1 Introduction

Segmentation is the partition of an ordered set of elements to continuous groups.

One example for segmentation in Molecular biology is partitioning a DNA sequence according to its G-C content. Some segments will be G-C rich, others G-C poor.

Another example is the analysis of several microarray samples taken under the same condition at different times. Each segment would mark a different stage in the sampled process. For instance, if we want to learn about the cell cycle, each segment will mark a different stage. So a typical result will be something like “Samples 1-7 are taken from stage G1, samples 8-10 are from stage S...”.

Today, we will discuss automatic, computational methods for performing such segmentation.

2 Case Study: Copy Number Variation

When thinking of genes, we’re accustomed to deal with the simple case where there is a single locus with two possible alleles, which cause the genetic variation.

Multiple gene copies add a layer of complexity to that basic model. Moreover, variation in copy number affects overall expression variation. Recent researches suggest that the phenomenon of copy number variation within the population is more common than what was once assumed.

This is emphasized in the case of cancer, where damage to the cellular control mechanisms might cause entire chromosome segments to be duplicated following a SS or DS break.

Microarray analysis could be used in order to identify variation in copies number. We can take two samples of genomic DNA - a ‘normal’ control sample and a test sample. Using the techniques we leaned in the Microarray lessons, we can get the ratio between the amount of genomic DNA from the control sample and the microarray sample. If we position the microarray results according to their position in the chromosome, we will be able to identify regions in the chromosome where entire segments were duplicated or deleted.

As it always happens in real life, the idea is not ideal. For example, one can expect some copy number variation even within a single cancerous sample. Even when assuming that a certain variant of cancerous cell (with specific duplications / deletions) will be dominant, this still leaves us with noisy data. Consequently determining where one segment ends and another one starts isn’t trivial.
Figure 1: Microarray data segmentation. Note that in the first segment, ratio to normal is 1:1. The second segment features a reduction in copy numbers (deletion) and the third segment features an increase (duplication).

3 Segmentation Methods

Let’s consider several methods to partition unfriendly data to segments:

3.1 Sliding window

We define a window size on the data’s ‘X axis’. For example, if our samples are taken over time (measured in seconds), then for time T our window could contain all the samples in \([T - 10 \ldots T + 10]\). For each sample, we will take all the samples within its window, calculate the average of their values and change that sample’s value to the average.

This will smoothen the results, and thus once we pass between samples which belong to different segments, we will expect their values to change significantly.

This approach is relatively immune to noise, but the changes between the segments are less sharp and clear.

Figure 2: Sliding window approach - the average value function is interpolated using the windows. Sharp changes in the function might suggest segment switch.

3.2 Hidden Markov Model

We can model the various segment types as latent variables. We can define the probability of changing segments by a parameter, and learn that parameter. If the segments are long, the transition probability defined the parameter will be low - and vice versa.
While we all love and cherish HMMs, there are a couple drawbacks with applying it to the segmentation problem:

- The number of 'states' (segment types) must be finite.
- These states need to be pre-determined.
- We need to learn the parameter using Expectation-Maximization or another parameter-discovering algorithm. This is a heavy, complex task.

### 3.3 Dynamic Programming

In the end, it all boils down to Dynamic Programming. Dynamic Programming will not suffer from the HMM’s drawbacks mentioned earlier.

The basic idea is that the data contains several independent 'segment switch’ point. If we divide a segment of data to two sub-segments, the optimal segmentation will be based on the optimal segmentation of for each subsegment.

**Definition 3.1** Segmentation is defined as a set of sample indices, which mark the samples which are on the border between two segments:

\[ \sigma = \{ i_1 \ldots i_k \} \]

**Definition 3.2** Segment Score is a function that gives a score to a single segment, \([X_{i_j}, X_{i_{j+1}}]\):

\[ \text{sscore}(X_{i_j}, X_{i_{j+1}}) \]

**Definition 3.3** Segmentation Score is a function that ranks an entire segmentation, by summing the segment scores and adding a penalty for the amount of segments, \( f(\cdot) \).

\[ \text{score}(\sigma) = \sum_{j=1}^{k} \text{sscore}(X_{i_j}, X_{i_{j+1}}) + f(\sigma) \]

We assume that the scores are additive.

In order to include all of the samples, the first and last segmentation indices will match the first and last samples indices. For \( n \) samples and \( k - 1 \) segments, it means that \( i_1 = 1 \) and \( i_k = n \)

In our Dynamic Programming algorithm, we will iteratively calculate the optimal segmentation for a subsequence of samples, \( X_1 \ldots X_m \) for \( m = 1 \ldots n \). We will denote this calculation \( V[m] \).

\[ V[m] = \max_{\sigma} \text{score}(\sigma, X_1, \ldots, X_n) = * \]

Since the segment scores are additive, every possible score would be achieved by the optimal solution for all the segments excluding the last segment, plus the last segment’s score. All we need to do is check every possibility for the last segment’s start index, and choose the one that provides maximal score:

\[ V[m] = * = \max_{l} V[l] + \text{sscore}(X_l, X_m) \]

Our next task is to define the segment score function. We will consider two approaches:

**L2 distance** - the sum of distances for each sample in the segment from the expectation.

\[ L2 = \sum_{i \in l} (X_i - \mu)^2 \]
where \( I \) is the set of samples contained in the segment.

The problem with this method is that partitioning the data to 1-sample-long segments yields the best score. A possible solution would be to predetermine the optimal amount of segments and define a tradeoff between that number to the score.

**Likelihood** - we assume that the data in a segment comes from the same distribution model, and estimate its parameters with Maximum Likelihood.

Some models will be more complex to estimate - for example, a linear function which is dependant on the x-axis.

We obtain a Maximum Likelihood-equivalent estimation:

\[
\text{score}([X_1, \ldots, X_k]) = \max_{\theta} \log P(X_1, \ldots, X_k|\theta) = *
\]

And, assuming that the samples are independant, we conclude that:

\[
* = \max_{\theta} \sum_i \log P(X_i|\theta)
\]

For example, if the distribution model is Gaussian with a fixed variance \( \sigma^2 \), we can calculate the likelihood as follows:

\[
\text{score}([X_1, \ldots, X_k]) = \max_{\mu} \sum_i \log P(X_i|\mu, \sigma^2)
= -\frac{k}{2} \cdot \log(2 \cdot \pi \sigma^2) - \frac{1}{2} \sigma^2 \sum_i (X_i - \mu)^2
\]

To enhance the segmentation score function, we can add a prior on the segment lengths, \( P_I(k) \).

\[
\text{score}([X_1, \ldots, X_k]) = \max_{\theta} \sum_i \log P(X_i|\theta) + \log P_I(\text{SegmentLength}_i)
\]

### 4 Bayesian Approach to Segmentation

In the Bayesian approach, we have a different confidence level for each probable segmentation, and we can create a 'distribution of segmentations'.

Let us denote a segmentation as:

\[
\sigma = \{i_1 \ldots i_{k+1}\}
\]

and the segmentation distribution parameters as:

\[
\Theta_\sigma = \{\theta_1 \ldots \theta_k\}
\]

The probability of some event \( A \) in the newfound distribution is:

\[
P(A|X_1 \ldots X_n) = \frac{1}{P(X_1 \ldots X_n)} \cdot P(A, X_1, \ldots, X_n)
= \frac{1}{P(X_1 \ldots X_n)} \cdot \sum_\sigma \int_{\theta_\sigma} P(A, X_1, \ldots, X_n) \cdot P(\theta_\sigma|\sigma) \cdot P(\sigma)
\]

We can ask a few inference-related questions here:

- Do two samples, \( X_{i_a} \) and \( X_{i_b} \), belong to the same segment?
• What are the parameters which define the distribution of the segment from which a specific sample $X_{k_{a}}$ was sampled? Or in other words, if $X_{k_{a}} \in \sigma_{k}$, what is $\theta_{\sigma_{k}}$?

The practical difference between the Maximum Likelihood approach and the Bayesian approach is, that in MLE we find a maximum value whereas here we sum over all the weighted probabilities. Our Dynamic Programming formula changes accordingly.

$$B[m] = P(X_1 \ldots X_n) = \sum_{1<l<m} B[l] \cdot P([X_{l+1} \ldots x_m]) \cdot P(\sigma_m|\sigma_l)$$

Note that, given the distribution parameterization,

$$P([X_{l+1} \ldots x_m]) = \int_{\theta} P(X_{l+1} \ldots x_m|\theta) \cdot P(\theta)$$

How are we going to answer the first question, which is equivalent to asking whether there is a 'segment switch' between two points? We simply sum the probabilities of all of the segmentations which include a segment switch between the two points.

That’s it for CBio - good luck in the exam!