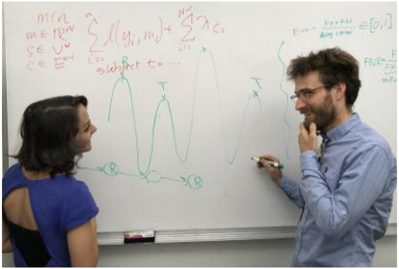
Why does functional pruning yield such fast algorithms for optimal changepoint detection?

Toby Dylan Hocking — toby.hocking@nau.edu
Northern Arizona University
School of Informatics, Computing, and Cyber Systems
Machine Learning Research Lab — http://ml.nau.edu



Come to Flagstaff! Graduate Research Assistantships available!

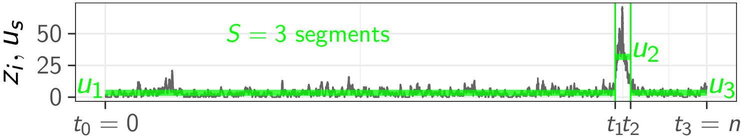
Classical dynamic programming for optimal changepoint detection

Functional pruning algorithms

Empirical time complexity

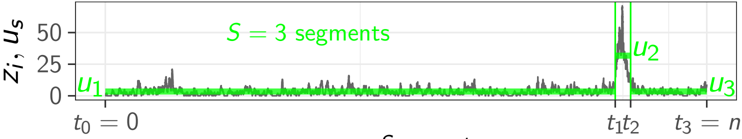
Theoretical time complexity

Statistical model is a piecewise constant mean



- We have n data $z_1, \dots, z_n \in \mathbb{Z}_+$.
- Fix the number of segments $S \in \{1, 2, \dots, n\}$.
- Optimization variables: $S - 1$ changepoints $t_1 < \cdots < t_{S-1}$ and S segment means $u_1, \dots, u_S \in \mathbb{R}_+$.
- Let $0 = t_0 < t_1 < \cdots < t_{S-1} < t_S = n$ be the segment limits.
- Statistical model: for every segment $s \in \{1, \dots, S\}$, $z_i \overset{\text{iid}}{\sim} \text{Poisson}(u_s)$ for every data point $i \in (t_{s-1}, t_s]$ implies convex loss function $\ell(u, z) = u - z \log u$ to minimize.
- Other models: real-valued $z_i \overset{\text{iid}}{\sim} N(u_s, \sigma^2)$ implies square loss $\ell(u, z) = (u - z)^2$, etc.

Maximum likelihood inference for S segments and n data is a non-convex minimization problem



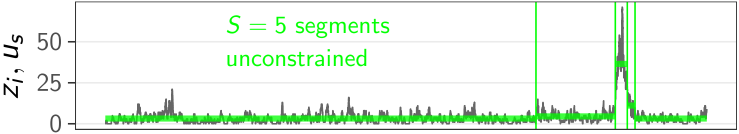
$$\mathcal{L}_{S,n} = \min_{\substack{\mathbf{u}\in\mathbb{R}^S\\ 0=t_0<t_1<\cdots<t_{S-1}<t_S=n}} \sum_{s=1}^{S} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, z_i)$$

$$= \min_{t_{S-1}} \underbrace{\min_{\substack{u_1,\dots,u_{S-1}\\ t_1<\cdots<t_{S-2}}} \sum_{s=1}^{S-1} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, z_i)}_{\mathcal{L}_{S-1,t_{S-1}}} + \underbrace{\min_{u_S} \sum_{i=t_{S-1}+1}^{t_S=n} \ell(u_S, z_i)}_{c_{(t_{S-1},t_S=n]}}$$

- Hard optimization problem, naively $O(n^S)$ time.
- Auger and Lawrence (1989): $O(Sn^2)$ time classical dynamic programming algorithm:

$$\mathcal{L}_{s,t} = \min_{t'<t} \mathcal{L}_{s-1,t'} + c_{(t',t]}$$

Maximum likelihood inference for S segments and n data is a non-convex minimization problem



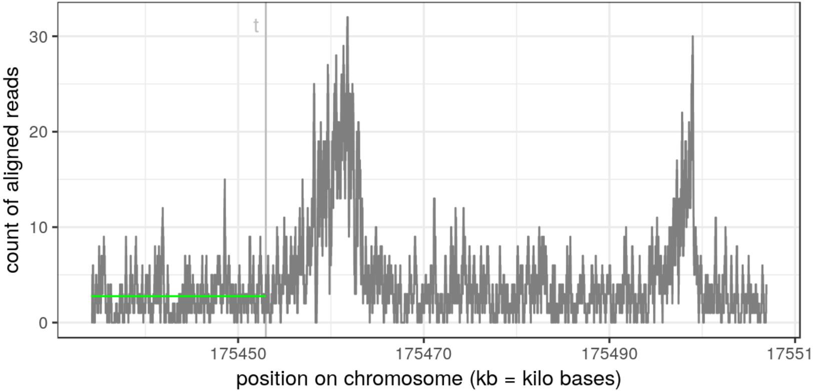
$$\mathcal{L}_{S,n} = \min_{\substack{\mathbf{u}\in\mathbb{R}^S \\ 0=t_0<t_1<\cdots<t_{S-1}<t_S=n}} \sum_{s=1}^{S} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, z_i)$$

$$= \min_{t_{S-1}} \underbrace{\min_{\substack{u_1,\dots,u_{S-1} \\ t_1<\cdots<t_{S-2}}} \sum_{s=1}^{S-1} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, z_i)}_{\mathcal{L}_{S-1,t_{S-1}}} + \underbrace{\min_{u_S} \sum_{i=t_{S-1}+1}^{t_S=n} \ell(u_S, z_i)}_{c_{(t_{S-1},t_S=n]}}$$

- Hard optimization problem, naively $O(n^S)$ time.
- Auger and Lawrence (1989): $O(Sn^2)$ time classical dynamic programming algorithm:

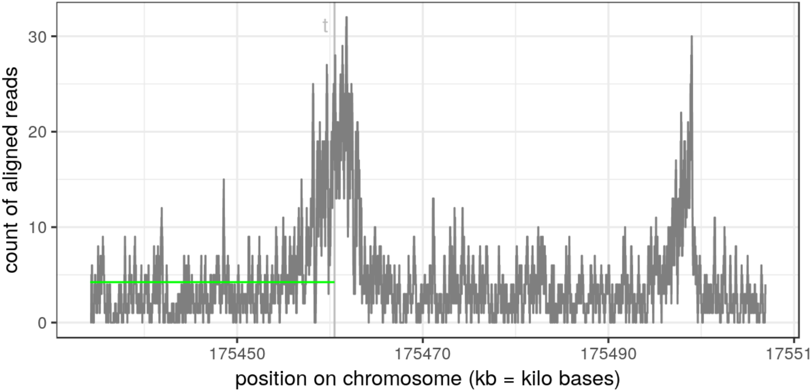
$$\mathcal{L}_{s,t} = \min_{t'<t} \mathcal{L}_{s-1,t'} + c_{(t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 1$ segments up to data point t



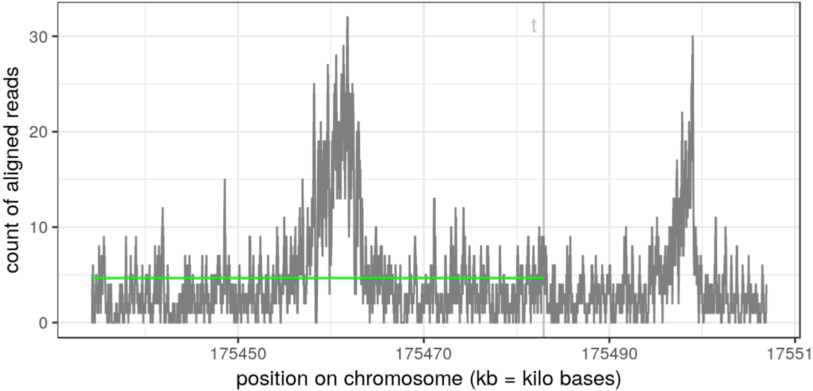
$$\mathcal{L}_{1,t} = \underbrace{c_{(0,t]}}_{\text{optimal loss of 1st segment } (0,t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 1$ segments up to data point t



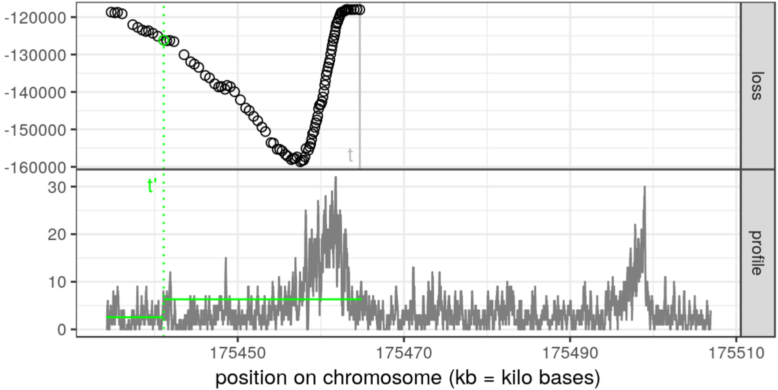
$$\mathcal{L}_{1,t} = \underbrace{c_{(0,t]}}_{\text{optimal loss of 1st segment } (0,t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 1$ segments up to data point t



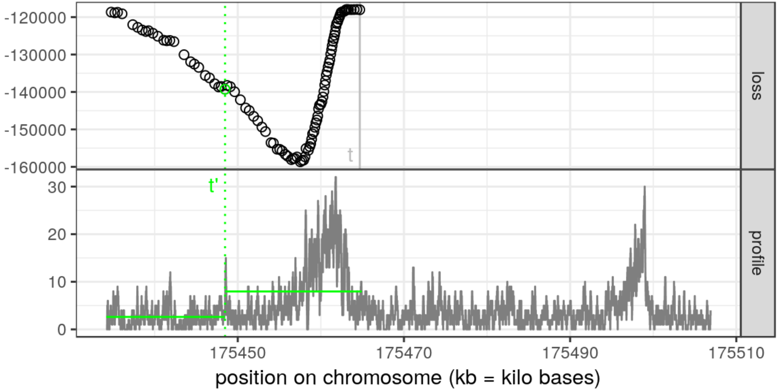
$$\mathcal{L}_{1,t} = \underbrace{c_{(0,t]}}_{\text{optimal loss of 1st segment } (0,t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to data point $t < d$



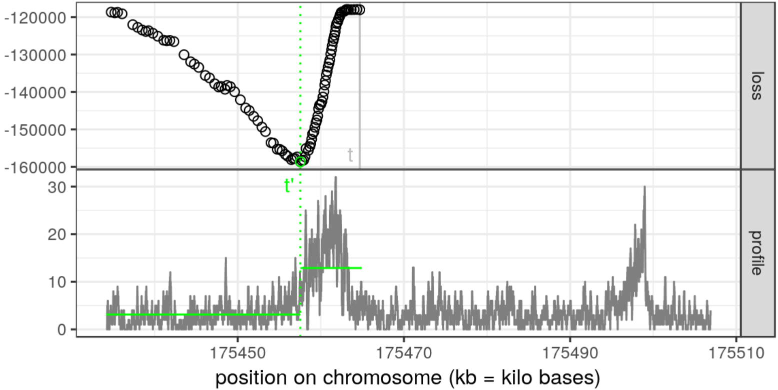
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to data point $t < d$



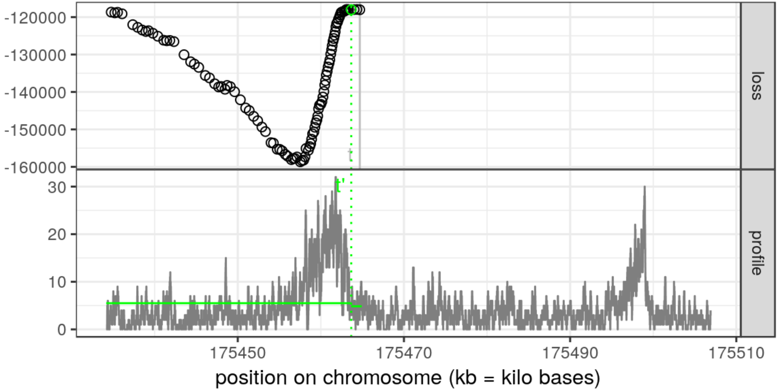
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to data point $t < d$



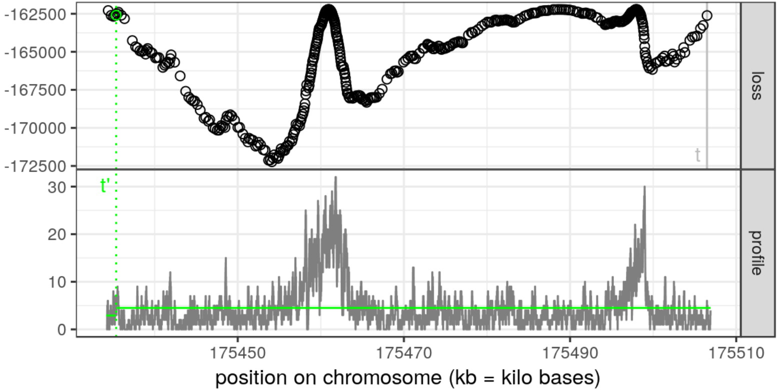
$$\mathcal{L}_{2,t} = \min_{t' < t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',\, t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to data point $t < d$



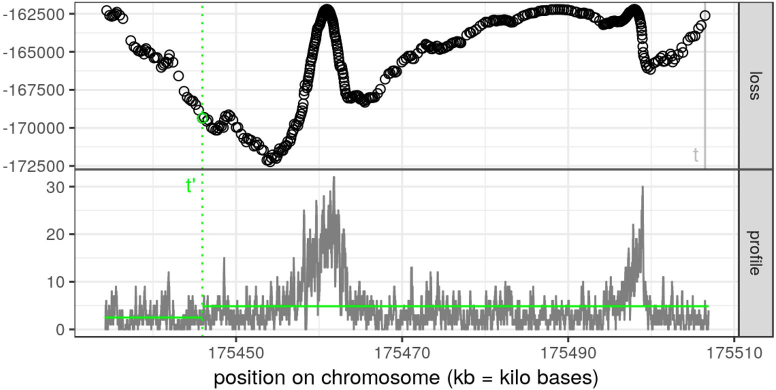
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to last data point $t = d$



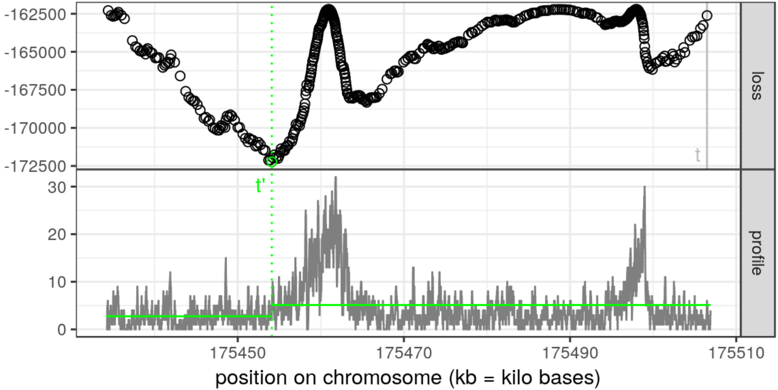
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to last data point $t = d$



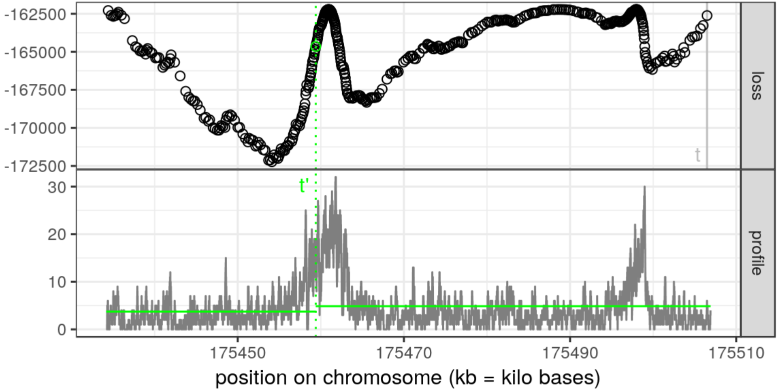
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t', t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to last data point $t = d$



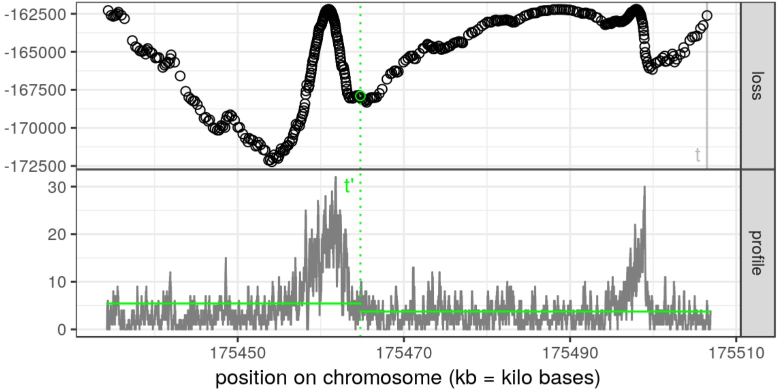
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to last data point $t = d$



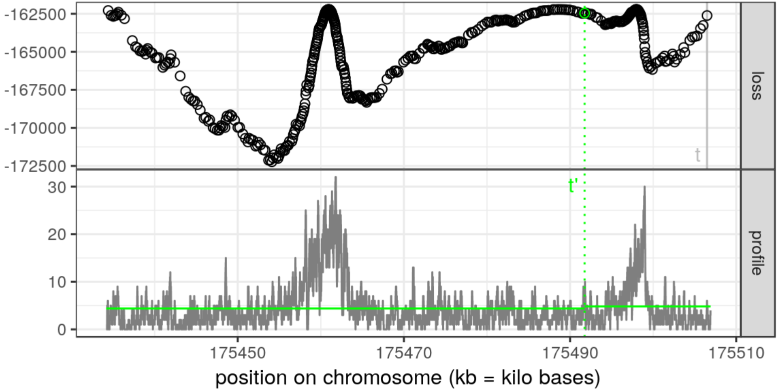
$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to last data point $t = d$



$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 2$ segments up to last data point $t = d$



$$\mathcal{L}_{2,t} = \min_{t'<t} \underbrace{\mathcal{L}_{1,t'}}_{\text{optimal loss in 1 segment up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 2nd segment } (t',t]}$$

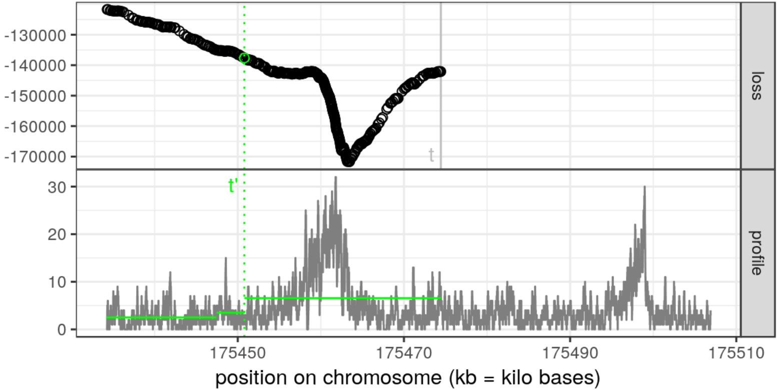
Dynamic programming is faster than grid search for $s > 2$ segments

Computation time in number of data points n:

segments s	grid search	dynamic programming
1	$O(n)$	$O(n)$
2	$O(n^2)$	$O(n^2)$
3	$O(n^3)$	$O(n^2)$
4	$O(n^4)$	$O(n^2)$
$\vdots$	$\vdots$	$\vdots$

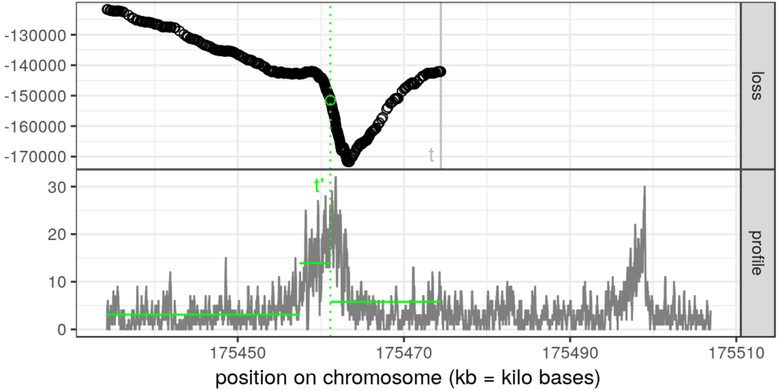
For example $n = 5735$ data points to segment.
$n^2 = 32890225$
$n^3 = 188625440375$
$\vdots$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 3$ segments up to data point t



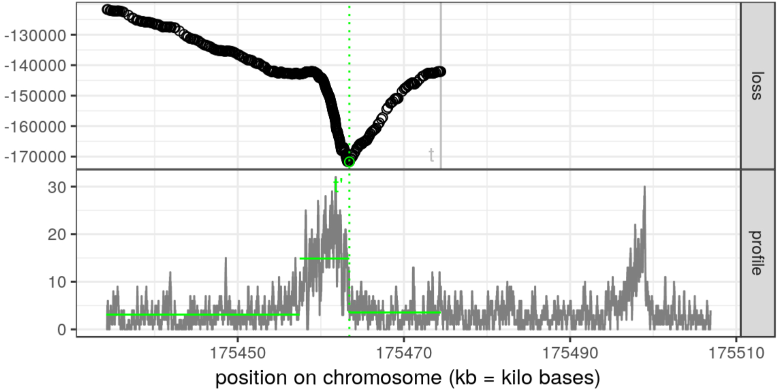
$$\mathcal{L}_{3,t} = \min_{t'<t} \underbrace{\mathcal{L}_{2,t'}}_{\text{optimal loss in 2 segments up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 3rd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 3$ segments up to data point t



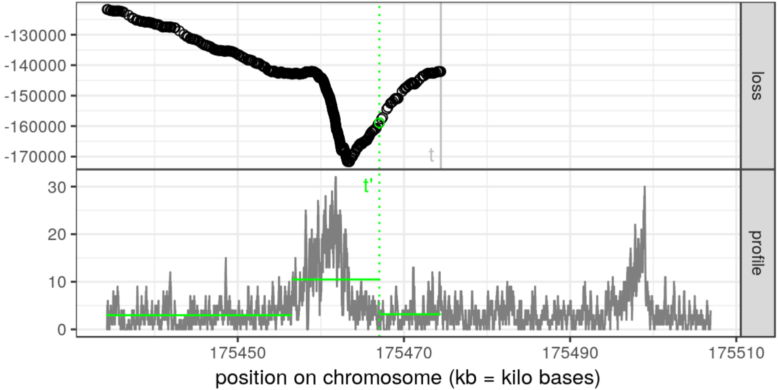
$$\mathcal{L}_{3,t} = \min_{t'<t} \underbrace{\mathcal{L}_{2,t'}}_{\text{optimal loss in 2 segments up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 3rd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 3$ segments up to data point t



$$\mathcal{L}_{3,t} = \min_{t'<t} \underbrace{\mathcal{L}_{2,t'}}_{\text{optimal loss in 2 segments up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 3rd segment } (t',t]}$$

Computation of optimal loss $\mathcal{L}_{s,t}$ for $s = 3$ segments up to data point t



$$\mathcal{L}_{3,t} = \min_{t'<t} \underbrace{\mathcal{L}_{2,t'}}_{\text{optimal loss in 2 segments up to } t'} + \underbrace{c_{(t',t]}}_{\text{optimal loss of 3rd segment } (t',t]}$$

Classical dynamic programming for optimal changepoint detection

Functional pruning algorithms

Empirical time complexity

Theoretical time complexity

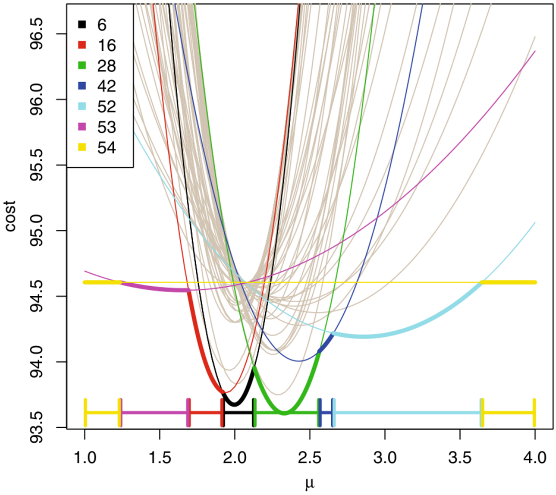
Classical dynamic programming is too slow for big data

- Motivated by big data sequences $n > 1000$ in genomics and other fields, for which $O(n^2)$ is too slow.
- Recent work into functional pruning algorithms which compute the same solution in $O(n \log n)$ (empirically).
- Independent discovery by Rigaill arXiv:1004.0887, JFdS 2015; Johnson PhD 2011, JCGS 2013. Main idea: first minimize on the last changepoint t_{S-1}, then on the last segment mean u_S:

$$
\begin{aligned}
\mathcal{L}_{S,n} &= \min_{t_{S-1}} \mathcal{L}_{S-1,t_{S-1}} + \underbrace{\min_{u_S} \sum_{i=t_{S-1}+1}^{t_S=n} \ell(u_S, z_i)}_{c_{(t_{S-1}, t_S=n]}} \text{ — classical} \\
&= \min_{u_S} \underbrace{\min_{t_{S-1}} \mathcal{L}_{S-1,t_{S-1}} + \sum_{i=t_{S-1}+1}^{t_S=n} \ell(u_S, z_i)}_{C_{S,n}(u_S)} \text{ — functional}
\end{aligned}
$$

Functional cost representation results in pruning

Maidstone, *et al.* Statistics and Computing 2016.



- Only need to store the minimum cost (colored lines/intervals).
- No need to consider the cost functions/changepoints which are not optimal for any mean values (grey lines).

Dynamic programming recursion with functional pruning

- τ is first data point on last segment.
- μ is last segment mean.

$$
\begin{aligned}
C_{S,n}(\mu) &= \min_{\tau \in \{S,\dots,n\}} \mathcal{L}_{S-1,\tau-1} + \sum_{i=\tau}^{n} \ell(\mu, z_i) \\
&= \min\{\mathcal{L}_{S-1,S-1} + \sum_{i=S}^{n} \ell(\mu, z_i), \dots, \\
&\qquad \mathcal{L}_{S-1,n-1} + \ell(\mu, z_n)\} \\
&= \ell(\mu, z_n) + \min\{\mathcal{L}_{S-1,S-1} + \sum_{i=S}^{n-1} \ell(\mu, z_i), \dots, \\
&\qquad\qquad \mathcal{L}_{S-1,n-2} + \ell(\mu, z_{n-1})\} \\
&\qquad\qquad \mathcal{L}_{S-1,n-1}\} \\
&= \ell(\mu, z_n) + \min\{C_{S,n-1}(\mu), \mathcal{L}_{S-1,n-1}\}
\end{aligned}
$$

Example data set with $n = 4$

Rigaill, arXiv:1004.0887.

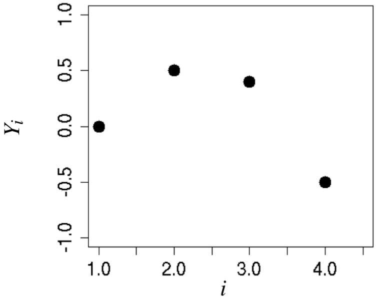
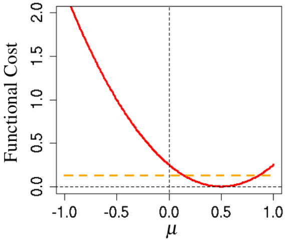


FIGURE 1. *Four-point signal. y_i as a function of i.* $y_1 = 0$, $y_2 = 0.5$, $y_3 = 0.4$, $y_4 = -0.5$

Functional cost computation at $t = 3$

Rigaill, arXiv:1004.0887.

- Data: 0, 0.5, 0.4, -0.5.
- $\mathcal{L}_{1,1} = \min_\mu (\mu - 0)^2 = 0$.
- $\mathcal{L}_{1,2} = \min_\mu (\mu - 0)^2 + (\mu - 0.5)^2 = 0.125$.
- Computing $C_{2,3}(\mu) = \ell(\mu, z_3) + \min\{C_{2,2}(\mu), \mathcal{L}_{1,2}\}$:
- Change before $\tau = 2$: $C_{2,2}(\mu) = \mathcal{L}_{1,1} + (\mu - 0.5)^2$.
- Change before $\tau = 3$: $\mathcal{L}_{1,2}$.



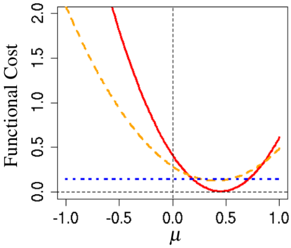
τ	Functional cost	$\mathscr{S}^1_{1:3,\tau}$
2	$0.25 - \mu + \mu^2$	$[\,0.146\ ,\ 0.854\,]$
3	0.125	$[\,-\infty, 0.146\,] \cup [\,0.854, +\infty\,]$

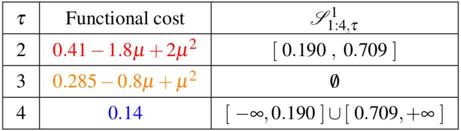
FIGURE 2. *Functional cost of $Y_{1:3}$ for $K = 1$ using the quadratic loss. (Left) Functional cost as a function of μ of segmentations having a change-point at $\tau = 2$ (solid red) and $\tau = 3$ (orange dashed). (Right) Analytical expression of the functional costs for $\tau = 2$ and $\tau = 3$ and the set of μ, for which they are optimal: $\mathscr{S}^1_{1:3,\tau}$.*

Functional cost computation at $t = 4$

Rigaill, arXiv:1004.0887.

- Data: 0, 0.5, 0.4, -0.5.
- Computing $C_{2,4}(\mu) = \ell(\mu, z_4) + \min\{C_{2,3}(\mu), \mathcal{L}_{1,3}\}$:
- Change before $\tau = 2$: $\mathcal{L}_{1,1} + (\mu - 0.5)^2 + (\mu - 0.4)^2$.
- Change before $\tau = 3$: $\mathcal{L}_{1,2} + (\mu - 0.4)^2$.
- Change before $\tau = 4$:
  $\mathcal{L}_{1,3} = \min_\mu (\mu - 0)^2 + (\mu - 0.5)^2 + (\mu - 0.4)^2$.



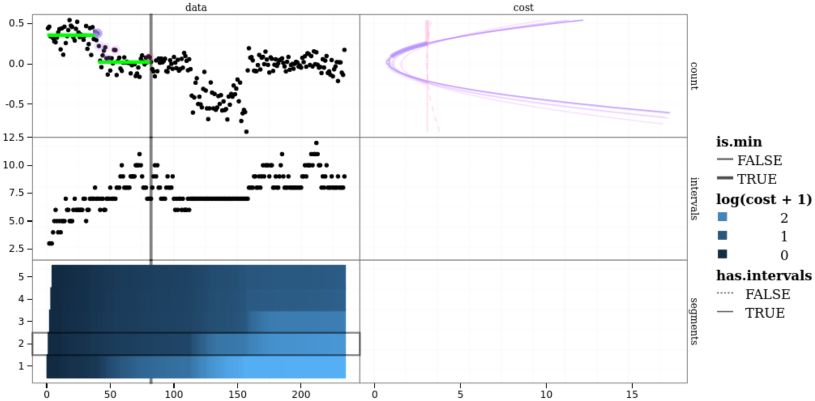


τ	Functional cost	$\mathscr{S}^1_{1:4,\tau}$
2	$0.41 - 1.8\mu + 2\mu^2$	$[\,0.190\,,\ 0.709\,]$
3	$0.285 - 0.8\mu + \mu^2$	$\emptyset$
4	0.14	$[\,-\infty, 0.190\,] \cup [\,0.709, +\infty\,]$

FIGURE 3. *Functional cost of $Y_{1:4}$ for $K = 1$ using the quadratic loss. (Left) Functional cost of a segmentations having a change-point at $\tau = 2$ (solid red) $\tau = 3$ (orange dashed) and $\tau = 4$ (blue dotted). (Right) Analytical expression of the functional costs for $\tau = 2$, 3 and 4 and the set of μ, for which they are optimal: $\mathscr{S}^1_{1:4,\tau}$.*

Functional pruning larger example

- Computing each $C_{s,t}(\mu)$ is an $O(I)$ operation where I is the number of intervals (candidate changepoints).
- Need to compute $O(Sn)$ functions; total complexity is $O(SnI)$.
- Empirically $I = O(\log n)$ due to pruning so overall $O(Sn \log n)$.



https://rcdata.nau.edu/genomic-ml/figure-unconstrained-PDPA-normal-big/

Classical dynamic programming for optimal changepoint detection

Functional pruning algorithms

Empirical time complexity

Theoretical time complexity

Number of intervals in real and simulated data

Rigaill, arXiv:1004.0887.

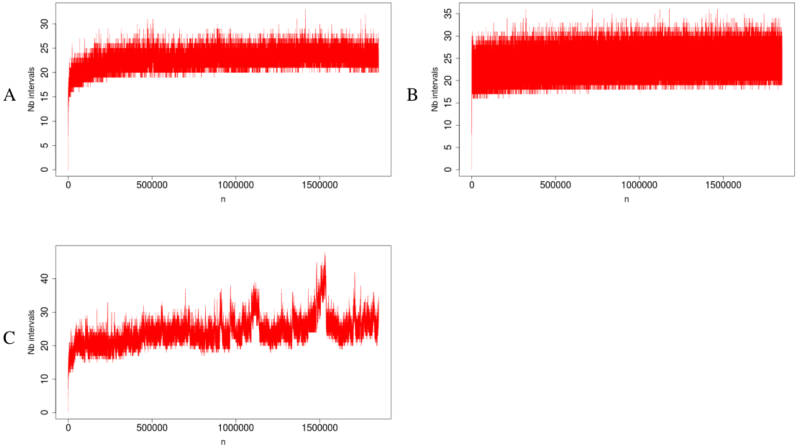
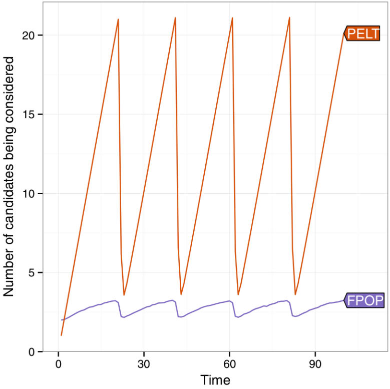


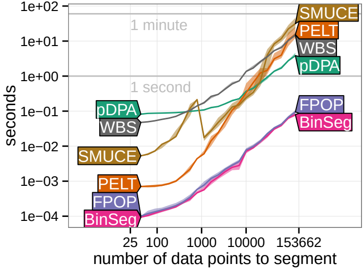
FIGURE 5. *Maximum number of intervals stored by the pDPA at each point of the sequence for $K = 1$. A: For 100 sequences of $1.8\ 10^6$ points simulated with a constant signal plus an additional normal noise of variance 1. B: For 100 sequences of $1.8\ 10^6$ points simulated with a sine wave signal plus an additional normal noise of variance 1. C: For the 18 profiles of length $1.8.\ 10^6$ of the GSE17359 dataset.*

Another fast functional pruning algorithm

Maidstone, *et al.* Statistics and Computing 2016.

$$\underset{\mathbf{m}\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{n} \ell(m_i, z_i) + \lambda \sum_{i=1}^{n-1} I[m_i \neq m_{i+1}]$$





Algorithm with constraints is also fast

H, *et al.* Journal of Machine Learning Research 21(87):1–40, 2020.

$$\underset{\mathbf{m}\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{n} \ell(m_i, z_i)$$

$$\text{subject to} \quad \sum_{i=1}^{n-1} I[m_i \neq m_{i+1}] = S - 1,$$

...up-down constraints on m.

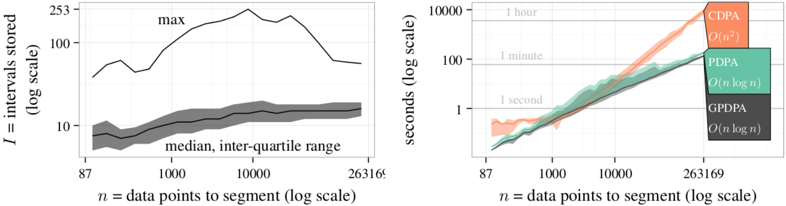


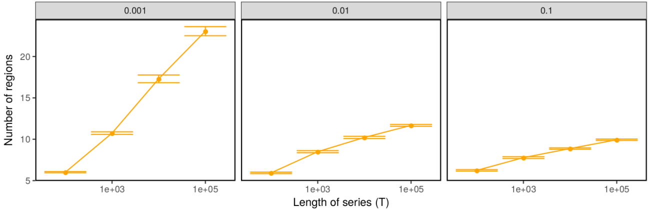
Figure 3: Empirical speed analysis on 2752 count data vectors from the histone mark ChIP-seq benchmark. For each vector we ran the GPDPA with the up-down constraint and a max of $K = 19$ segments. The expected time complexity is $O(KnI)$ where I is the average number of intervals (function pieces; candidate changepoints) stored in the $C_{k,t}$ cost functions. **Left**: number of intervals stored is $I = O(\log n)$ (median, inter-quartile range, and maximum over all data points t and segments k). **Right**: time complexity of the GPDPA is $O(n \log n)$ (median line and min/max band).

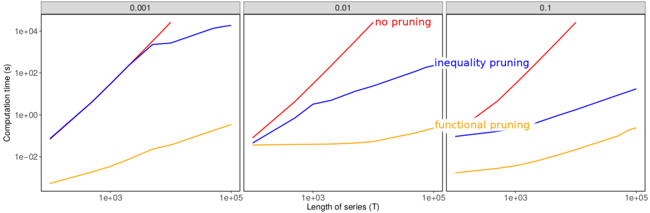
Another fast constrained algorithm for neuroscience

Jewell, *et al.* *Biostatistics* 2019.

$$\underset{c_1,\ldots,c_T,z_2,\ldots,z_T}{\text{minimize}} \quad \frac{1}{2}\sum_{t=1}^{T}(y_t - c_t)^2 + \lambda \sum_{t=2}^{T} 1_{(z_t \neq 0)}$$

$$\text{subject to} \quad z_t = c_t - \gamma c_{t-1} \geq 0.$$



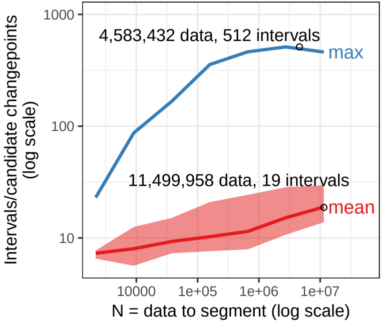


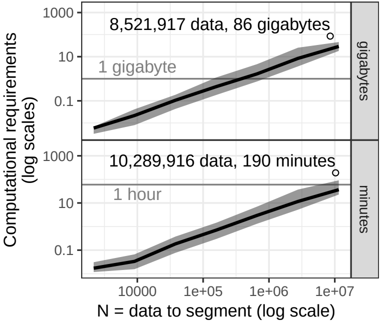
Another fast constrained algorithm for genomics

H, *et al.* arXiv:1810.00117, accepted in *J. Statistical Software*.

$$\underset{\mathbf{m}\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{n} \ell(m_i, z_i) + \lambda \sum_{i=1}^{n-1} I[m_i \neq m_{i+1}]$$

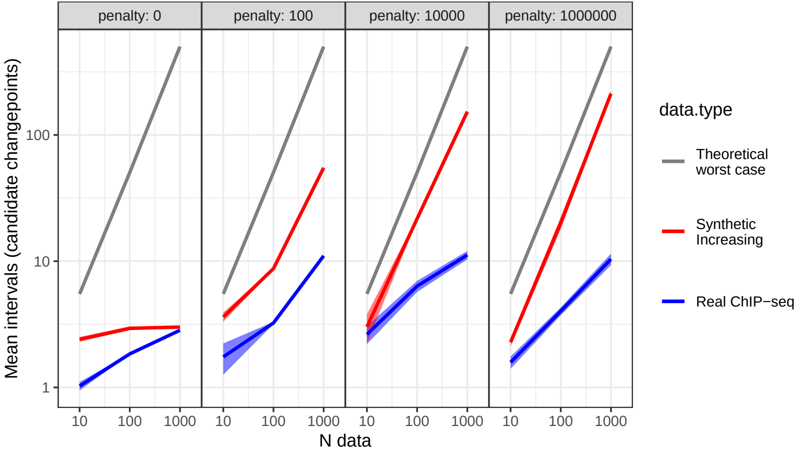
subject to ...up-down constraints on m.





$I = O(\log n)$ intervals. Overall $O(n \log n)$ complexity.

Worst case quadratic complexity only observed with synthetic data and large penalties



Mean ± standard deviation over 10 data sets.

Classical dynamic programming for optimal changepoint detection

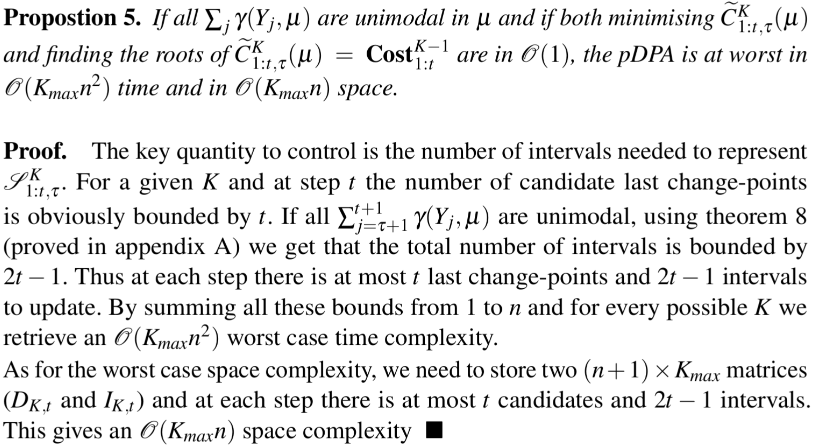
Functional pruning algorithms

Empirical time complexity

Theoretical time complexity

Worst case complexity is quadratic

Rigaill, arXiv:1004.0887.

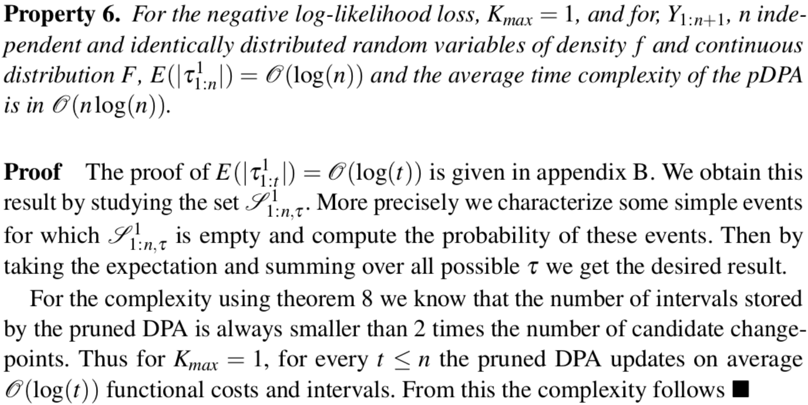


Propostion 5. *If all $\sum_j \gamma(Y_j, \mu)$ are unimodal in μ and if both minimising $\widetilde{C}^K_{1:t,\tau}(\mu)$ and finding the roots of $\widetilde{C}^K_{1:t,\tau}(\mu) = \mathbf{Cost}^{K-1}_{1:t}$ are in $\mathcal{O}(1)$, the pDPA is at worst in $\mathcal{O}(K_{max}n^2)$ time and in $\mathcal{O}(K_{max}n)$ space.*

Proof. The key quantity to control is the number of intervals needed to represent $\mathcal{S}^K_{1:t,\tau}$. For a given K and at step t the number of candidate last change-points is obviously bounded by t. If all $\sum_{j=\tau+1}^{t+1} \gamma(Y_j, \mu)$ are unimodal, using theorem 8 (proved in appendix A) we get that the total number of intervals is bounded by $2t-1$. Thus at each step there is at most t last change-points and $2t-1$ intervals to update. By summing all these bounds from 1 to n and for every possible K we retrieve an $\mathcal{O}(K_{max}n^2)$ worst case time complexity.
As for the worst case space complexity, we need to store two $(n+1) \times K_{max}$ matrices ($D_{K,t}$ and $I_{K,t}$) and at each step there is at most t candidates and $2t-1$ intervals. This gives an $\mathcal{O}(K_{max}n)$ space complexity ■

Average case complexity proof for uniform loss

Rigaill, arXiv:1004.0887.



Property 6. *For the negative log-likelihood loss, $K_{max} = 1$, and for, $Y_{1:n+1}$, n independent and identically distributed random variables of density f and continuous distribution F, $E(|\tau^1_{1:n}|) = \mathscr{O}(\log(n))$ and the average time complexity of the pDPA is in $\mathscr{O}(n \log(n))$.*

Proof The proof of $E(|\tau^1_{1:t}|) = \mathscr{O}(\log(t))$ is given in appendix B. We obtain this result by studying the set $\mathscr{S}^1_{1:n,\tau}$. More precisely we characterize some simple events for which $\mathscr{S}^1_{1:n,\tau}$ is empty and compute the probability of these events. Then by taking the expectation and summing over all possible τ we get the desired result.

For the complexity using theorem 8 we know that the number of intervals stored by the pruned DPA is always smaller than 2 times the number of candidate change-points. Thus for $K_{max} = 1$, for every $t \leq n$ the pruned DPA updates on average $\mathscr{O}(\log(t))$ functional costs and intervals. From this the complexity follows ■

Conclusions

- Optimal detection of $S-1$ changepoints in n data is naively a $O(n^S)$ computation.
- Functional pruning method yields algorithms with worst case time complexity of $O(n^2)$ (same as classical dynamic programming).
- Empirically the functional pruning algorithms are much faster, $O(n \log n)$.
- Only one proof of average time complexity for 1 changepoint and the uniform loss function (never used in practice).
- Would be interesting to prove $O(n \log n)$ average time complexity in other more realistic situations. (square/Poisson loss, λ) How?
- Let's collaborate! toby.hocking@nau.edu