Clustering of Multidimensional Random Variables to Improve HMM Sequence Alignment Accuracy

Denis Grachev

April 16, 2022

Introduction to bioinformatics

- The most useful biological data is DNA (genome) sequence.
- Each chromosome is represented as a string over alphabet $\{A, C, G, T\}$.
- Most species are diploid $\Rightarrow$ each chromosome is paired.
- It is very large (human genome is 6.4 billion bp).
- Genomes of unrelated humans are 99.9% similar.
- Determine reference genome and identify difference for any individual.

Reading genome

- Read random peaces of genome (reads).
- Find similar peaces in reference genome (Alignment).
- Estimate difference (variant calling).

Example

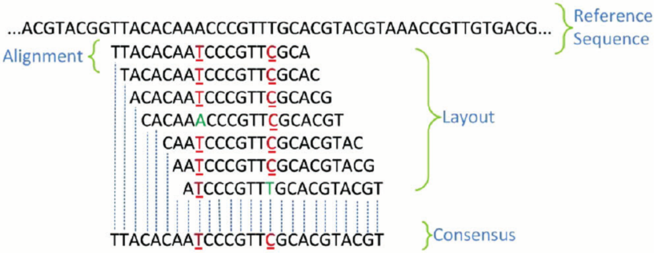

Figure: Reading genome example.

Alignment

Idea

Assume that string s_1 was obtained from s_2 by applying 3 types of errors to it.

- Substitution. Replace one letter with another.
- Insertion. A letter was inserted to string.
- Delition. A letter was deleted from string.

Task is to find most likely sequence of errors for given s_1 and s_2.

Alignment

Example

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

Initial strings.

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

Aligned strings.

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

Representing alignment as pair HMM

Hidden states

- M: Match or Mismatch.
- X: Insertion to s_1.
- Y: Insertion to s_2 (Delition).

Observations

- M: $\{(x, y) \mid x, y \in \Sigma\}$.
- X: $\{(x, -) \mid x \in \Sigma\}$.
- Y: $\{(-, y) \mid y \in \Sigma\}$.

Representing alignment as pair HMM

Example

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

Initial strings.

s_1	C	A	B	C	A	A	B	A
s_2	A	B	A	D	B	B	A	

Aligned strings.

$\hat{s}_1$	C	A	B	C	-	A	A	B	A
$\hat{s}_2$	-	A	B	-	A	D	B	B	A
O	X	M	M	X	Y	M	M	M	M

Clustering

Clustering

Given $X = \{x_i | x_i \in \mathbb{R}^d, i \in (1 \ldots n)\}$ and l - number of clusters. Clustering is assigning each point to one cluster $C = \{c_i | c_i \in (1 \ldots l), i \in (1 \ldots n)\}$.

Example

for $\mathbb{R}^2$ and $l = 2$

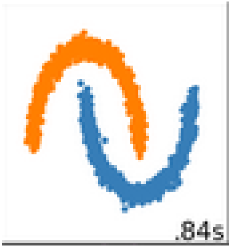

Figure: Example of clustering

Genome

Defenition

Reference genome - S_r string over alphabet $\{A, C, G, T\}$.
Example genome - S_e string over alphabet $\{A, C, G, T\}$.
Example genome does not differ much from reference genome.
Reads - $R = \{r_i | r_i - \text{random substring from } S_e \text{ with errors}\}$

Task

We have S_r and R, want to find out S_e.

Allignment

Explanation

To find out S_e we try to locate substrings from R on S_r with least possible amount of errors. This process is called alignment.
Then we try to estimate most likely difference between S_e and S_r. This process is called variant calling.

Example

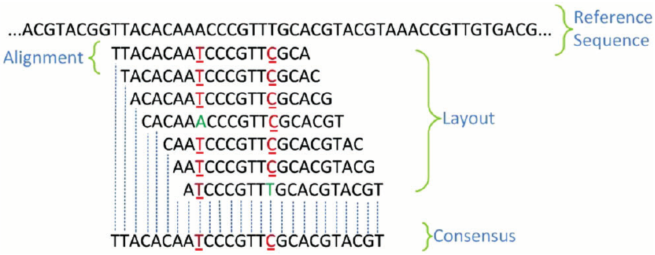

Figure: Example of alignment

Markov chains

The probabilities of transitions are not equal.

We can use it for better accuracy of variant calling.

Also these probabilities varies depending on a region.

For every read we can calculate transition probabilities and cluster them.

Then use probabilities obtained in each cluster for corresponding reads.

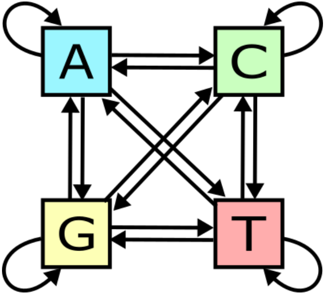

Figure: transitions

Algorithm

Preprocess

1. For each read calculate each transitions.
2. Each feateture is devided by standart deviation.
3. PCA method is applied for resulting data.

Assume that distribution in each cluster is multidimensional normal distribution.

Likelihood

$\Theta = \{\theta_i | \theta_i \text{ - parameters of ith cluster}\}$
We can estimate parameters of each cluster based on X and C.
Denote ith class as ω_i, then probability of x belong to ith cluster is

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

Algorithm

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

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

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

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

Algorithm

Move $\hat{x}$ from χ_i to χ_j, then

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

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

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

Algorithm

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 ΔL_j)
   2.2 Update clusters and their parameters.
3. Repeat step 2 while it makes changes.

Advantage

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

Problem

- Updateting parameters after every step is very slow.

Algorithm

Fix 1

- Update parameters every k points.
- If overall likelihood decreased, revert changes.

Bad

- Can stuck in a loop, transfering and reverting same points.

Fix 2

- Pick points randomly and apply algorithm for them. Then repeat for another points.

Algorithm

Fixed

```
1. Initialize clusters and estimate their parameters
2. Devide $X$ into $p$ random disjoined groups $g_1 \dots g_p$.
3. Loop c from 1 to $p$.
    3.1 Loop $x$ over $g_c$.
    3.2 Let $x$ currently be in cluster $i$.
        3.2.1 If $n_i <= 1$, then pass to next point.
        3.2.2 Calculate $\delta_j = \begin{cases} \Delta L_j, & j \neq i \\ \Delta L_i, & j = i \end{cases}$
        3.2.3 Trasfer $x$ to argmax$(\delta_j)$ cluster.
    3.3 Update parameters.
    3.4 If overall likelihood decreased, revert changes.
4. If any changes were made, repeat step 2.
```

Algorithm

We can cluster reads and use different Markov models for each cluster.