به نام خداوند بخشنده و مهربان

# مبانی بیوانفورماتیک

## گزارش تئوری پروژه درس

محسن نقی‌پورفر ۹۴۱۰۶۷۵۷

# قسمت اول

# قسمت دوم

در این قسمت عملیات Alignment و بدست آوردن ماتریس فاصله را به صورت دستی پیاده سازی کرده ایم. این پیاده سازی ها در توابع فایل Alignment.Py و همچنین در دفترچه jupyter می‌باشند. الگوریتم glocal_alignment که در کد پیاده سازی شده است، در واقع یک هم‌ترازی سراسری است که برای ژنوم gap_penalty را در نظر نمی‌گیریم و برای خانه های جدول در صورتی که ژنوم ستون های آن و ژن سطرهای آن باشد، داریم:

$$scores[i][0] = i \times GapPenalty$$
$$scores[0][j] = 0 \quad (۱)$$

این الگوریتم، در واقع یک الگوریتم Global-Local Alignment می‌باشد و محل ژن موردنظر را در ژنوم بهتر از الگوریتم های globalو local پیدا می کند. همچنین برای بدست آوردن Edit Distance در نیز در پیاده سازی الگوریتم،تعداد Indel ها و Missmatchها را به عنوان این فاصله ویرایشی بازگردانده‌ایم.

# قسمت سوم

ابتدا توضیح مختصری در مورد نحوه کار الگوریتم های UPGMAو NP می‌دهیم و سپس به تفاوت های آنها می‌پردازیم.

## UPGMA

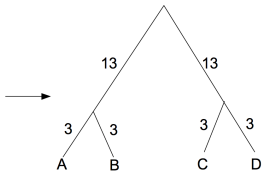
الگوریتم UPGMA یک الگوریتم ساخت درخت بر مبنای ماتریس فاصله است که فرض می کند که ماتریس دارای خاصیت Additive بودن و داده‌ها به صورت Ultrametric می‌باشند.این الگوریتم شبیه خوشه‌بندی سلسه‌مراتبی(Hierarchical Clustereing) عمل می کند.خاصیت Ultrametric بودن در واقع یک خاصیت قوی‌تر از Additivity می‌باشد، به این معنی که اگر ماتریس دارای خاصیت Ultrametric باشد، Additiveهم هست. [۱] این الگوریتم برای هر دو خوشه، پارامتری به نام $D(C, C^*)$ را محاسبه می کند 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.

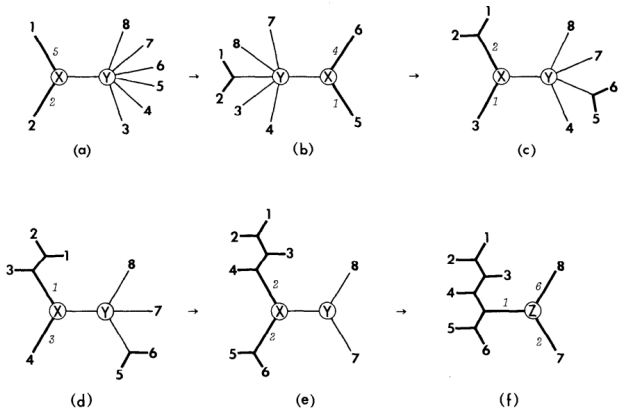
| | A | B | C | D |
|---|---|---|---|---|
| A | | 6 | 32 | 32 |
| B | | | 32 | 32 |
| C | | | | 6 |
| D | | | | |




## Neighbor Joining

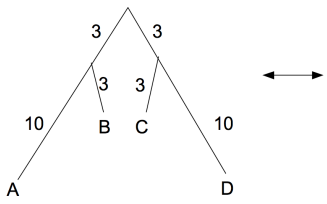
الگوریتم UPGMA یک الگوریتم ساخت درخت بر مبنای ماتریس فاصله است که فرض می کند که ماتریس دارای خاصیت Additive بودن و داده‌ها به صورت Ultrametric می‌باشند. در این الگوریتم یک درخت بدون ریشه ساخته می‌شود. این الگوریتم به این صورت عمل می کند که پارامتری را برای هر جفت گونه(هر گونه یک سطر و ستون ماتریس فاصله می‌باشد) تعریف می کند که بر اساس آن معیار دو گونه را انتخاب می کند و توسط یک راس دیگر(راس والد) به هم متصل می کند و به جای آن دو در ماتریس فاصله پدرشان را با فاصله های جدید نسبت به گونه های دیگر اضافه می کند. در واقع دو سطر و دو ستون از ماتریس فاصله را حذف کرده و یک سطر و ستون جدید را به آن اضافه می کند. این کار را تا زمانی ادامه می‌دهد که ماتریس فاقد سطر ستون بشود. این الگوریتم ، به صورت حریصانه عمل می کند و یک نتیجه‌ای تولید می کند که به اصطلاح یک جواب subOptimal می‌باشد.[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 جمله‌ای است. پس برای اعمال روی داده های بزرگ مناسب می‌باشد. 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 می‌شود. همانطور که می‌دانیم 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 ساخته‌شدن طی می‌کند و همانطور که می‌دانیم، درخت حاصل از UPGMA درختی ریشه دار است اما درخت حاصل از NJ درختی فاقد ریشه می‌باشد. [۲] همچنین باتوجه به نتایج و تجربه‌های قبلی در استفاده از الگوریتم UPGMA، این الگوریتم، قابل اعتماد نیست و به درستی گونه ها را تحلیل نمی‌کند.[۳] در نتیجه استفاده از الگوریتم NJ بهتر است. در واقع الگوریتم UPGMA برای ساخت درخت از روی ماتریس فاصله مناسب نمی‌باشد و تنها داده‌های Ultra metric برای این الگوریتم خوب هستند و نتایج خوبی تولید می‌کنند و در بقیه موارد این الگوریتم به درستی عمل نمی‌کند(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 می‌باشد و این نشاندهنده مناسب بودن آن برای داده های بزرگ می‌باشد چرا که سریعتر از بقیه به جواب می‌رسد(زیرا زمان اجرای آن چند جمله‌ای است).همچنین یک برتری دیگر الگوریتم NJ این است که این الگوریتم، این فرض را نمی‌پذیرد که تمام خطوط با سرعت ثابت تکامل پیدا می‌کنند. شکل زیر، مثال دیگری از نحوه اجرای اشتباه الگوریتم UPGMA به صورت کلی و روی ماتریس Non-Ultrametric می‌باشد.[۴] تمام عکس های مربوط به نتایج حاصل از این دو الگوریتم، در پوشه images در مخزن گیت‌هاب و کنار همین توضیحات آمده است.




| | A | B | C | D |
|---|---|---|---|---|
| A | | 13 | 19 | 26 |
| B | | | 12 | 19 |
| C | | | | 13 |
| D | | | | |

# قسمت چهارم

# مراجع


[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.