National Research University
Higher School of Economics
Faculty of Mathematics

Project Proposal

Clustering of Multidimensional Random Variables to Improve HMM Sequence Alignment Accuracy Кластеризация многомерной случайно величины для поднятия точности выравнивания строк.

Student: Denis Grachev, БМТ181

Scientific advisor:
PhD candidate,
Timofey Prodanov

Moscow 2022

Contents

1 Abstract

Whole-genome sequencing is a complex and important task of bioinformatics. Different technologies can generate data of different nature. Most popular technologies, such as Illumina, have low error rate, but give limited information about genome. Newer technologies, for example Pacific Biosciences and Oxford Nanopore, can give more details about genome, but have higher error rate. Combination of these methods can help improve accuracy of genome sequencing. During this work a new method of string clustering was developed, to identify different data profiles and new functionality to existing tools were added, to work with multiple datasets.

2 Introduction

Bioinformatics is an interdisciplinary science that aims to develop methods and software tools for understanding biological data. The most useful biological analysis is related to DNA (genome) exploration. One of the ways to represent genome is as a set of strings over the alphabet $\{A, C, G, T\}$, each corresponding to a chromosome. Most species are diploid, so every chromosome is paired, one copy from each parent. Modern technologies of reading human genome (genome sequencing), such as Illumina [1], do not allow to read it as one continuous string, but a number of random substrings that are called reads. In most species unrelated individuals have highly similar genome sequences (99.9% among humans [2]) As within one biological species genomes coincide almost completely, it is convenient to determine one reference genome for one species and identify for every individual it's deviations from the reference. Most common deviations are single nucleotide substitutions, which are called SNVs. A number of problems appears for example:

- **Genome assembly** is process of deciphering genome using reads obtained from it.
- **Sequence alignment** is process of arranging sequences to spot similarities between them.
- **Variant calling** is process of identifying SNVs of an individual based on reads aligned on the reference genome.

Solving this problems are complicated by errors in genome sequencing. The most commonly used technologies, Illumina [1], allow to sequence reads of length 200-500 bp. Sequencing human genome using such short length reads has many limitations. First, due to diploidy of humans, it is important to obtain long-range relation between SNVs on each homologous chromosome. This might be difficult with short-reads provided by Illumina [1]. Second, 3.6% [3] of human genome consists of long highly repetitive duplicated regions. Calling SNVs in such regions using short-read sequencing technologies is increasingly difficult, which lowers accuracy of SNVs. Third-generation single-molecule sequencing (SMS) technologies, such as Oxford Nanopore Technologies (ONT) [4] and Pacific Bioscience (PacBio) [5] allow to generate longer reads of length 10-30 kb, however they exhibit higher error rates. These technologies might help overcome limitations that short-reads have. Variant calling tools that were developed for short-reads, such as GATK HaplotyperCaller [6] and FreeBayes [7] do not show high accuracy when applied to long-reads sequencing, due to significant difference in error rates and type of errors between these two types of reads. Also these tools process data in short windows of a few

hundred bases lengths and are not designed to aggregate haplotype information present in long-reads, which might be crucial for distinguishing true deviations from errors.
A number of new methods of variant calling were developed to work with long-reads, such as Deepvariant [8] and Longshot [3] that use deep learning and pair-Hidden Markov model algorithms to effectively work with such data. Accuracy of these methods can be improved by efficiently using information obtained with different technologies. Spades [9] genome assembly tool is an example of effectiveness of sequence data aggregation.

Longshot variant calling tool is able to process standard long-read sequencing datasets as well as HiFi (CCS) [10] data, but it is unable to usefully combine multiple datasets at the same time. By effectively aggregating data from different sequencing sources we can improve Longshot [3] sensitivity and accuracy. Previous research show that sequencing reads of different profiles and clustering them by their profiles can improve variant calling accuracy.

In this paper we are extending the functionality of Longshot tool to work effectively with multiple datasets. For that we developed a clustering algorithm for read profiles and implemented such feature into Longshot.

3 Preliminaries

3.1 Strings

Definition 3.1. String is a sequence of letters of finite size.
String of length l over alphabet $A = \{1 \dots m\}$ is a map $s : \{1 \dots l\} \to A$. Usually elements of A are denoted as characters for convenience.

Definition 3.2. Alignment of strings is a way of representing them to spot similarities. Alignment of strings s_1 and s_2 of lengths l_1 and l_2 respectively, over alphabet A is a pair of strings $\hat{s}_1$ and $\hat{s}_2$ of length l over alphabet $A \sqcup \{'-'\}$, such that there exists increasing functions $f_i : \{1 \dots l_i\} \to \{1 \dots l\}, i \in \{1, 2\}$ such that $\hat{s}_i \circ f_i = s_i$ and $\mathrm{Im}(f_i) = \hat{s}_i^{-1}(A)$.

Letter $'-'$ represents gap in a string. String $\hat{s}_i$ represents string s_i with inserted letter $'-'$, that was not present in alphabet A, into some places, and function f_i maps indexes of letters in s_i to corresponding indexes in $\hat{s}_i$.

Example 3.1. Alignment of strings $s_1 = CABCAABA$ and $s_2 = ABADBBAD$ over alphabet $\{A, B, C, D\}$.

Initial strings.

$$\left\| \begin{array}{c|cccccccc} s_1 & \mathrm{C} & \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{A} & \mathrm{A} & \mathrm{B} & \mathrm{A} \\ s_2 & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{D} & \mathrm{B} & \mathrm{B} & \mathrm{A} & \end{array} \right\|$$

Aligned strings.

$$\left\| \begin{array}{c|ccccccccc} \hat{s}_1 & \mathrm{C} & \mathrm{A} & \mathrm{B} & \mathrm{C} & \text{-} & \mathrm{A} & \mathrm{A} & \mathrm{B} & \mathrm{A} \\ \hat{s}_2 & \text{-} & \mathrm{A} & \mathrm{B} & \text{-} & \mathrm{A} & \mathrm{D} & \mathrm{B} & \mathrm{B} & \mathrm{A} \end{array} \right\|$$

Definition 3.3. Define score function for an alignment $S(\hat{s}_1, \hat{s}_2)$. The *better* alignment the more score it gets. Different score functions will be introduced later.

Definition 3.4. We are interested in the alignment with the highest possible score. So we define score between two strings as

$$S(s_1, s_2) = \max_{\hat{s}_1, \hat{s}_2} S(\hat{s}_1, \hat{s}_2)$$

Definition 3.5. Substring is continuous piece of a given string.
For a string s of length l, sub-string s_s is a string of length l_s, such that there exists a function $f : \{1 \dots l_s\} \to \{1 \dots l\}$ such that

$$f(i) = i + d$$

$$s \circ f = s_s.$$

Definition 3.6. We want to find similarities between reference string and reads. For this we define local alignment and score for a string and a substring.
For a string s_1 and s_2 of lengths l_1, l_2 correspondingly, define string-sub-string score as

$$S_s(s_1, s_2) = \max\{S(s_s, s_2) \mid s_s \text{ is a sub-string of } s_1\}$$

and corresponding alignment $\hat{s}_1, \hat{s}_2$.

Definition 3.7. For a string s of length l and set of strings $R = \{s_1 \dots s_n\}$ of lengths $\{l_1 \dots l_n\}$ correspondingly, multiple alignment is tuple $\hat{s}, \hat{s}_1 \dots \hat{s}_n$, of strings of length l over alphabet $A \sqcup \{'-'\}$, such that $\sum_{i=1}^{n} S(\hat{s}, \hat{s}_i)$ is maximal.

Definition 3.8. Set of reads R for string s of length l and rate r is

$$R = \{s_s \mid \text{length of } s_s > l, S_s(s, s_s) < r\}$$

3.2 Task

Given reference string s_r and reads R for an unknown target string s_t, we know that $S(s_r, s_t) < D$ and want to find s_t.

Plan:

1. Make multiple alignment of R over s_r.
2. Estimate most likely difference between s_r and s_t.

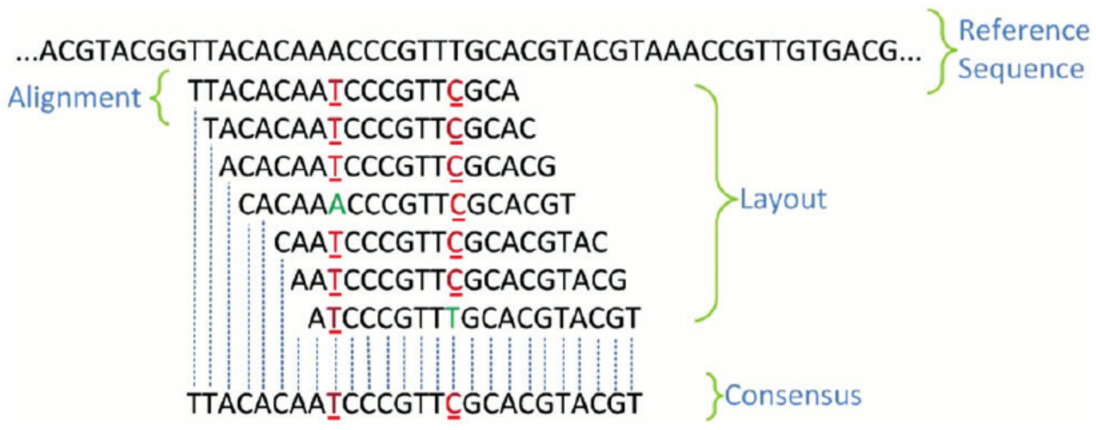

Figure 1: Example of reference string, target string and reads.

3.3 Additive scoring functions

One of the ways to score an alignment is to add award for matches and subtract penalty for mismatches and gaps.

Definition 3.9. For a given matrix $G \in \mathbb{R}^{|A \sqcup \{'-'\}| \times |A \sqcup \{'-'\}|}$ and $p \in \mathbb{R}$ score of alignment $\hat{s}_1, \hat{s}_2$ is

$$S(\hat{s}_1, \hat{s}_2) = \sum_{i=1}^{l} G_{\hat{s}_1(i), \hat{s}_2(i)}.$$

Matrix G stores predefined penalties for mismatches and gaps and encouragement for matches.
Sometimes fines penalty for first gap is higher than prolonging a continuous gap, because one continuous gap is more likely to appear than several small gaps.
Best alignment for such score function can be found using Needleman-Wunsch algorithm.

3.4 Pair Hidden Markov Model

Each step of pairwise alignment can be assigned to one of the three states $\{M, X, Y\}$, where M is a match, X is a gap in s_1, Y is a gap in s_2. Given transition probabilities each alignment can be assigned a probability. The most likely alignment can be found using Viterbi algorithm.

3.5 Clustering

Definition 3.10. Clustering algorithm aims to group points together into predefined number of sets.
Clustering algorithm is a map

$$\text{cluster}(X, m) \to C$$

$$X = \{x_i \mid x_i \in \mathbb{R}^d, i \in (1 \dots n)\},\ m \in \mathbb{N}$$

$$C = \{c_i \mid c_i \in (1 \dots m), i \in (1 \dots n)\}$$

where m is number of clusters and n is number of points.

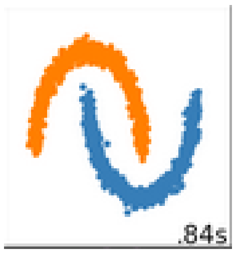

Figure 2: Example of clustering for $d = 2$, $m = 2$, color represents class.

3.6 Stepwise iterative maximum likelihood algorithm (SIML)

To distinguish different read profiles a new method of clustering was developed.

3.7 Likelihood

Assume that the data consists of different multidimensional normal distributions.
Let's denote data as

$$X = \{x_i \mid x_i \in \mathbb{R}^d, i \in (1 \dots n)\}, m \in \mathbb{N}$$

where m is number of clusters and n number of points. Assume that we have some clustering C, we will try to increase it's likelihood.
Denote ith cluster as ω_i and it's estimated parameters as θ_i

$$\Theta = \{\theta_i \mid i \in (1 \dots m)\}.$$

Then probability of x in ith cluster

$$p(x \mid \Theta) = p(x \mid \omega_i, \theta_i) P(\omega_i).$$

Denote clusters as $\chi_1 \dots \chi_l \subset X$, than logarithm of probability for all points in cluster i is

$$\begin{aligned} L_i &= \sum_{x \in \chi_i} \log(p(x \mid \omega_i, \theta_i) P(\omega_i)) \\ &= \log\left(\frac{\exp\left(\frac{-1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right)}{(2\pi)^{d/2}|\Sigma_i|^{1/2}}\right) + n_i \log(P(\omega_i)) \\ &= -\frac{1}{2} n_i d - \frac{n_i d}{2}\log(2\pi) - \frac{n_i}{2}\log|\Sigma_i| + n_i \log\frac{n_i}{n}. \end{aligned}$$

Where μ_i is mean value and Σ_i - covariation of ith cluster. Overall likelihood is

$$L = \sum_{i=1}^{l} L_i.$$

Move $\hat{x}$ from χ_i to χ_j, then

$$\begin{aligned} \Delta L_i = &-\frac{1}{2}\log|\Sigma_i| + \frac{n_i - 1}{2}\log\left(1 - \frac{(\hat{x}-\mu_i)^T \Sigma_i^{-1}(\hat{x}-\mu)}{n_i - 1}\right) + \\ &+ \log\frac{n_i}{n} - (n_i - 1)(\frac{d}{2}+1)\log\frac{n_i - 1}{n_i} \end{aligned}$$

$$\begin{aligned} \Delta L_j = &-\frac{1}{2}\log|\Sigma_j| - \frac{n_j + 1}{2}\log\left(1 + \frac{(\hat{x}-\mu_j)\Sigma_j^{-1}(\hat{x}-\mu_j)}{n_j + 1}\right) + \\ &+ \log\frac{n_j}{n} + (n_j + 1)(\frac{d}{2}+1)\log\frac{n_j + 1}{n_j}. \end{aligned}$$

$$\Delta L = \Delta L_i + \Delta L_j$$

3.8 Algorithm

Main idea

```
1. Initialize clusters (randomly or using another algorithm)
2. Iterate over all points
    2.1 Move point to a cluster, such that overall likelihood increases the most. (With most $\Delta L_j$)
    2.2 Update clusters and their parameters.
3. Repeat step 2 while it makes changes.
```

Advantages

- After every step overall likelihood increases.
- This implies that the cycle will end.

Problems

- Updateting parameters after every step is very slow.

Transition of one point does not change estimated parameters significantly, so to reduce iteration complexity, we will update parameters once in k points. Also to avoid possible loops we will iterate over points in different orders.
Final algorithm:

```
1. Initialize clusters and estimate their parameters
2. Decide $X$ into $p$ random disjoint groups $g_1 \dots g_p$.
3. Loop c from 1 to $p$.
    3.1 Loop $x$ over $g_c$.
        3.1.1 Let $x$ currently be in cluster $i$.
        3.1.2 If $n_i <= 1$, then pass to next point.
        3.1.3 Calculate $\delta_j = \begin{cases} \Delta L_j, & j \neq i \\ \Delta L_i, & j = i \end{cases}$
        3.1.4 Transfer $x$ to argmax($\delta_j$) cluster.
    3.2 Update parameters.
    3.3 If overall likelihood is not increased, revert changes.
4. If any changes were made, repeat step 2.
```

Theorem 1. Proposed algorithm is finite.

Proof. As there are m^n different clusterings, we can order them by their likelihood. After every processed group g_c total likelihood is increased or clustering is not changed, hence we cannot be looped, because we can not increase likelihood infinitely many times, and when no changes were made the algorithm stops. □

4 Main results

4.1 Clustering

The algorithm was tested on data generated by simlord [11] with different generating parameters. For every read

- length,
- number of matches,
- number of mismatches,
- number of insertions (gaps in reference),
- number of deletions (gaps in read),
- number of all transitions from states match, mismatch, insertion, deletions,
- length of left soft-clipping (continuous region of mismatches before first match, insertion or deletion),
- length of right soft-clipping (continuous region of mismatches after last match, insertion or deletion).

were calculated.
For clustering

- number of all transitions from states match, mismatch, insertion, deletions,
- length of left soft-clipping,
- length of right soft-clipping

were used.
To make read's profile independent from it's length, all transitions were divided by length of the read. Also all features were normalized by their standard deviations, to equal their contribution when calculate distance between points. After that PCA method [12] was applied to the data, to reduce linear dependencies and extract important information. To make comparison with existing clustering algorithms, preproccessed data was clustered by 3 groups using Kmeans [13], Gaussian mixture [13] and SIML clustering algorithms. Axes represent two main PCA components, color represent cluster.

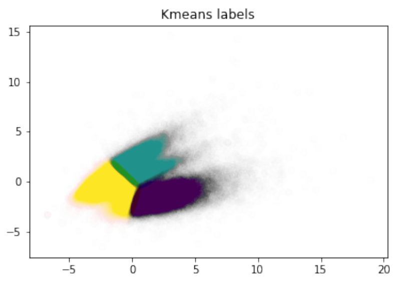

Figure 3: K-Means

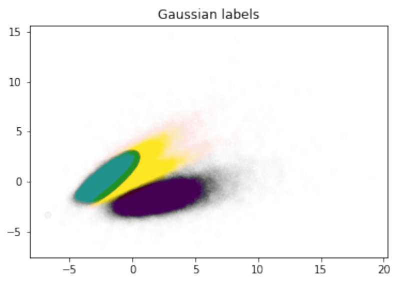

Figure 4: GaussianMixture

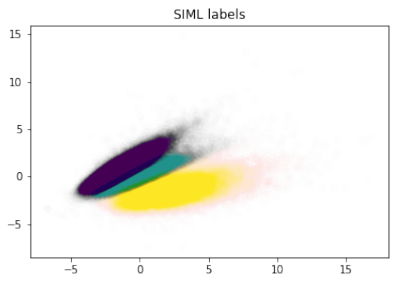

Figure 5: SIML

For SIML we can see how likelihood of clustering changed over iterations.

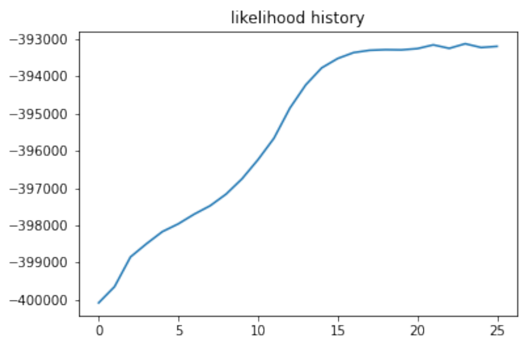

Figure 6: Likelihood history over iterations.

More precise testing is yet to be completed. Algorithm was implemented using python.

4.2 Longshot

Fork of Longshot [3] with features of clustering and processing multiple datasets is in development.

References

[1] S. Goodwin, J. D. McPherson, and W. R. McCombie, "Coming of age: ten years of next-generation sequencing technologies," *Nature Reviews Genetics*, vol. 17, no. 6, pp. 333–351, 2016.

[2] N. H. G. R. Institute, "Genetics vs. genomics fact sheet," 2022.

[3] P. Edge and V. Bansal, "Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing," *Nature Comm.*, 2019.

[4] M. Jain, H. E. Olsen, B. Paten, and M. Akeson, "The oxford nanopore minion: delivery of nanopore sequencing to the genomics community," *Genome biology*, vol. 17, no. 1, pp. 1–11, 2016.

[5] A. Rhoads and K. F. Au, "Pacbio sequencing and its applications," *Genomics, proteomics & bioinformatics*, vol. 13, no. 5, pp. 278–289, 2015.

[6] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, *et al.*, "The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data," *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.

[7] E. Garrison and G. Marth, "Haplotype-based variant detection from short-read sequencing," *arXiv preprint arXiv:1207.3907*, 2012.

[8] R. Poplin, P.-C. Chang, D. Alexander, S. Schwartz, T. Colthurst, A. Ku, D. Newburger, J. Dijamco, N. Nguyen, P. T. Afshar, *et al.*, "A universal snp and small-indel variant caller using deep neural networks," *Nature biotechnology*, vol. 36, no. 10, pp. 983–987, 2018.

[9] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, *et al.*, "Spades: a new genome assembly algorithm and its applications to single-cell sequencing," *Journal of computational biology*, vol. 19, no. 5, pp. 455–477, 2012.

[10] A. M. Wenger, P. Peluso, W. J. Rowell, P.-C. Chang, R. J. Hall, G. T. Concepcion, J. Ebler, A. Fungtammasan, A. Kolesnikov, N. D. Olson, *et al.*, "Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome," *Nature biotechnology*, vol. 37, no. 10, pp. 1155–1162, 2019.

[11] B. K. Stöcker, J. Köster, and S. Rahmann, "Simlord: simulation of long read data," *Bioinformatics*, vol. 32, no. 17, pp. 2704–2706, 2016.

[12] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.