Discovering bursts revisited: guaranteed optimization of the model parameters

Nikolaj Tatti
HIIT, Aalto University, Finland, nikolaj.tatti@aalto.fi

Abstract

One of the classic data mining tasks is to discover bursts, time intervals, where events occur at abnormally high rate. In this paper we revisit Kleinberg's seminal work, where bursts are discovered by using exponential distribution with a varying rate parameter: the regions where it is more advantageous to set the rate higher are deemed bursty. The model depends on two parameters, the initial rate and the change rate. The initial rate, that is, the rate that is used when there are no burstiness was set to the average rate over the whole sequence. The change rate is provided by the user.

We argue that these choices are suboptimal: it leads to worse likelihood, and may lead to missing some existing bursts. We propose an alternative problem setting, where the model parameters are selected by optimizing the likelihood of the model. While this tweak is trivial from the problem definition point of view, this changes the optimization problem greatly. To solve the problem in practice, we propose efficient $(1+\epsilon)$ approximation schemes. Finally, we demonstrate empirically that with this setting we are able to discover bursts that would have otherwise be undetected.

1 Introduction

Many natural phenomena occur unevenly over time, and one of the classic data mining tasks is to discover bursts, time intervals, where events occur at abnormally high rate. In this paper we revisit a seminal work by Kleinberg [13] that has been used, for example, in discovering trends in citation literature [3], analyzing topics [17], recommending citations [10], analyzing disasters [4], and analyzing social networks [1] and blogs [15].

Kleinberg [13] discovers bursts by modelling the time between events with an exponential model with varying rate parameter. The rate starts at the base level β and can be raised (multiple times) by a change parameter α, but it cannot descend β. Every time we raise the parameter, we need to pay a penalty. In the original approach, the change rate α is given as a parameter and the base rate is selected to be $\beta = 1/\mu$, where μ is the average of the sequence.

We argue that this choice of β is suboptimal: (i) it does not maximize the likelihood of the model, and, more importantly, (ii) a more optimized β may reveal bursts that would have gone undetected.

We propose a variant of the original burstiness problem, where we are no longer given the base parameter β but instead we are asked to optimize it along with discovering bursts. We also consider variants where we optimize α as well. These tweaks are rather mundane from the problem definition point of view but it leads to a surprisingly difficult optimization problem.

We consider two different models for the delays: exponential and geometric. First, we will show that we can solve our problem for exponential model in polynomial time, when α is given. Unfortunately, this algorithm requires $\mathcal{O}\left(n^3k^4\right)$ time,[1] thus being impractical. Even worse, we cannot apply the same approach for geometric model. This is a stark contrast to the original approach, where the computational complexity is $\mathcal{O}(nk)$.

Fortunately, we can estimate burst discovery in quasi-linear time w.r.t. the sequence length; see Table 1 for a summary of the algorithms. We obtain $(1+\epsilon)$ approximation guarantee for the geometric model. We also obtain, under some mild conditions, $(1+\epsilon)$ approximation guarantee for the exponential model.

In all four cases, the algorithm is simple: we test multiple values of β (and α), and use the same efficient dynamic program that is used to solve the original problem. Among the tested sequences we select the best one. The main technical challenge is to test the multiple values of α and β such that we obtain the needed guarantee while still maintaining a quasi-linear running time with respect to sequence length.

The remainder of the paper is as follows. We review the original burstiness problem in Section 2, and define our variant in Section 3. We introduce the exact algorithm in Section 4, and present the approximation algorithms in Section 5–6. In Section 7, we present the related work. In Section 8, we compare demonstrate empirically that our approach discovers busts that may go unnoticed. We conclude with discussion in Section 9. The proofs are given in Appendix, available in the full version of this paper.

[1] Here, n is the sequence length and k is the maximum number of times the rate can be increased.

Table 1: Summary of algorithms discussed in this paper. Here k is the number of allowed levels, n is the length of the sequence, μ is the arithmetic mean, and g is the geometric mean, Ω is the maximum of the sequence, and ω is the minimum of the sequence. We assume that $\omega > 0$. $\textsc{Exp}(\alpha, \beta)$ is the original problem considered by Kleinberg [13], and $\textsc{Geo}(\alpha, \beta)$ is a minor variation of the problem. The remaining results are the main contribution of this paper.

Problem	guarantee	running time
$\textsc{Exp}(\alpha, \beta)$	exact	$\mathcal{O}(nk)$
$\textsc{Exp}(\alpha)$	exact	$\mathcal{O}\big(n^3k^4\big)$
$\textsc{Exp}(\alpha)$	$SOL - n\log g \leq (1+\epsilon)(OPT - n\log g)$	$\mathcal{O}\big(\epsilon^{-1}nk^2\log\alpha\big)$
$\textsc{Exp}$	$SOL - n\log g \leq (1+\epsilon)(OPT - n\log g)$	$\mathcal{O}\big(\epsilon^{-2}nk^3\log^2(\Omega/\omega)\big)$
$\textsc{Geo}(\alpha, \beta)$	exact	$\mathcal{O}(nk)$
$\textsc{Geo}(\alpha)$	$SOL \leq (1+\epsilon)\,OPT$	$\mathcal{O}\big(\epsilon^{-1}nk\log\log n\big)$
$\textsc{Geo}$	$SOL \leq (1+\epsilon)\,OPT$	$\mathcal{O}\big(\epsilon^{-2}nk\log(n\mu k/\epsilon)\log\log n\big)$

2 Preliminaries

In this section, we review the setting proposed by Kleinberg [13], as well as the dynamic program used to solve this setting.

Assume that we observe an event at different time points, say $t_0, \ldots, t_n$. The main idea behind discovering bursts is to model the delays between the events, $s_i = t_i - t_{i-1}$: if the events occur at higher pace, then we expect s_i to be relatively small.

Assume that we are given a sequence of delays $S = s_1, \ldots, s_n$. In order to measure the burstiness of the sequence, we will model it with an exponential distribution, $p_{exp}(s; \lambda) = \lambda \exp(-\lambda s)$. Larger λ dictates that the delays should be shorter, that is, the events should occur at faster pace.

The idea behind modelling burstiness is to allow the parameter λ fluctuate to a certain degree: We start with $\lambda = \beta$, where β is a parameter. At any point we can increase the parameter by multiplying with another parameter α. We can also decrease the parameter by dividing by α. We can have multiple increases and decreases, however, we cannot decrease the parameter below β. Every time we change the rate from x to y, we have to pay a penalty, $\tau(x, y; \gamma)$, controlled by a parameter γ.

More formally, assume that we have assigned the burstiness levels for each delays $L = \ell_1, \ldots, \ell_n$, where each ℓ_i is a non-negative integer. We will refer to this sequence as the *level sequence*. For convenience, let us write $\ell_0 = 0$. Then the score of burstiness $q_{exp}(L, S; \alpha, \beta, \gamma)$ is equal to

$$\sum_{i=1}^{n} -\log p_{exp}\big(s_i; \beta\alpha^{\ell_i}\big) + \tau(\ell_{i-1}, \ell_i; \gamma) \quad .$$

The first term—negative log-likelihood of the data—measures how well the burstiness model fits the sequence, while the second term penalizes the erratic behavior in L. Ideally, we wish to have both terms as small as possible. To reduce clutter we will often ignore γ in notation, as this parameter is given, and is kept constant.

We will use the penalty function given in [13],

$$\tau(x, y) = \max(y - x, 0)\gamma \log n,$$

where n is the length of the input sequence. Note that τ depends on γ and n but we have suppressed this from the notation to avoid clutter.

We can now state the burstiness problem.

Problem 2.1 ($\textsc{Exp}(\alpha, \beta)$)**.** *Given a delay sequence S, parameters α, β, γ, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, where ℓ_i is an integer $0 \leq \ell_i \leq k$, minimizing $q_{exp}(L, S; \alpha, \beta, \gamma)$.*

Two remarks are in order: First of all, the original problem definition given by Kleinberg [13] does not directly use k, instead the levels are only limited implicitly due to τ. However, in practice, k is needed by the dynamic program, but it is possible to select a large enough k such that enforcing k does not change the optimal sequence [13]. Since our complexity analysis will use k, we made this constraint explicit. Secondly, the parameter β is typically set to $1/\mu$, where $\mu = \frac{1}{n}\sum s_i$ is the average delay.

We also study an altenative objective. Exponential distribution is meant primarily for real-valued delays. If the delays are integers, then the natural counterpart of the distribution is the geometric distribution $p_{geo}(s; \lambda) = (1 - \lambda)\lambda^s$. Here, *low* values of λ dictate that the delays should occur faster. We can now define

$$q_{geo}(L, S; \alpha, \beta, \gamma) = \sum_{i=1}^{n} -\log p_{geo}(s_i; \beta\alpha^{\ell_i}) + \tau(\ell_{i-1}, \ell_i) \ .$$

Note that in q_{exp} we use $\alpha > 1$ while here we use $\alpha < 1$. We can now define a similar optimization problem.

Problem 2.2 (GEO(α, β)). *Given an integer delay sequence S, parameters α, β, γ, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, where ℓ_i is an integer $0 \leq \ell_i \leq k$, minimizing $q_{geo}(L, S; \alpha, \beta, \gamma)$.*

We can solve Problem 2.1 or Problem 2.2 using the standard dynamic programming algorithm by Viterbi [18]. Off-the-shelf version of this algorithm requires $\mathcal{O}(nk^2)$ time. However, we can easily speed-up the algorithm to $\mathcal{O}(nk)$; for completeness we present this speed-up in Appendix A.

3 Problem definition

We are now ready to state our problem. The difference between our setting and Problem 2.1 is that here we are asked to optimize β, and possibly α, along with the levels, while in the original setting β was given as a parameter.

We consider two problem variants. In the first variant, we optimize β while we are given α.

Problem 3.1 (EXP(α)). *Given a delay sequence S, parameters α, γ, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, where ℓ_i is an integer $0 \leq \ell_i \leq k$,* and a parameter β, *minimizing $q_{exp}(L, S; \alpha, \beta, \gamma)$.*

In the second variant, we optimize both α and β.

Problem 3.2 (EXP). *Given a delay sequence S, a parameter γ, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, where ℓ_i is an integer $0 \leq \ell_i \leq k$,* and parameters α and β, *minimizing $q_{exp}(L, S; \alpha, \beta, \gamma)$.*

While this modification is trivial and mundane from the problem definition point of view, it carries several crucial consequences. First of all, optimizing β may discover bursts that would otherwise be undetected.

Example 3.1. *Consider a sequence given in Figure 1, which shows a sequence of* 500 *delays. The burst between 100 and 400 is generated using exponential model with $\lambda = 1/2$, the remaining delays are generated using $\lambda = 1$. We applied Viterbi with β^{-1} equal to the average of the sequence, the value used by Kleinberg [13], and compare it to $\beta = 1/2$, which is the correct ground level of the generative model. The remaining parameters were set to $\alpha = 2$, $\gamma = 1$, and $k = 1$. We see that in the latter case we discover a burst that is much closer to the ground truth.* □

Our second remark is that if α and β are given, one can easily discover the optimal bursts using *Viterbi*.

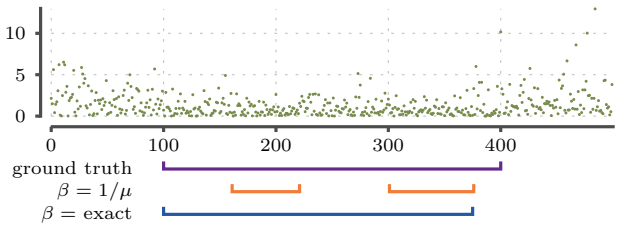

Figure 1: A toy data set S with a burst between 100 and 400. Low values indicate short delays, bursts. The indicated regions are (*i*) the ground truth, (*ii*) bursts discovered with $\beta = 1/\mu$, where μ is the average delay, and (*iii*) bursts discovered with β set to the exact value of the generative model.

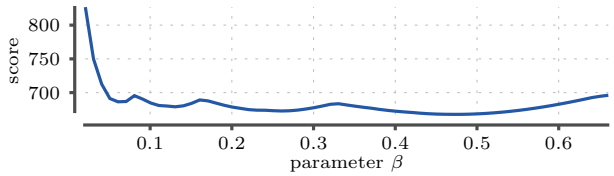

Figure 2: Score $p_{exp}(S, L^*; \alpha, \beta, k)$ as a function of β, where $\alpha = 2$, $k = 4$, and L^* is the optimal level sequence for the given parameters. Low values are better.

The optimization becomes non-trivial when we need to optimize α and β as well. To make matters worse, the score as a function of β is non-convex, as demonstrated in Figure 2. Hence, we can easily get stuck in local minima.

Next, we introduce discrete variants of EXP(α) and EXP.

Problem 3.3 (GEO(α)). *Given an integer delay sequence S, parameters α, γ, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, where ℓ_i is an integer $0 \leq \ell_i \leq k$,* and a parameter β, *minimizing $q_{geo}(L, S; \alpha, \beta, \gamma)$.*

Problem 3.4 (GEO). *Given an integer delay sequence S, a parameter α, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, where ℓ_i is an integer $0 \leq \ell_i \leq k$,* and parameters α and β, *minimizing $q_{geo}(L, S; \alpha, \beta, \gamma)$.*

Despite being very similar problems, we need to analyze these problems individually. We will show that EXP(α) can be solved exactly in polynomial time, although, the algorithm is too slow for practice. This approach does not work for other problems but we will show that all four problems can be $(1+\epsilon)$-approximated efficiently.

Before we continue, we need to address a pathological case when solving Exp: the problem of Exp is illdefined if the delay sequence S contains a zero. To see this, assume that $s_i = 0$. Then a level sequence $\ell_i = 1$, and $\ell_j = 0$, for $j \neq i$, with $\alpha = \infty$ and $\beta = 1$ leads to a score of $-\infty$. This is because $p_{exp}(s_i; \beta\alpha) = \infty$ and the remaining terms are finite. This is why we assume that whenever we deal with Exp, we have $s_i > 0$. If we have $s_i = 0$, then we can either set α manually by using Exp(α) or shift the delays by a small amount.

4 Exact algorithm for Exp(α)

In this section we present an exact polynomial algorithm for solving Exp(α). Unfortunately, this algorithm is impractically slow for large sequences: the time complexity is $\mathcal{O}(n^3k^4)$ and the space complexity is $\mathcal{O}(n^3k^3)$. Thus, it only serves as a theoretical result. More practical algorithms are given in the next sections.

In order to solve Exp we introduce a more complicated optimization problem.

Problem 4.1 (BndBurst)**.** *Given a delay sequence $S = s_1, \ldots, s_n$, a parameter α, budget parameters d and m, and a maximum number of levels k, find a level sequence $L = \ell_1, \ldots, \ell_n$, with $0 \leq l_i \leq k$, minimizing*

$$\sum_{i=1}^{n} \alpha^{l_i} s_i \quad \textit{such that}$$
$$\sum_i \max(\ell_i - \ell_{i-1}, 0) = d \quad \textit{and} \quad \sum_i \ell_i = m \quad .$$

We will show that this problem can be solved in polynomial time. But before, let us first show that Exp and BndBurst are intimately connected. See Appendix B for the proof.

Proposition 4.1. *Assume a delay sequence S, and parameters α and γ, and an upper bound for levels k. There are budget parameters $d \leq k(n+1)/2$ and $m \leq kn$ for which the level sequence L solving* BndBurst *also solves* Exp(α) *along with*

$$\beta = \frac{n}{\sum_i s_i \alpha^{\ell_i}} \quad .$$

We can solve BndBurst with a dynamic program. In order to do this, let us define a table o, where an entry $o[i, j, a, b]$ is the optimal score of the first i symbols of the input sequence such that

$$\ell_i = j, \ \sum_{x=1}^{i} \max(\ell_x - \ell_{x-1}, 0) = a, \text{ and } \sum_{x=1}^{i} \ell_x = b \quad .$$

In case, there is no level sequence satisfying the constraints, we set $o[i, j, a, b] = \infty$. Due to Proposition 4.1, we can limit $a \leq k(n+1)/2$ and $b \leq kn$. Consequently, o contains $\mathcal{O}(n^3k^3)$ entries. We can compute a single entry with

(4.1)
$$o[i, j, a, b] = \alpha^j s_i + \min_{j'} o[i-1, j', a - \max(0, j - j'), b - j] \quad .$$

The computation of a single value thus requires $\mathcal{O}(k)$ time. So computing the whole table can be done in $\mathcal{O}(n^3k^4)$. Moreover, if we also store the optimal j' as given in Equation 4.1, for each cell, we can recover the level sequence responsible for every $o[i, j, a, b]$.

Proposition 4.1 now guarantees that we can solve Exp by comparing the level sequences responsible for $o[n, j, a, b]$, where $j = 0, \ldots, k$, $a = 0, \ldots, (k+1)n/2$, and $b = 0, \ldots, kn$.

5 Approximating discrete burstiness

In this section we will provide a $(1+\epsilon)$-approximation algorithms for Geo(α) and Geo. The time complexities are stated in Table 1.

5.1 Approximating Geo(α) Note that if we knew the optimal β, then Geo(α) reduces to Problem 2.2, which we can solve in $\mathcal{O}(nk)$ time by applying *Viterbi*. The idea behind our approximation is to test several values of β, and select the best solution among the tested values. The trick is to select values densely enough so that we can obtain $(1+\epsilon)$ guarantee while keeping the number of tests low, namely $\mathcal{O}(\epsilon^{-1} \log\log n)$. The pseudo-code of the algorithm is given in Algorithm 1.

Algorithm 1: *GeoAlpha*$(S, \alpha, \gamma, k, \epsilon)$

```
μ ← (1/n) Σ_i s_i;
if μ = 0 then return L = (0, ..., 0) ;
η ← μ/(μ + 1);
c ← 1;
while η^c ≤ μ/(μ + 1/n) do
    β ← η^c;
    L ← Viterbi(S, α, β, γ, k, p_geo);
    c ← c/(1 + ε);
return the best observed L;
```

Next we state that the algorithm indeed yields an $(1+\epsilon)$-approximation ratio, and can be executed in $\mathcal{O}(\epsilon^{-1}nk \log\log n)$ time. The proofs are given in Appendix C–D.

Proposition 5.1. *Let S be an integer delay sequence, and let α, γ, and k be the parameters. Let L^*, β^* be*

the solution to GEO(α). *Assume* $\epsilon > 0$. *Let* L, β *be the solution returned by* $GeoAlpha(S, \alpha, \gamma, k, \epsilon)$. *Then*

$$q_{geo}(S, L; \beta) \leq (1+\epsilon) q_{geo}(S, L^*; \beta^*) \quad .$$

Proposition 5.2. *The computational complexity of GeoAlpha is* $\mathcal{O}\big(\epsilon^{-1} nk \log\log n\big)$.

5.2 Approximating Geo We now turn to approximating GEO. The approach here is similar to the previous approach: we test multiple values of α and invoke *GeoAlpha*. The pseudo-code for the algorithm is given in Algorithm 2.

Algorithm 2: $ApproxGeo(S, \gamma, k, \epsilon)$

```
L ← GeoAlpha(S, 0, γ, k, ε);
μ ← (1/n) Σ_i s_i;
η ← 1/(1 + nk);
σ ← μ/(μ + 1/n);
c ← 1;
while η^c ≤ σ^{ε/k} do
    α ← η^c;
    L ← GeoAlpha(S, α, γ, k, ε);
    c ← c/(1 + ε);
return the best observed L;
```

Next we establish the correctness of the method as well as the running time. The proofs are given in Appendix E–F.

Proposition 5.3. *Let* L^*, α^*, β^* *be the solution to* GEO. *Assume* $\epsilon > 0$. *Let* L, α, β *be the solution returned by* $ApproxGeo(S, \gamma, k, \epsilon)$. *Then*

$$q_{geo}(S, L; \alpha, \beta) \leq (1+\epsilon) q_{geo}(S, L^*; \alpha^*, \beta^*) \quad .$$

Proposition 5.4. *The computational complexity of ApproxGeo is*

$$\mathcal{O}\big(nk \log\log n (\log n + \log\mu + \log k - \log\epsilon)\epsilon^{-2}\big) \quad .$$

6 Approximating continuous burstiness

In this section we will provide a $(1+\epsilon)$-approximation algorithms for EXP(α) and EXP. The time complexities are stated in Table 1.

6.1 Approximating Exp(α) In this section we introduce an approximation algorithm for EXP(α). The general approach of this algorithm is the same as in *GeoAlpha*: we test several values of β, solve the resulting subproblem with *Viterbi*, and select the best one. The pseudo-code is given in Algorithm 3.

Unlike with *GeoAlpha*, *ExpAlpha* does not yield an unconditional $(1+\epsilon)$-approximation guarantee. The key problem is that since exponential distribution is continuous, the term $p_{exp}(s; \lambda)$ may be larger than 1. Consequently, $-\log p_{exp}(s; \lambda)$, as well as the actual score q_{exp}, can be negative. However, if the delay sequence has a geometric mean larger or equal than 1, we can guarantee the approximation ratio.

Algorithm 3: $ExpAlpha(S, \alpha, \gamma, k, \epsilon)$

```
μ ← (1/n) Σ_i s_i;
β ← 1/μ;
while β ≥ 1/(α^k μ) do
    L ← Viterbi(S, α, β, γ, k, p_exp);
    β ← β/(1 + ε);
return the best observed L and β;
```

The proofs for the next two propositions are given in Appendix G–H.

Proposition 6.1. *Assume a delay sequence* S, *and parameters* α *and* γ, *and an upper bound for levels* k. *Let* β^* *and* L^* *be the solution to* EXP($\alpha(\alpha)$). *Let* $g = [\prod_i s_i]^{1/n}$ *be the geometric mean. Assume* $\epsilon > 0$. *Let* L, β *be the solution returned by ExpAlpha. Then*

$$q_{exp}(S, L; \beta) - n\log g \leq (1+\epsilon)(q_{exp}(S, L^*; \beta^*) - n\log g) \quad .$$

Moreover, if $g \geq 1$, *then*

$$q_{exp}(S, L; \beta) \leq (1+\epsilon) q_{exp}(S, L^*; \beta^*) \quad .$$

Note that if the geometric mean g is less than 1, then we still have a guarantee, except now we need to shift the score by a (positive) constant of $-n\log g$.

Proposition 6.2. *The computational complexity of ExpAlpha is* $\mathcal{O}\big(\epsilon^{-1} nk^2 \log\alpha\big)$.

6.2 Approximating Exp We now turn to approximating EXP. The approach here is similar to the previous approach: we test multiple values of α and invoke *ExpAlpha*. The pseudo-code for the algorithm is given in Algorithm 4.

Algorithm 4: $ApproxExp(S, \gamma, k, \epsilon)$

```
α ← (max s_i)/(min s_i);
c ← ∛(1 + ε);
while α ≥ 1 do
    L ← ExpAlpha(S, α, γ, k, ε/2);
    α ← α/c;
return the best observed L;
```

Next we establish the correctness of the method as well as the running time. The proofs given in Appendix I–J.

Proposition 6.3. *Assume a delay sequence S, a parameter γ, and an upper bound for levels k. Let α^*, β^* and L^* be the solution to* Exp. *Let $g = [\prod_i s_i]^{1/n}$ be the geometric mean, and let $\psi = n \log g$. Assume $\epsilon > 0$. Let L, α, β the solution returned by ApproxExp. Then*

$$q_{exp}(S, L; \alpha, \beta) - \psi \leq (1+\epsilon)(q_{exp}(S, L^*; \alpha^*, \beta^*) - \psi) \quad .$$

Moreover, if $g \geq 1$, then

$$q_{exp}(S, L; \alpha, \beta) \leq (1+\epsilon) q_{exp}(S, L^*; \alpha, \beta^*) \quad .$$

Proposition 6.4. *Let $\Omega = \max s_i$ and let $\omega = \min s_i$. The computational complexity of ApproxExp is $\mathcal{O}\big(\epsilon^{-2} n k^3 \log^2(\Omega/\omega)\big)$.*

6.3 Speeding up Exp(α) Our final step is to describe how can we speed-up the computation of Exp(α) in practice. The following proposition allows us to ignore a significant amount of tests.

Proposition 6.5. *Assume a delay sequence S, and parameters α and γ. Let β be a parameter, and let L be the optimal solution for* Exp(α, β). *Define*

$$\beta' = \frac{n}{\sum_i s_i \alpha^{\ell_i}} \quad .$$

Let β^* be the optimal parameter to Exp(α). *Then either*

$$\beta^* \leq \min(\beta, \beta') \quad \textit{or} \quad \beta^* \geq \max(\beta, \beta') \quad .$$

Proposition 6.5 allows us to ignore some tests: Let β_i be the parameters tested by *ExpAlpha*, that is, $\beta_i = \mu^{-1}(1+\epsilon)^{-i}$. Assume that we test β_i, and compute β' as given in Proposition 6.5. If $\beta' > \beta_i$, we can safely ignore testing any β_j such that $\beta_i < \beta_j < \beta'$. Similarly, if $\beta' < \beta_i$, we can safely ignore testing any β_j such that $\beta' < \beta_j < \beta_i$.

The testing order of β_i matters since we want to use both cases $\beta' < \beta_i$ and $\beta' > \beta_i$ efficiently. We propose the following order which worked well in our experimental evaluation: Let t be the number of different β_i, and let m be the largest integer for which $2^m \leq t$. Test the parameters in the order

$$0, 2^m, 2^{m-1}, 2^{3(m-1)}, \ldots, 1, 3, 5, 7, \ldots,$$

that is, we start with 0 and increment by 2^m until we reach the end of the list. Then we decrease m by 1, and repeat. During the traverse, we ignore the parameters that were already tested, as well as the redundant parameters.

Interestingly enough, this approach cannot be applied directly to the discrete version of the problem. First of all, the technique for proving Proposition 6.5 cannot be applied directly to the score function for the geometric distribution. Secondly, there is no closed formula for computing the discrete analogue of β' given in Proposition 6.5.

7 Related work

Discovering bursts Modelling and discovering bursts is a very well-studied topic in data mining. We will highlight some existing techniques. We are modelling delays between events, but we can alternatively model event counts in some predetermined window: high count indicate burst. Ihler et al. [11] proposed modelling such a statistic with Poisson process, while Fung et al. [5] used Binomial distribution. If the events at hand are documents, we can model burstiness with time-sensitive topic models [12, 14, 20]. As an alternative methods to discover bursts, Zhu and Shasha [21] used wavelet analysis, Vlachos et al. [19] applied Fourier analysis, and He and Parker [9] adopted concepts from Mechanics. Lappas et al. [16] propose discovering maximal bursts with large discrepancy.

Segmentation A sister problem of burstiness is a classic segmentation problem. Here instead of penalizing transitions, we limit the number of segments to k. If the overall score is additive w.r.t. the segments, then this problem can be solved in $\mathcal{O}\big(n^2 k\big)$ time [2]. For certain cases, this problem has a linear time solution [6]. Moreover, under some mild assumptions we can obtain a $(1+\epsilon)$ approximation in linear time [8].

Concept drift detection in data streams: A related problem setting to burstiness is concept drift detection. Here, a typical goal is to have an online algorithm that can perform update quickly and preferably does not use significant amount of memory. For an overview of existing techniques see an excellent survey by Gama et al. [7]. The algorithms introduced in this paper along with the original approach are not strictly online because in every case we need to know the mean of the sequence. However, if the mean is known, then we can run *Viterbi* in online fashion, and, if we are only interested in the burstiness of a current symbol, we need to maintain only $\mathcal{O}(k)$ elements, per β.

8 Experimental evaluation

In this section we present our experiments. As a baseline we use method by Kleinberg [13], that is, we derive the parameter β from μ, the mean of the sequence. For exponential model, $\beta = \mu^{-1}$; we refer to this model as *ExpMean*. For geometrical model, $\beta = \mu/(\mu+1)$; we refer to this approach as *GeoMean*. Throughout the experiments, we used $\epsilon = 0.05$ and $\gamma = 1$ for our algorithms.

Experiments with synthetic data: We first focus on demonstrating when optimizing β is more advantageous than the baseline approach.

For our first experiment we generated a sequence of 500 data points. We planted a single burst with a varying length 50–250. The burst was generated with

$p_{exp}(\cdot; 1)$, while the remaining sequence was generated with $p_{exp}(\cdot; 2)$. We computed bursts with *ExpMean* and *ExpAlpha*, the parameters were set to $k = 1$, $\alpha = 2$. The obtained level sequence was evaluated by computing the hamming distance, $\sum_i |\ell_i - \ell_i^*|$, where ℓ_i^* is the ground truth level sequence. We repeated each experiment 100 times.

We see from the results given in Figure 3 that the bursts discovered by *ExpAlpha* are closer to the ground truth, on average, than the baseline. This is especially the case when burst becomes larger. The main reason for this is that short bursts do not affect significantly the average of the sequence, μ, so consequently, μ is close to the base activity level. As the burst increases, so does μ, which leads to underestimating of β.

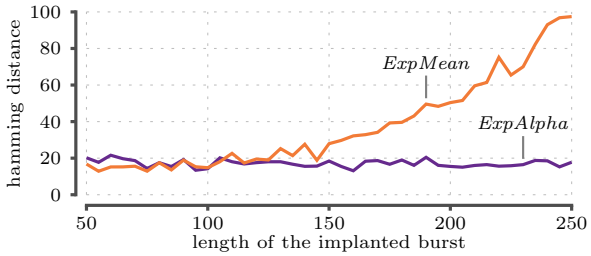

Figure 3: Hamming distance between the ground truth and the discovered level sequence as a function of the length of the planted burst. Low values are better.

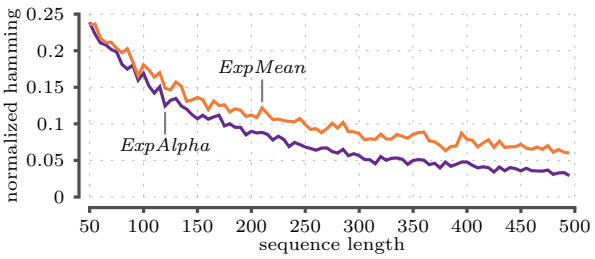

Figure 4: Hamming distance, normalized by the sequence length, between the ground truth level sequence and the discovered level sequence as a function of the sequence length. Low values are better.

Our next experiment is similar, expect now we vary the sequence length, n, (50–500) and set the burst length to be $n/3$. We generated the sequence as before, and we use the same parameters. In Figure 4 we report, $\frac{1}{n}\sum_i |\ell_i - \ell_i^*|$, the number of disagreements compared with the ground truth, normalized by n. Each experiment was repeated 300 times.

We see that for the shortest sequences, the number of disagreement is same for both algorithm, around 0.2–0.25. This is due that we do not have enough samples to override the transition penalty τ. Once the sequence becomes longer, we have more evidence of a burst, and here *ExpAlpha* starts to beat the baseline, due to a better model fit.

Experiments with real-world data: We considered two datasets: The first dataset, *Crimes*, consists of 17 033 crimes related to narcotics in Chicago between January and October, 2015. The second dataset, *Mine*, consists of 909 fatalities in U.S. mining industry dating from 2000, January.[2] This data is visualized in Figure 6. In both datasets, each event has a time stamp: in *Crimes* we use minutes as granularity, whereas in *Mine* the time stamp is by the date. Using these time stamps, we created a delay sequence.

We applied *ApproxExp*, *ExpAlpha*, and *ExpMean* to *Crimes*. We set $k = 4$, and for *ExpAlpha* and *ExpMean* we used $\alpha = 2$. Since *Crimes* contains events with 0 delay, we added 1 minute to each delay to avoid the pathological case described in Section 3. The obtained bursts are presented in Figure 5. We also applied *ApproxGeo*, *GeoAlpha*, and *GeoMean* to *Mine*. Here we set $\alpha = 1/2$ and $k = 4$, however the algorithm used only 3 levels. The obtained bursts are presented in Figure 7.

In *Mine*, the results by *GeoAlpha* and *GeoMean* are the same. However, we noticed that the results differ if we use different α. The biggest difference between *ApproxGeo* and *GeoAlpha* is the last burst: *GeoAlpha* (and *GeoMean*) set the last burst to be on level 2, while *ApproxGeo* uses level 1. The reason for this is that *ApproxGeo* selects α to be very close to 0, that is, much smaller than 1/2, the parameter used by the other algoritms. This implies that when going one level up, the model expects the events to be much closer to each other.

In *Crimes*, *ApproxExp* and *ExpAlpha* discover burstier structure than *ExpMean*. *ExpAlpha* uses 4 different levels. Interestingly enough, in this level sequence, we spent most of the time at level 1, and we descended to level 0 for 3 short bursts. In other words, in addition to finding crime streaks, *ExpAlpha* also found three short periods when narcotics related crime rate was lower than usual. *ApproxExp* also spends most of its time on level 1 but often descends on level 0, while also highlighting one burst in early January.

Number of *Viterbi* calls: Next, we study relative efficiency when compared *Viterbi*. Since all 4 approximation schemes use *Viterbi* as a subroutine, a natural way of measuring the efficiency is to study the number of *Viterbi* calls. We report the number of calls as a

[2]Both datasets are available at `http://data.gov/`.

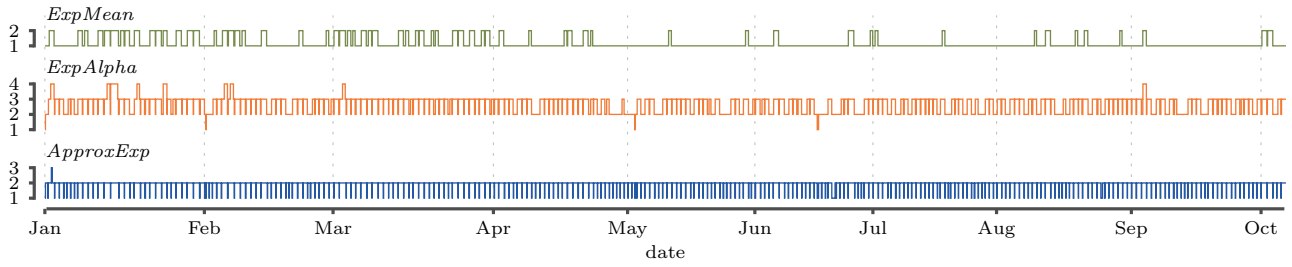

Figure 5: Discovered bursts in *Crimes* dataset.

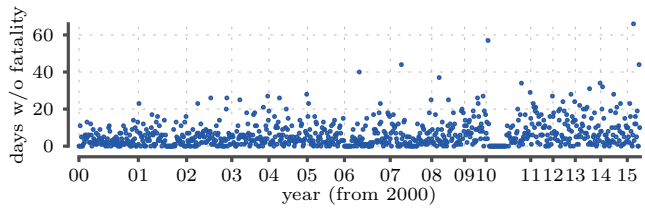

Figure 6: The delay sequence *Mine*, as well as the discovered bursts.

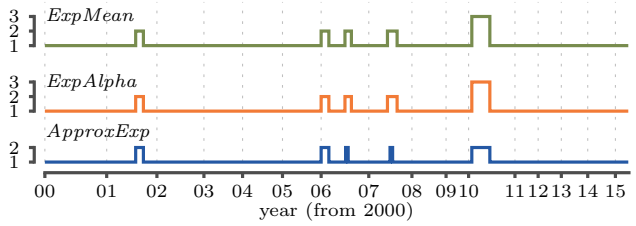

Figure 7: The delay sequence *Mine*, as well as the discovered bursts.

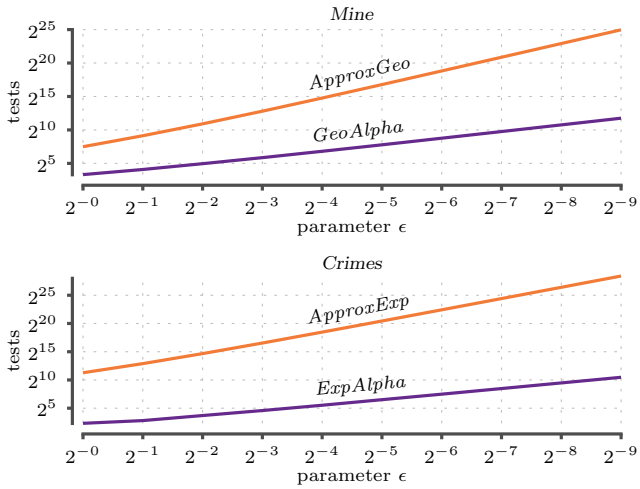

Figure 8: Number of *Viterbi* calls as a function of ϵ. Both x and y-axis are logarithmic. We set $k = 4$, $\alpha = 0.5$ for *ExpAlpha*, and $\alpha = 2$ for *ExpAlpha*. Here, we did not use the speed-up version of *ExpAlpha*.

function of ϵ for datasets *Mine* and *Crimes* in Figure 8. Here, we did not use the speed-up version of *ExpAlpha*.

We see that the behaviour depends heavily on the accuracy parameter ϵ: for example, if we use $\epsilon = 0.5$, then *GeoAlpha* uses 17 calls while *ApproxGeo* uses 561 calls; if we set $\epsilon = 2^{-9}$, then *GeoAlpha* needs 3453 calls while *ApproxGeo* needs 32 810 406 calls. This implies that we should not use extremely small ϵ, especially if we also wish to optimize α. Nevertheless, the algorithms are fast when we use moderately small ϵ.

Effect of a speed-up: Finally, we compare the effect of a speed-up for *ExpAlpha* described in Section 6. Here we used both datasets *Mine* and *Crimes* to which we apply *ExpAlpha* with $k = 5$ and $\alpha = 2$. We vary ϵ from 2^{-13} to $1/2$ and compare the plain version vs. speed-up in Figure 9.

We see in Figure 9 that the we gain significant speed-up as we decrease ϵ: At best, we improve by two orders of magnitude.

9 Concluding remarks

In this paper we presented variants of [13] for discovering bursts: instead of deriving the base rate from μ, the average delay time between the events, we optimize this parameter along with the actual burst discovery. We showed that this leads to better burst discovery, especially if the bursts are long. We also propose variants, where we optimize the change parameter α, instead of having it as a parameter.

Despite being a minor tweak, the resulting optimization problems are significantly harder. To solve the problems, we introduce efficient algorithms yielding $(1 + \epsilon)$ approximation guarantee. These methods are based on testing multiple values for the base rate, and selecting the burst sequence with the best score. Despite being similar problems, discrete and continuous versions of the problem required their own algorithms. In addition, we were able significantly speed-up the exponential model variant by safely ignoring some candidate values

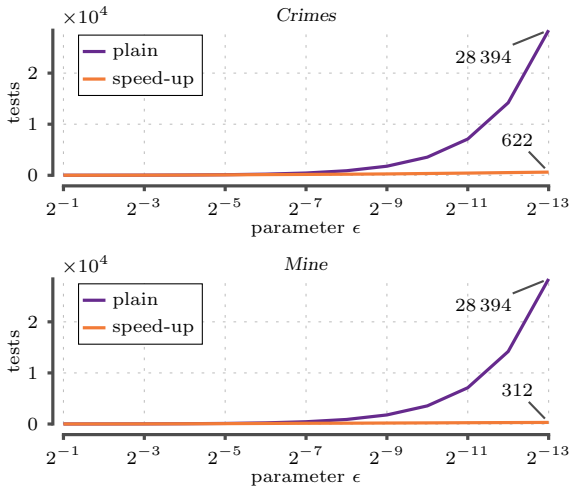

Figure 9: Number of tests needed as a function of ϵ. Speed-up (see, Section 6) vs. vanilla version.

for the base rate.

The approximation algorithms are quasi-linear with respect to sequence length. However, especially when we optimize α, the algorithms depend also on the actual values of the sequence, see Table 1. A potential future work is to improve the algorithms, and develop polynomially strong approximation schemes. The other fruitful direction is to develop heuristics that allow us to ignore large parts of the parameters, similar to the speed-up we propose for the exponential model variant of the problem.

References

[1] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *KDD*, pages 44–54, 2006.

[2] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6), 1961.

[3] C. Chen. Citespace II: Detecting and visualizing emerging trends and transient patterns in scientific literature. *J. Am. Soc. Inf. Sci. Technol.*, 57(3): 359–377, 2006.

[4] R. Fontugne, K. Cho, Y. Won, and K. Fukuda. Disasters seen through flickr cameras. In *SWID*, 2011.

[5] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *VLDB*, pages 181–192, 2005.

[6] Z. Galil and K. Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Inf. Process. Lett.*, 33(6):309–311, 1990.

[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[8] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *TODS*, 31(1):396–438, 2006.

[9] D. He and D. S. Parker. Topic dynamics: An alternative model of bursts in streams of topics. In *KDD*, 2010.

[10] Q. He, D. Kifer, J. Pei, P. Mitra, and C. L. Giles. Citation recommendation without author supervision. In *WSDM*, pages 755–764, 2011.

[11] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *KDD*, pages 207–216, 2006.

[12] N. Kawamae. Trend analysis model: Trend consists of temporal words, topics, and timestamps. In *WSDM*, pages 317–326, 2011.

[13] J. Kleinberg. Bursty and hierarchical structure in streams. *DMKD*, 7(4):373–397, 2003.

[14] A. Krause, J. Leskovec, and C. Guestrin. Data association for topic intensity tracking. In *ICML*, pages 497–504, 2006.

[15] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *WWW*, pages 568–576, 2003.

[16] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. On burstiness-aware search for document sequences. In *KDD*, pages 477–486, 2009.

[17] K. K. Mane and K. Börner. Mapping topics and topic bursts in PNAS. *PNAS*, 101(suppl 1):5287–5290, 2004.

[18] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE IT*, 13(2):260–269, 1967.

[19] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, pages 131–142, 2004.

[20] X. Wang and A. McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *KDD*, pages 424–433, 2006.

[21] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD*, pages 336–345, 2003.