زهرا گنجی 9531802

پیاده‌سازي روش مدل مخفی مارکوف گسسته براي ساخت پروفایل

در این پروژه قصد داریم روش hmm را برای ساخت پروفایل از روی یک هم ترازی چندگانه پیاده سازی کنیم.

خواندن رشته های ورودی:

برای این کار بعد از خواندن رشته های ورودی و قرار دادن آن ها در آرایه ی msa_sequences، باید تشخیص دهیم دنباله ها مربوط به پروتئین هستند یا DNA . سپس تمام باقی مانده های ممکن آن نوع دنباله ها و گپ را در مجموعه ی X قرار می دهیم و طول X را برابر T قرار می دهیم. از این X به عنوان observation در ماتریس های a و b و pi استفاده می گردد.

```
# recognize that we have DNA or AminoAside sequences
T = 0
X = []
seq_type = ''
characters = []
for seq in msa_sequences:
    for i in seq:
        if i not in characters:
            characters.append(i)

for char in characters:
    if char != 'A' and char != 'T' and char != 'G' and char != 'C' and char != '-':
        seq_type = 'Protein'
        T = 21
        X = ['K', 'R', 'H', 'D', 'E', 'N', 'Q', 'S', 'T', 'Y', 'A', 'V', 'L', 'I', 'P', 'F', 'M', 'W', 'G', 'C',
             '-']
        break

if T == 0:
    seq_type = 'DNA'
    T = 5
    X = ['A', 'T', 'G', 'C', '-']

# remove columns if the fraction of gap symbols (“-”) exceeds m
```

حذف ستون های با تعداد گپ بیش تر از threshold یا m:

سپس شماره ستون هایی که تعداد گپ کم تر از m دارند را در acceptable_columns قرار می دهیم وبه تعداد این ستون ها match state و deletion state داریم. تعداد insertion state ها یکی از تعداد match state ها بیش تر است. همچنین دو حالت SوE را برای شروع و پایان داریم. پس بدین ترتیب تعداد کل state ها را بدست می آوریم.

```
# remove columns if the fraction of gap symbols (“-”) exceeds m
# acceptable columns are those columns that the number of gaps in them are less than m
acceptable_columns = []
# acceptable_columns.append(0)
for i in range(len(msa_sequences[0])):
    gap_number = 0
    for j in msa_sequences:
        if j[i] == '-':
            gap_number = gap_number + 1

    if gap_number < int(m):
        acceptable_columns.append(i)
# acceptable_columns.append(len(msa_sequences[0])-1)

# M is the number of match states
M = len(acceptable_columns)

# we have M+1 insertion states and M deletion states , N is the total number of states
N = M + M + 1 + M + 2
```

ساخت توپولوژی:

سپس توپولوژی را می سازیم. و ترتیبی از state ها که در ماتریس ها استفاده می کنیم را مشخص می کنیم.

Stateها به این ترتیب هستند: S,I0,M1,I1,D1,.....,E

مقدار دهی اولیه ماتریس های a و b و pi:

- ماتریس b

سپس ماتریس های a و bو pi را مقداردهی اولیه می کنیم. برای مقدار دهی b، برای match state ها در هر observation به جز گپ، تعداد آن observation در ستون مربوطه در acceptable_columns را تقسیم بر تعداد کل باقی مانده ها در آن ستون به جز گپ می کنیم و برای گپ نیز در match state همیشه احتمال emission probability صفر است.

برای insertion state ها، مقدار اولیه را برای همه ی کاراکتر ها به جز گپ 1/(T-1) قرار می دهیم و برای گپ 0 قرار می دهیم. این مقادیر بعد از هر بار train شدن باید دوباره به همین مقادیر اولیه بازگردند.

برای deletion stateها ، فقط در گپ مقدار 1 می گیرند، و در بقیه ی کاراکتر ها مقدار 0 می گیرند و تا انتها 0 باقی می مانند.

- ماتریس a

برای ماتریس a نیز فقط یال هایی که می توانند وجود داشته باشند، احتمال transition probability را مقدار دهی اولیه می کنیم و بقیه ی احتمالات در صورتی که یالی وجود نداشته باشند باید تا انتها مقدار 0 داشته باشند.

- ماتریسpi

بردارpi نیز تنها در start state مقدار 1 می گیرد و برای بقیه ی مقادیر تا انتها 0 است.

آموزش مدل با الگوریتم baum-welch:

بعد از مقداردهی اولیه، مدل را با دنباله های ورودی آموزش می دهیم. در یک for با 10 بار تکرار، هر بار ماتریس های آلفا، بتا، گاما و سای را طبق فرمول های موجود در تئوری hmm، برای هر دنباله ایجاد می کنیم و از این ماتریس ها را در یک لیست نگه می داریم و برایlearning و آپدیت کردن مقادیر ماتریس های a و b استفاده می کنیم. در learning تنها درایه های غیر 0 در ماتریس های a و b با مقادیر جدید update می شوند. چون برای مثال در ماتریس a مقادیر 0 به معنی یال هایی است که اصلا وجود ندارند و احتمال transition در آن ها باید 0 بماند.

بعد از learning و آپدیت شدن مقادیر ماتریس ها در هر مرحله، مقادیر آن ها چک می کنیم و آن ها را با pseudo count آپدیت می کنیم. در ضمن چک میکنیم که احتمال transition از match به match وdeletion به deletion و insertion به insertion خیلی کم نشود.

بعد از 10 بار تکرار این حلقه ، از مقادیر ماتریس های آموزش دیده شده ی a وb و pi برای الگوریتم Viterbi و بدست آوردن محتمل ترین دنباله state برای sequenceداده شده استفاده می کنیم.

الگوریتم Viterbi:

در این تابع برای هر state یک تاپل تعریف می کنیم. این تاپل شامل اندیس state، کاراکتر های ایجاد شده تا این state، t (شماره observation) و دلتای state در t مورد نظر می باشد.

Path_list_item شامل تاپل مرتبط با state و t ای که می خواهیم پردازش کنیم، است. در ابتدا در این لیست فقط تاپل مربوط به start state با t=0 و کاراکتر – وجود دارد.

در هر پردازش stateهایی که از state مورد نظر قابل دسترسی هستند(یعنی بین آن ها یال وجود دارد) را بدست می آوریم و دلتای آن ها را بدست می آوریم و از بین لیست تاپل های بدست آمده با مقادیر t یکی بیش تر از t مرتبط با تاپل پردازش شده، تاپل مرتبط با آن state ای که بیش ترین دلتا را دارد را در max_path_list قرار می دهیم و در مرحله ی بعد آن را پردازش می کنیم.

t در تاپل مرتبط با deletion states با t در تاپل مرحله ی قبل فرقی نمی کند و اضافه نمی شود. ولی در صورتی که از پردازش یک تاپل به تاپل های مربوط به insertion و deletion برسیم باید مقدار t یکی بیش تر از قبل باشد. همچنین در تاپل های deletion، در قسمت کاراکتر، کاراکتر – اضافه می شود، در حالی که در تاپل های insertion ها وdeletion ها کاراکتر بعدی در sequence ورودی که می خواهیم Viterbi را روی آن انجام دهیم اضافه می شود.

در انتها در هر مرحله (در هر t) تاپل مرتبط با state دارای بیش ترین دلتا را وارد لیست max_list_path کرده ایم. آخرین درایه ی آن کل دنباله ی stateهای محتمل اعمال شده روی sequence ورودی Viterbi را به ما باز میگرداند.

خروجی برای تست کیس ها با فایل های ورودی input1 تا input6: