DNA copy number segmentation by dynamic programming

Pierre Neuvial

December 4, 2015

The goal of the session is to implement the algorithm for segmenting univariate signals by dynamic programming proposed by Picard et al. (2005).

1 Statistical model

1.1 Notation

- $j = 1 \dots n$: genomic loci
- $(\gamma_j)_{j=1\dots n}$: true DNA copy numbers
- $(y_j)_{j=1\dots n}$: observations

1.2 Assumptions

- breakpoints: $(t_k)_{0 \leq k \leq K}$, with $t_0 = 1$ et $t_K = n + 1$
- region-level copy numbers $(\Gamma_k)_{1 \leq k \leq K}$ such that $\gamma_j = \Gamma_k, \forall j \in [t_{k-1}, t_k), \forall k \in \{1, \dots, K\}$

We observe $y_j = \gamma_{k(j)} + \varepsilon_j$, with $k(j) = \max\{k, t_k \leq j\}$, where the noise $(\varepsilon_j)_{j=1\dots n}$ is iid and assumed to follow $\mathcal{N}(0, \sigma^2)$, where σ is unknown.

1.3 Homoscedastic vs heteroscedastic models

Here, we assume that the σ does not depend on the region (homoscedastic model). However, the variance of copy number signals has been reported to be increasing with their mean. Therefore, an heteroscedastic model where $\sigma = \sigma_k$ may be more realistic. A common practice in applications is to to transform the raw copy number signals using $\sqrt{\cdot}$, $\log(\cdot)$, or $(\cdot)^{1/3}$, in order to stabilize the variance of the signal. Then the above homoscedastic model makes sense. We refer the interested reader to Picard et al. (2005) for a discussion on the estimation of the homoscedastic vs the heteroscedasctic model.

1.4 Likelihood of the model

The log-likelihood of the model is given by:

$$\ell(K, 1:n) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{j=1}^{J} (y_j - \gamma_j)^2$$

or, equivalently,

$$\ell(K, 1:n) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=1}^{K}\sum_{j=t_{k-1}}^{t_k}\left(y_j - \Gamma_{k(j)}\right)^2 .$$

The maximum Likelihood (ML) estimator of $\Gamma[k(j)]$ is

$$\widehat{\Gamma[k(j)]}^{ML} = \frac{1}{t_k - t_{k-1}}\sum_{j=t_{k-1}}^{t_k} y_j$$

Proof: $\sum_1^n (y_j - \gamma_j)^2 = \sum_{k=1}^K \sum_{j=t_{k-1}}^{t_k-1} (y_j - \Gamma_k)^2$ by the piece-wise constant assumption.

The number of possible breakpoint positions for a given K is $C_{n-1}^{K-1} = O(n^K)$ which is too large for genomic applications where $n \sim 10^5$ and $K \sim 100$ (or more).

1.4.1 Formulation as a non-convex optimization problem

The problem of finding the change point locations can be treated independently of the estimation of the noise level σ. This is only true in the homoscedastic model. With this remark, we can rewrite the change point location problem as an optimization problem for the ℓ^2 loss:

$$\min_{(\gamma_j)_{1\leq j\leq n}} \sum_{j=1}^{J}(y_j - \gamma_j)^2 \quad \text{s.c.} \quad \sum_{j=1}^{n-1}\mathbf{1}_{\gamma_{j+1}\neq\gamma_j} \leq K$$

With this formulation, the piecewise constant assumption is simply written in terms as a constraint on the ℓ_0 norm of the first order differences of γ. Because the ℓ_0 norm is non-convex, there is no computationally-efficient way to solve this optimization problem, which is coherent with the above remark on the computational complexity of an exhaustive search for the maximum likelihood of the model.

2 Dynamic programming

Let $R(k, j_1 : j_2)$ be the RSE of the best model with k segments between j_1 and j_2:

$R(k, j_1 : j_2) = \sum_{j_1}^{j_2}(y_j - \hat{\gamma}^{\mathrm{ML}})^2$.

Note that $\hat{\gamma}^{\mathrm{ML}}$ depends on the breakpoint positions. The trick is to calculate V by induction on K:

2.1 Idea of the algorithm

- Compute $R(1, j_1 : j_2)$ for all (j_1, j_2) such that $1 \leq j_1 < j_2 \leq n$
- Compute $R(K+1, \cdot)$ from $R(K, \cdot)$ by noting that for all (j_1, j_2) such that $1 \leq j_1 < j_2 \leq n$,

$$R(K+1, j_1 : j_2) = \max_{h\in[j_1,j_2]} R(K, j_1 : h) + R(1, (h+1) : j_2)$$

Using this induction formula, we can compute $R(K, j_1 : j_2)$ for all K in $O(n^3)$. A simpler induction formula is the following:

- Compute $R(K+1, 1 : j)_{1\leq j\leq n}$ from $(R(K, 1 : j))_{1\leq j\leq n}$ by noting that for all $j \in \{1 \dots n\}$,

$$R(K+1, 1 : j) = \max_{h\in[1,j]} R(K, 1 : h) + R(1, (h+1) : J)$$

This formula only requires the calculation (and storage) of $O(n^2)$ terms at each iteration. If the initialization step is implemented efficiently, the total time complexity of this algorithm is $O(Kn^2)$, for a space complexity of $O(n^2)$ (due to the storage of the $R(1,\cdot)$ matrix calculated at initialization).

3 Implementation

= your work for this session!

Goal: write an R function `dpseg` that takes as input the signal y to be segmented and the maximum number K of breakpoints to be retrieved, that returns for each k in $\{1,\dots K\}$ the best segmentation of the input signal in k breakpoints.

Intermediate steps:

1. Calculate the matrix `J` of size $n \times n$ defined by $J[i,j] = R(1, i:j)$ for $j \geq i$ and $J[i,j] = 0$ for $j < i$.
2. The complexity of a naive implementation of this step is already cubic ($O(n^3)$). How can we improve on this?
3. Calculate by induction on k the matrix `V` of size $K+1 \times n$ defined by $V[k,j] = R(K, 1:j)$ for $1 \leq k \leq K$ and $1 \leq j \leq n$. Also make sure to store for each (k,j) the index $h(k,j)$ of the last breakpoint (i.e. the k-th breakpoint) of the best segmentation of $[1,j]$ in k segments in a matrix called `bkp` of size $K+1 \times n$
4. ("backtracking") Deduce from `bkp` the best segmentation of $[1,n]$ in k segments for each k.

3.1 To go further

- Can this algorithm be extended to a multivariate signal segmentation?
- How can we speed up the code?
- How can this algorithm be adapted to handle missing values?
- How can this algorithm be extended to *prune* a set of candidate change points?

References

Picard, Franck, Stephane Robin, Marc Lavielle, Christian Vaisse, and Jean-Jacques Daudin. 2005. "A Statistical Approach for Array CGH Data Analysis." *BMC Bioinformatics* 6 (January): 27. doi:10.1186/1471-2105-6-27.