

به نام خداوند بخشنده و مهربان

مبانی بیوانفورماتیک

گزارش تئوری پروژه درس

محسن نقی پورفر ۹۴۱۰۶۷۵۷

قسمت اول

همانطور که می‌دانیم، پروتئین‌های ویروسی (Viral Proteins)، پروتئین‌هایی هستند که یا جزء ساختار ویروس هستند یا محصول آن. این پروتئین‌ها با توجه به عملکردشان گروه‌بندی می‌شوند. حال دو گروه از این پروتئین‌ها ساختاری (Structural) و غیرساختاری (NonStructural) می‌باشند. البته گروه‌های دیگری نظیر تنظیم‌کنندگی (Regulatory) و جانبی (Accessory) نیز وجود دارند. با توجه به اینکه ویروس‌ها در حالت کلی، موجودات زنده نیستند، پس به متابولیسم سلول‌های میزبان برای انرژی، آنزیم‌ها و ... برای تولیدمثل بستگی دارند. حال پروتئین‌های ساختارمند، معمولاً اجزای کپسید و Envelope ویروس می‌باشند. در مقابل، پروتئین‌های غیرساختاری، پروتئین‌های کد شده برای ژنوم ویروس هستند و در سلولی که عفونی شده بیان می‌شوند. با این حال، این پروتئین‌ها در ویروئین‌ها در مقابل، پروتئین‌های غیرساختاری و ویروس‌ها بعضی از پروتئین‌های غیرساختاری عملکرد مهمی دارند که در فرآیند تولید مثل تأثیر می‌گذارند. [۴] برای پروتئین‌های حاصل از بیان ژن این ویروس، می‌دانیم که GP, ssGP, LsGP پروتئین‌های غیرساختاری، GP پروتئین ساختاری، می‌باشند. [۹] همچنین ژنوم این ویروس برای ۷ پروتئین ساختاری و یک پروتئین غیرساختاری کپی می‌کند. [۱۰] همچنین VP24 نیز، با توجه به اینکه مشاهده شده که در فرآیند Budding و Assembly آن نقش دارد، یک پروتئین ساختاری می‌باشد. همچنین این پروتئین ویژگی‌های ساختاری دارد که با پروتئین‌های ماتریسی ویروسی (Viral Matrix Proteins) مشترک می‌باشد. [۸] پروتئین VP35 نیز مانند این پروتئین، ساختاری می‌باشد. [۱۰]

قسمت دوم

در این قسمت عملیات Alignment و بدست آوردن ماتریس فاصله را به صورت دستی پیاده‌سازی کرده‌ایم. این پیاده‌سازی‌ها در توابع فایل Alignment.Py و همچنین در دفترچه jupyter می‌باشند. الگوریتم glocal_alignment که در کد پیاده‌سازی شده است، در واقع یک هم‌ترازی سراسری است که برای ژنوم gap_penalty را در نظر نمی‌گیریم و برای خانه‌های جدول در صورتی که ژنوم ستون‌های آن و ژن سطرهای آن باشد، داریم:

$$\begin{aligned} scores[i][0] &= i \times GapPenalty \\ scores[0][j] &= 0 \end{aligned} \quad (1)$$

این الگوریتم، در واقع یک الگوریتم Global-Local Alignment می‌باشد و محل ژن موردنظر را در ژنوم بهتر از الگوریتم‌های global و local پیدا می‌کند. همچنین برای بدست آوردن Edit Distance در نیز در پیاده‌سازی الگوریتم، تعداد Indel و Missmatch‌ها را به عنوان این فاصله و پیرایشی بازگردانده‌ایم.

قسمت سوم

ابتدا توضیح مختصری در مورد نحوه کار الگوریتم‌های UPGMA و NP می‌دهیم و سپس به تفاوت‌های آنها می‌پردازیم.

UPGMA

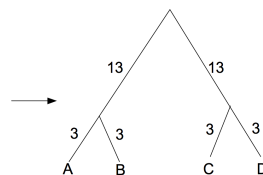
الگوریتم UPGMA یک الگوریتم ساخت درخت بر مبنای ماتریس فاصله است که فرض می‌کند که ماتریس دارای خاصیت Additive بودن و داده‌ها به صورت Ultrametric می‌باشند. این الگوریتم شبیه خوشه‌بندی سلسه‌مراتبی (Hierarchical Clustering) عمل می‌کند. خاصیت Ultrametric بودن در واقع یک خاصیت قوی‌تر از Additivity می‌باشد، به این معنی که اگر ماتریس دارای خاصیت Ultrametric باشد، Additive هم هست. [۱] این الگوریتم برای هر دو خوشه، پارامتری به نام $D(C, C^*)$ را محاسبه می‌کند که از روی ماتریس فاصله بدست می‌آید. شکل مربوط به رابطه این پارامتر در زیر آمده است. [۱]:

for every cluster $C^* \neq C$

$$D(C, C^*) = \frac{1}{|C| \cdot |C^*|} \sum_{i \in C} \sum_{j \in C^*} D(i, j)$$

همچنین شکلی از نحوه اجرای آن روی یک ماتریس فاصله در زیر آمده است.

	A	B	C	D
A		6	32	32
B			32	32
C				6
D				

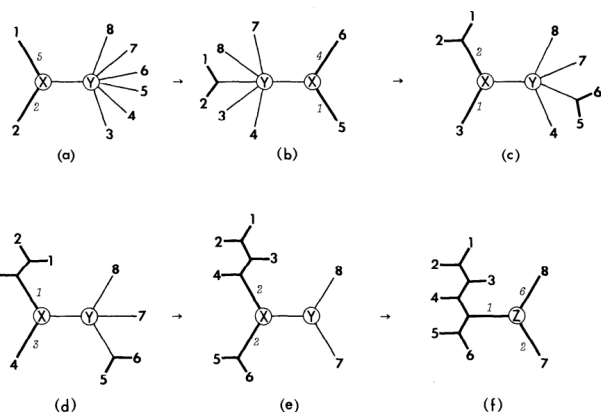


Neighbor Joining

الگوریتم UPGMA یک الگوریتم ساخت درخت بر مبنای ماتریس فاصله است که فرض می‌کند که ماتریس دارای خاصیت Additive بودن و داده‌ها به صورت Ultrametric می‌باشند. در این الگوریتم یک درخت بدون ریشه ساخته می‌شود. این الگوریتم به این صورت عمل می‌کند که پارامتری را برای هر جفت گونه (هر گونه یک سطر و ستون ماتریس فاصله می‌باشد) تعریف می‌کند که بر اساس آن معیار دو گونه را انتخاب می‌کند و توسط یک راس دیگر (راس والد) به هم متصل می‌کند و به جای آن دو در ماتریس فاصله پدرشان را با فاصله‌های جدید نسبت به گونه‌های دیگر اضافه می‌کند. در واقع دو سطر و دو ستون از ماتریس فاصله را حذف کرده و یک سطر و ستون جدید را به آن اضافه می‌کند. این کار را تا زمانی ادامه می‌دهد که ماتریس فاقد سطر ستون نباشد. این الگوریتم، به صورت حریصانه

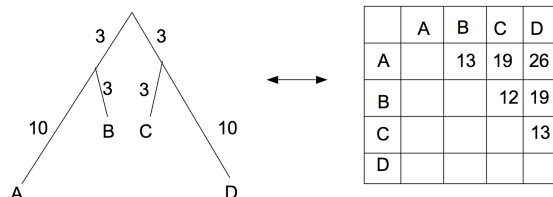
عمل می کند و یک نتیجه ای تولید می کند که به اصطلاح یک جواب subOptimal می باشد. [۱] فرض کنید ماتریس فاصله D باشد. آنگاه این الگوریتم در هر مرحله متغیر L_{XY} را برای هر دو گونه تعریف می کند و هر بار به صورت حریصانه کمترین L_{XY} را انتخاب می کند و دو راس را باهم ترکیب می کند و فاصله هارا بروزرسانی می کند. در واقع این الگوریتم علاوه بر ماتریس فاصله D یک ماتریس L هم تعریف می کند. همچنین همانطور که مشخص است، این الگوریتم دارای زمان اجرای چند جمله ای است. پس برای اعمال روی داده های بزرگ مناسب می باشد. شکل [۶، ۷] مربوط به نحوه اجرای این الگوریتم و رابطه L_{XY} در زیر آمده است:

$$L_{XY} = \frac{1}{2(N-2)} \left[\sum_{k=3}^N (D_{1k} + D_{2k}) - (N-2)(L_{1x} + L_{2x}) - 2 \sum_{i=3}^N L_{iY} \right]$$



مقایسه دو الگوریتم

در این قسمت که از الگوریتم های NJ و UPGMA استفاده شده است، بین نتایج بدست آمده از درخت ها تفاوت هایی مشاهده می شود. همانطور که می دانیم الگوریتم NJ نیاز به ویژگی Additive بودن ماتریس دارد اما الگوریتم UPGMA علاوه بر این ویژگی نیاز به داشتن Ultrametric بودن آنها نیز دارد. همچنین درخت ساخته در این دو الگوریتم رویه متفاوتی را را برای ساخته شدن طی می کند و همانطور که می دانیم، درخت حاصل از UPGMA درختی ریشه دار است اما درخت حاصل از NJ درختی فاقد ریشه می باشد. [۲] همچنین باتوجه به نتایج و تجربه های قبلی در استفاده از الگوریتم UPGMA، این الگوریتم، قابل اعتماد نیست و به درستی گونه ها را تحلیل نمی کند. [۳] در نتیجه استفاده از الگوریتم NJ بهتر است. در واقع الگوریتم UPGMA برای ساخت درخت از روی ماتریس فاصله مناسب نمی باشد و تنها داده های Ultra metric برای این الگوریتم خوب هستند و نتایج خوبی تولید می کنند و در بقیه موارد این الگوریتم به درستی عمل نمی کند (مانند نتایج بدست آمده در همین پروژه). همچنین الگوریتم NJ دارای خاصیت سریعتر بودن نیز نسبت به سایر الگوریتم (UPGMA و متد های Parsimony و ...) های ساخت درخت می باشد و این نشاندهنده مناسب بودن آن برای داده های بزرگ می باشد چرا که سریعتر از بقیه به جواب می رسد (زیرا زمان اجرای آن چند جمله ای است). همچنین یک برتری دیگر الگوریتم NJ این است که این الگوریتم، این فرض را نمی پذیرد که تمام خطوط با سرعت ثابت تکامل پیدا می کنند. شکل زیر، مثال دیگری از نحوه اجرای اشتباه الگوریتم UPGMA به صورت کلی و روی ماتریس Non-Ultrametric می باشد. [۴] تمام عکس های مربوط به نتایج حاصل از این دو الگوریتم، در پوشه images در مخزن گیت هاب و کنار همین توضیحات آمده است.



قسمت چهارم

این قسمت در فایل Section4.R پیاده سازی شده است و با زبان R زده شده است.

- [1] Neil C. Jones, Pavel A. Pevzner, An Introduction to Bioinformatics Algorithms
- [2] <https://www.biostars.org>
- [3] <https://www.researchgate.net>
- [4] <http://wikipedia.org/>
- [5] <http://biopython.org>
- [6] Naruya Saitou, Masatoshi Nei, The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees (1987).
- [7] Terry Speed, Neighbour Joining Method (Saitou and Nei, 1987), Berkely Slide, 2006.
- [8] Ziyang Han, Hani Boshra, J. Oriol Sunyer, Susan H. Zwiers, Jason Paragas, and Ronald N. Harty, Biochemical and Functional Characterization of the Ebola Virus VP24 Protein: Implications for a Role in Virus Assembly and Budding
- [9] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2829775/>
- [10] <http://www.ebolavirusnet.com/ebola-virus.html>