a crowd-funding project to reach its target amount, on the Kickstarter platform (§4.2).

4.1 Progress Bars for Pre-Recorded Downloads

Figure 3 shows results from downloads of two single files in separate settings: A relatively large (605MB) file over a reliable pipe, and a small (5MB) file over an unreliable connection. See the figure caption for a description of the plots. A comparison between the predicted completion times (light green quantile shades) and ground truth (gray line) shows how the predictor converges to a good prediction, but also assigns meaningful uncertainty around its prediction. Note the strongly asymmetric form of the prediction, with median and most likely prediction typically close to the true value. The ramp-up phase early in the download is actually a feature of the download itself, and not of the estimation procedure (note that the rates r, plotted as black lines, are initially low).

4.2 Kickstarter

Kickstarter[3] is an online platform on which companies and individuals can ask for financial support for their projects. Every project chooses a deadline and a financial target, community members pledge money during this time window.

[3] `http://www.kickstarter.com`

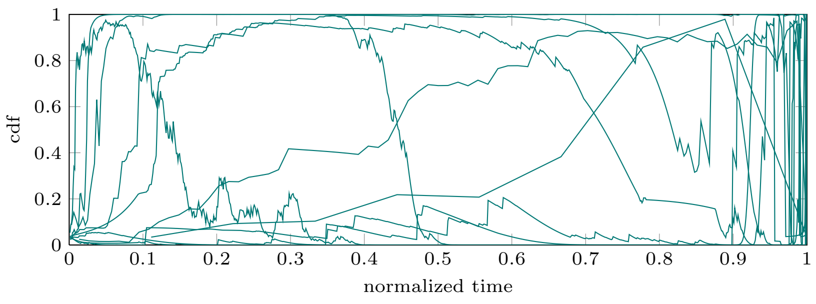

Fig. 4. Trajectories of ten randomly selected kickstarter projects scaled relative to the posterior cumulative distribution function (cdf) of the algorithm. If the model were perfect, these curves should be uniformly distributed across the [0,1] simplex.

If the set amount of money is reached within the time period, the project is considered successfully funded, and the pledged amount is transferred from to the project owners.

To get a sense of the algorithm's probabilistic calibration, we predicted the completion (or failure) time of kickstarter projects with a volume higher or equal to 20.000 USD from a dataset[4] in [4]. 300 projects were set aside as a training set to select the hyperparameters (a_0, b_0) of the prior (7). Independent of the success of the project we fixed the time window to either the point in time when the project got funded, or the original deadline occurred. The funding rate can then be approximated well from finite differences on the 1000 collected funding states.

Figure 4 shows, for 10 randomly selected projects, the position of the true finishing time within the algorithm's cumulative density function over the course of the pledge. The trajectories cover the entire range of the distribution, and often move through it over the course of the window, indicating good coverage of the distribution. At the same time, they are also not truly uniformly distributed, reflecting minor model flaws, the price paid for the low computational cost.

5 Conclusion

We derived a nonparametric algorithm for the probabilistic prediction of the completion time of a stochastically increasing process. Using a Gauss-Markov process prior with a parametric mean function, and analytically integrating over several hyperparameters, we arrived at a filtering algorithm of constant, very low cost, which nevertheless provides a nonparametric probabilistic prediction for the completion time. As pointed out in the introduction, such algorithms have numerous potential applications. One of them is to provide enhanced visual feedback on the progress of a file download in a web browser, a probabilistic error bar. An implementation of this method can be found on the project page[5].

[4] Dataset available at `http://sidekick.epfl.ch/data`.
[5] `http://people.tuebingen.mpg.de/mkiefel/projects/mlprogressbar/`

Bibliography

[1] Ajne, B., Daleniua, T.: Några tillämpningar av statistika ideer på numerisk integration. Nordisk Math. Tidskrift 8, 145–152 (1960)

[2] Bishop, C.: Pattern Recognition and Machine Learning. Springer (2006)

[3] Diaconis, P.: Bayesian numerical analysis. Statistical decision theory and related topics IV(1), 163–175 (1988)

[4] Etter, V., Grossglauser, M., Thiran, P.: Launch hard or go home!: Predicting the success of kickstarter campaigns. In: Proceedings of the First ACM Conference on Online Social Networks. pp. 177–182. COSN '13, ACM, New York, NY, USA (2013)

[5] Garnett, R., Osborne, M., Hennig, P.: Active learning of linear embeddings for Gaussian processes. In: Uncertainty in Artificial Intelligence (2014)

[6] Minka, T.: Deriving quadrature rules from Gaussian processes. Tech. rep., Statistics Department, Carnegie Mellon University (2000)

[7] Osborne, M., Duvenaud, D., Garnett, R., Rasmussen, C., Roberts, S., Ghahramani, Z.: Active learning of model evidence using bayesian quadrature. In: Advances in NIPS. pp. 46–54 (2012)

[8] Peres, S., Kortum, P., Stallmann, K.: Auditory progress bars: Preference, performance and aesthetics. In: Proceedings of the 13th International Conference on Auditory Display, Montreal, Canada, June 26. vol. 29, p. 2007 (2007)

[9] Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT (2006)

[10] Rogers, L., Williams, D.: Diffusions, Markov Processes and Martingales, vol. 1: Foundations. Cambridge, 2 edn. (2000)

[11] Särkkä, S.: Bayesian filtering and smoothing, vol. 3. Cambridge University Press (2013)

[12] Snelson, E., Rasmussen, C., Ghahramani, Z.: Warped Gaussian processes. In: Advances in Neural Information Processing Systems (2004)

[13] Uhlenbeck, G., Ornstein, L.: On the theory of the Brownian motion. Physical Review 36(5), 823 (1930)