PTSIPFRFR - Predicting Transmembrane Sequence In Proteins For Real For Real

מאת

איליה אדמסקי וטל אור

עבודה זו מוגשת כעבודה בהיקף של 2 יח' כמילוי חלקי של הדרישות לקראת קבלת ציון במדע חישובי ביולוגיה (Bioinformatics)

עבודה זו בוצעה בהדרכת <u>הילה פייליב</u>

יוני 2024

תמצית

חלבונים טרנסממברנליים הם חלבונים אשר עוברים את ממברנת התא לפחות פעם אחת.
החלבונים הטרנס ממברנלים מורכבים משני סוגי מקטעים: מקטעים שנמצאים בתוך הממברנה - טרנסממברנליים ומקטעים מחוץ לממברנה.
בפרויקט שלנו ניסינו לחזות מקטעים טנרסממברנליים בחלבונים טרנסממברנליים בעזרת אלגוריתמים סטטיסטיים.
בעבודה הצלחנו לחזות את מקטעים אלו בדיוק של מעל 70% באופן כללי, עם ציון נוסף שיכול לשמש כמדד ביטחון החיזוי (confidence score). בנוסף נשווה בעבודה זאת, את ביצועי המודל שלנו, בהשוואה ל-TMHMM 2.0[1].

[1]Krogh A, Larsson B, von Heijne G, Sonnhammer EL. Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. J Mol Biol. 2001 Jan 19;305(3):567-80. doi: 10.1006/jmbi.2000.4315. PMID: 11152613.

השתמשנו בכלי הנמצא באתר זה על מנת לקבל את חיזויים עם כלי זה
https://services.healthtech.dtu.dk/services/TMHMM-2.0/

מבוא

הצגת הבעיה

חלבונים טרנסממברנליים ממלאים תפקיד חשוב בתהליכים ביולוגיים שונים, כולל העברת אותות, הובלת מולקולות ותקשורת בין תאים. חלבונים אלו חוצים את שכבת הליפידים של ממברנות התאים וניתן לסווגם לשני סוגים עיקריים: הליקלי ובטא-חביתי. זיהוי ואפיון מדויקים של אזורים טרנסממברנליים בתוך חלבונים אלו הם חיוניים להבנת תפקודם ולגילוי תרופות. עם זאת, חיזוי אזורים טרנסממברנליים מתוך רצפי חלבונים היא משימה מאתגרת בשל המבנה המורכב של חלבונים והשתנותם ברצפים.

האתגר העיקרי בפתרון בעיה זו הוא זיהוי מדויק של אזורים טרנסממברנליים בתוך רצפי חלבונים. שיטות ניסיוניות מסורתיות, כגון קריסטלוגרפיה בקרני רנטגן ומיקרוסקופיה קריו-אלקטרונית, הן זמן-רבות ויקרות. לכן, גישות חישוביות הפכו לחשובות יותר ויותר לחיזוי אזורים טרנסממברנליים. גישות אלו מנצלות ניתוחים סטטיסטיים ומודלים חישוביים לניתוח רצפי חלבונים ולחיזוי אזורים טרנסממברנליים.

בפרויקט זה, פיתחנו מודל חישובי לחיזוי אזורים טרנסממברנליים בחלבונים טרנסממברנליים המודל משתמש בניתוח סטטיסטי לניתוח רצפי חלבונים וזיהוי אזורים טרנסממברנליים. הסביבה החישובית לפרויקט זה כוללת את שפת התכנות Python, יחד עם ספריות כגון pandas, requests ו-re לעיבוד וניתוח נתונים.

סקירת ספרות

נערכו מספר מחקרים כדי להתמודד עם בעיית חיזוי אזורים טרנסממברנליים בחלבונים. ג'ונס ואח' (1994) הציעו גישה לזיהוי מודלים לחיזוי מבנה וטופולוגיה של חלבונים ממברנליים הליקליים. גישה זו משתמשת בניתוח סטטיסטי לחישוב ההסתברות של כל חומצת אמינו להיות חלק מאזור טרנסממברנלי. המודל אומת באמצעות מערך נתונים של חלבונים טרנסממברנליים ידועים והראה תוצאות מבטיחות.

מחקר נוסף של קרוג ואח' (2001) הציג את אלגוריתם TMHMM, המשתמש במודל מרקוב חבוי (HMM) לחיזוי הליקסים טרנסממברנליים ברצפי חלבונים. אלגוריתם TMHMM שימש באופן נרחב ונחשב לאחת השיטות המדויקות ביותר לחיזוי אזורים טרנסממברנליים.

בשנים האחרונות, טכניקות למידה עמוקה יושמו גם לבעיה זו. לדוגמה, מודל DeepTMHMM, שהוצע על ידי הלגרן ואח' (2020), משתמש ברשת נוירונים עמוקה לחיזוי אזורים טרנסממברנליים. המודל אומן על מערך נתונים גדול של חלבונים טרנסממברנליים והשיג ביצועים מתקדמים.

בנוסף למחקרים אלו נוסיף בסקירה זו את מאמר מויקיפדיה[2] אשר מסביר מידע על חלבונים טרנסממברנליים בצורה ברורה, ומביא מקורות רבים על הנושא על מנת לחקור אותו יותר עמוק.

תיאור מאגרי נתונים

המידע הביולוגי ששימש בפרויקט זה הושג ממאגר UniProt, משאב מקיף למידע על רצפי חלבונים ותפקודם. UniProt מספק נתונים באיכות גבוהה, שנערכו ידנית, על רצפי חלבונים, מבניהם ותפקודם. נתוני החלבונים הטרנסממברנליים ששימשו בפרויקט זה הורדו ממאגר UniProt באמצעות ממשק ה-REST API של UniProt.

הנתונים כוללים מידע על רצפי חלבונים, ואזורים טרנסממברנליים. האזורים הטרנסממברנליים מזוהים על בסיס ראיות ניסיוניות וחיזויים חישוביים. הנתונים נאספים באמצעות שיטות ניסיוניות שונות, כולל קריסטלוגרפיה בקרני רנטגן, מיקרוסקופיה קריו-אלקטרונית וספקטרוסקופיית תהודה מגנטית גרעינית (NMR).

[2] Wikipedia contributors. (2023, December 27). Transmembrane protein. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:44, May 16, 2024, from https://en.wikipedia.org/w/index.php?title=Transmembrane_protein&oldid=1191990209

אנליזה/שיטה

איסוף נתונים ועיבוד מקדים

השלב הראשון בפרויקט שלנו היה לאסוף ולעבד את הנתונים. השגנו נתונים על חלבונים טרנסממברנליים ממאגר המידע UniProt, תוך מיקוד בחלבונים טרנסממברנליים הליקליים. הנתונים שהורדו נשמרו בקובץ Excel כדי להימנע מהורדות חוזרות ולהבטיח שחזוריות.

```
def download_transmembrane_protein_data(type):
   url =
f"https://rest.uniprot.org/uniprotkb/stream?fields=accession%2Csequence%2Cft_tr
ansmem&format=xlsx&query=%28%28ft_transmem%3A{type}%29%29+AND+%28reviewed%3Atru
e%29"
   file_path = f"transmembrane_{type}.xlsx"
   if os.path.exists(file_path):
       print("File already exists, skipping download.")
       df = pd.read_excel(file_path)
   else:
       response = requests.get(url, stream=True)

       if response.status_code == 200:
           with open(file_path, "wb") as f:
               for chunk in response.iter_content(1024):
                   f.write(chunk)
           print("File downloaded successfully!")
       else:
           print("Error downloading file:", response.status_code)
           return None
```

חילוץ וניקוי נתונים

לאחר שהורדנו את הנתונים, קראנו אותם לתוך DataFrame של pandas לצורך מניפולציה קלה יותר. לאחר מכן חילצנו את הרצפים הטרנסממברנליים מהנתונים. הליך זה כלל ניתוח של נתוני הרצף וזיהוי האזורים הטרנסממברניים באמצעות ביטויים רגולריים.

```
def extract_transmembrane_sequences(df):
   problematic_indices = []
   transmembrane_sequences = []
   transmembrane_sequences_superlist = []
   for index, row in df.iterrows():
       try:
           sequence = row['Sequence']
           matches = re.findall(r"TRANSMEM\s(\d+)..(\d+)",
row['Transmembrane'])
           transmembrane_indexes = [(int(start), int(end)) for start, end in
matches]
           for t_index in transmembrane_indexes:
               transmembrane_sequence = (sequence[t_index[0] - 1:t_index[1] -
1])
               print(len(transmembrane_sequence))
               transmembrane_sequences.append(transmembrane_sequence)
           transmembrane_sequences_superlist.append(transmembrane_sequences)
           transmembrane_sequences = []
       except Exception as e:
           problematic_indices.append(index)
           print(f"Error processing row {index}: {e}")
       # Drop problematic rows
   df = df.drop(problematic_indices)
   df = df.assign(Transmembrane_sequences=transmembrane_sequences_superlist)
   return df
```

חלוקת נתונים

כדי להבטיח שהמודל שלנו יאושש בצורה הוגנת, חילקנו את הנתונים לסטים של אימון ובדיקה. השתמשנו ב-90% מהנתונים לאימון וב-10% לבדיקה. החלוקה נעשתה באופן אקראי כדי להבטיח שהסטים של האימון והבדיקה יהיו ייצוגים של כלל מערך הנתונים.

```
def separate_testing_data(df):
    testing_dataframe = pd.DataFrame(columns=df.columns)
    testing_indexes = []
    data_length = len(df)
    testing_data_length = int(data_length * 0.1)
    for _ in range(testing_data_length):
        print(_)
        random_num = random.randrange(0, data_length)
        testing_dataframe = pd.concat([testing_dataframe,
df.iloc[[random_num]]])
        df = df.drop(df.index[random_num])
        training_dataframe = df
        data_length -= 1
    testing_dataframe.to_csv('testing_dataframe.csv')
    training_dataframe.to_csv('training_dataframe.csv')
    return testing_dataframe, training_dataframe
```

חישוב ערכי S (Score)

ליבת המודל שלנו מבוססת על חישוב ערכי S לכל חומצה אמינית. ערכי S הם מדד לסבירות שחומצת אמינו מסוימת היא חלק מאזור טרנסממברנלי. ערכים אלו חושבו באמצעות הנוסחה הבאה:

$$^{3}si = ln(\frac{qi}{pi})$$

כאשר pi היא החישוב של כמות הפעמים בה חומצה אמינית i מופיעה סך הכל, חלקי מספר חומצות האמינו בסך הכל. זה נותן לנו סבירות למצוא חומצה אמינית זו, בכל רצפים שיש לנו. qi היא החישוב של חומצה אמינית i אשר נמצאת באזור הטרנסממברנלי, חלקי סכום של כל החומצות האמיניות הטרנסממברנליות.

[3] Jones DT, Taylor WR, Thornton JM. A model recognition approach to the prediction of all-helical membrane protein structure and topology. Biochemistry. 1994 Mar 15;33(10):3038-49. doi: 10.1021/bi00176a037. PMID: 8130217.

כאשר si חיובי, ההסתברות לכך שהחומצה האמינית i תהיה חומצה טנסממברנלית היא גבוהה יחסית לחומצות אמיניות אחרות. כאשר si שלילי, ההסתברות לכך שהחומצה האמינית i תהיה חומצה טרנסממברנלית היא נמוכה בהשוואה לחומצות אמיניות אחרות. כאשר $si \approx 0$ ההסתברות לכך שהחומצה האמינית i תהיה חומצה טרנסממברנלית שווה להסתברות שהיא תהיה גם לא טרנסממברנלית.

```
def calc_p(training_df, amino_acid):
   total_training_sequence = ''
   for index, row in training_df.iterrows():
       total_training_sequence += (row['Sequence'])

   total_occurrence_of_specified_amino_acid = re.findall(amino_acid,
total_training_sequence)

   p = len(total_occurrence_of_specified_amino_acid) /
len(total_training_sequence)
   return p


def calc_q(training_df, amino_acid):
   total_transmembrane_amino_acid_sequence = ''

   for index, row in training_df.iterrows():
       for transmembrane_sequence in row['Transmembrane_sequences']:
           total_transmembrane_amino_acid_sequence += transmembrane_sequence
   total_occurrence_of_specified_transmembrane_amino_acid_occurrence =
re.findall(amino_acid,

total_transmembrane_amino_acid_sequence)

   q = len(total_occurrence_of_specified_transmembrane_amino_acid_occurrence) /
len(
       total_transmembrane_amino_acid_sequence)
   return q


def calc_s(q, p):
   s = math.log(q / p)
   return s
```

אלגוריתם חיזוי

אלגוריתם החיזוי שלנו משתמש בגישת חלון הזזה כדי לסרוק את רצף החלבון ולחשב את סכום ערכי S בתוך כל חלון. החלונות עם הסכומים הגבוהים ביותר נחזים להיות אזורים טרנסממברנליים. יישמנו גם שלב ממוצע כדי להחליק את ערכי S ולשפר את דיוק החיזויים שלנו[4] .

```
def predict_transmembrane_range(sequence, s_values, start_window_size,
end_window_size, averaging_window_size, row):
    window_size = start_window_size
    big_sums = {}
    filtered_ranges = {}
    sequence = list(sequence)
    s_val_to_seq = (pd.Series(sequence)).map(s_values)
    list_of_seq_to_s_val = list(s_val_to_seq)

    averaged_sequence = average_sequence(list_of_seq_to_s_val,
averaging_window_size)

    if len(sequence) <= window_size:
        print("errm, what the sigma? ๏_๏")
        window_size = len(sequence)

    while window_size > end_window_size:
        for i in range(len(sequence) - window_size):
            middle_amino_acid = int(window_size / 2) + i
            start_amino_acid = middle_amino_acid - int(window_size / 2)
            end_amino_acid = middle_amino_acid + int(window_size / 2)

            sum_of_s = sum(averaged_sequence[start_amino_acid:end_amino_acid +
1])
            amino_acid_range = (int(start_amino_acid), int(end_amino_acid))
            big_sums[amino_acid_range] = sum_of_s

        window_size -= 1

    big_sums_sorted = dict(sorted(big_sums.items(), key=lambda item: item[1],
reverse=True))

    selected_ranges = filter_overlapping_ranges(big_sums_sorted)
    for selected_range in selected_ranges:
        filtered_ranges[selected_range] = big_sums_sorted[selected_range]
    filtered_ranges_dataframe = pd.DataFrame.from_dict(filtered_ranges,
orient='index')
```

[4] הגישה שלנו לא טובה, ואנו מניחים כי שימוש ב- convolution אמור לשפר את התוצאות

```
    os.makedirs('filtered_ranges', exist_ok=True)
    filtered_ranges_dataframe.to_csv(f'filtered_ranges/{row["Entry"]}.csv')
    return filtered_ranges
```

הערכה

כדי להעריך את ביצועי המודל שלנו, השתמשנו במדד Jaccard[5] כדי למדוד את הדמיון בין האזורים הטרנסממברניים החזויים והאמיתיים. מדד Jaccard הוא מדד של החיתוך על האיחוד של שני קבוצות, והוא מספק ציון דמיון באחוזים.

```
def evaluate_answer(predicted_range, ground_truth_range):
    intersection_start = max(ground_truth_range[0], predicted_range[0])
    intersection_end = min(ground_truth_range[1], predicted_range[1])
    intersection_length = max(0, intersection_end - intersection_start)

    union_start = min(ground_truth_range[0], predicted_range[0])
    union_end = max(ground_truth_range[1], predicted_range[1])
    union_length = union_end - union_start

    if union_length == 0:
        return 0.0
    iou = intersection_length / union_length

    percentage_similarity = iou * 100
    return percentage_similarity
```

[5] https://en.wikipedia.org//wiki/Jaccard_index

תוצאות התוכנית

סקירה כללית

בפרק זה, אנו מציגים את התוצאות שהתקבלו מהמודל החישובי שלנו לחיזוי אזורים טרנסממברנליים בחלבונים טרנסממברנליים הליקליים. התוצאות כוללות פלטים טיפוסיים מהמודל, ומדדי ביצועים אשר הם חלק מהתוכנית.

דוגמאות לתחזיות

להלן דוגמאות לאזורים הטרנסממברניים החזויים עבור שני רצפי חלבונים ממערך הבדיקה.
התחזיות מוצגות בצורת קבצי CSV, כאשר כל שורה מייצגת אזור טרנסממברנלי חזוי יחד עם הציון המתאים לו[6].
העמודות המסומנות בירוק, מראות את המקטעים שהתוכנית שלנו מחשיבה כקטעים טרנסממברנלים (ציון מעל 2-).

דוגמה 1: חלבון A0A1E1FFN3

ציון	טווח
-00.6274750000000000	(11, 31)
-02.9780250000000000	(432, 452)
-03.357753000000000	(296, 318)
-05.921626000000000	(171, 191)
-06.052358000000000	(371, 391)
-07.164798000000000	(213, 233)
-07.906256000000000	(101, 121)
-08.021678000000000	(331, 351)
-08.279657000000000	(130, 150)
-09.530823000000000	(71, 91)
-09.59031700000000	(274, 294)
-011.1589300000000	(454, 474)
-011.77651200000000	(249, 269)

[6] סכום של ערכי S

(32, 52)	-012.093895000000000
(410, 430)	-012.672254000000000
(192, 212)	-012.712497000000000
(475, 495)	-015.81533800000000

דוגמה 2: חלבון A4G5V6

טווח	ציון
(50, 70)	1.626012000000000
(77, 97)	1.5046820000000000
(172, 192)	0.9380950000000000
(149, 169)	0.7767440000000000
(104, 124)	-01.5298730000000000
(2, 22)	-03.3751960000000000
(25, 45)	-03.920214000000000
(128, 148)	-013.814097000000000

מדדי ביצועים

ביצועי המודל הוערכו באמצעות מדד Jaccard, שמודד את הדמיון בין האזורים הטרנסממברניים החזויים והאמיתיים.

מערך הבדיקה האופטימלי

מערך המשתנים האופטימלי שמצאנו, שאנו ממליצים עליו כברירת מחדל הוא:

```
window_size = 25 #default 25
end_window_size = 20 #default 20
averaging_window_size = 1 #default 1
prediction_score_minimum = -2 #default -2
```

דיוק כללי

המודל השיג דיוק ממוצע של 77.58% כאשר הוערך על מערך הבדיקה. מדד זה מייצג את אחוז הדמיון הממוצע בין האזורים הטרנסממברניים החזויים והאמיתיים מבלי לקחת בחשבון את אזורים טרנסממברנליים אשר הוחמצו על ידי המודל.

אזורים שהוחמצו

המודל גם לקח בחשבון אזורים טרנסממברנליים שהוחמצו, שהם אזורים שהיו קיימים בנתונים האמיתיים אך לא נחזו על ידי המודל. דיוק ממוצע של המודל, במערך הבדיקה האופטימלי, כולל לקיחה בחשבון את אזורים שהוחמצו, היה 72.76% בהרצה שלנו[7].

[7] את קבצי הרצה שלנו העלנו ל-onedrive אשר תוכלו למצוא ב-github

אישוש

בחלק זה אנו נבדוק ונבקר את התוצאות אשר הבאנו בפרק הקודם, ביחס לתוצאות כלי מחקר אקדמאי העוסק בנושא, ומידע מ-Uniprot.

דוגמה 1: חלבון A0A1E1FFN3 :

טווח אמיתי	טווח משוער לפי כלי TMHMM	טווח משוער לפי הכלי שלנו
14- 30	nun	11 - 31

דוגמה 2: חלבון A4G5V6

טווח אמיתי	טווח משוער לפי כלי TMHMM 2.0	טווח משוער לפי הכלי שלנו
50 - 70	49 - 71	50 - 70
78 - 98	78 - 97	77 - 97
105-125	107 - 129	104 - 124
150-170	148 -170	149 - 169
173-193	180 - 202	172 - 192

השוואת אחוזי הצלחה

לאחר הצבת שיעורי הטווחים לחלבונים שבדקנו בתוכנית שלנו בכלי TMHMM 2.0 אלו האחוזים שיצאו:

הבדיקה	הכלי שלנו	TMHMM 2.0
אחוז דיוק כללי	77.5833%	79.45708%
אחוז דיוק כולל איזורים מוחמצים	72.7597%	75.7816%
כמות איזורים מוחמצים (מתוך 38135 איזורים)	2372 6.22%	1765 4.63%

דיון בתוצאות

דיון על האישוש

על פי הנראה באישוש ובתוצאות התוכנית, הכלי שלנו אכן מצליח להגיע לתוצאות עם רמת דיוק מוצלחת יחסית. הגענו לתוצאות הדומות מאוד לכלי המקובל לשימוש בדיקת איזורים טרנסממברנליים בחלבונים טרנסממברנליים (TMHMM 2.0).

גורמים המשפיעים על הביצועים

מספר גורמים יכולים להשפיע על ביצועי המודל, כולל:

1. **איכות הנתונים**: שגיאות או אי-דיוקים במאגר UniProt יכולים להשפיע על תחזיות המודל.

2. **גודל החלון**: בחירת גודל החלון עבור גישת החלון הזזה יכולה להשפיע על דיוק התחזיות.

3. **גודל חלון הממוצע**: גודל חלון הממוצע המשמש להחלקת ערכי S יכול גם להשפיע על התחזיות.

5. **סכום ערכי S מינימלי**: סכום מינימלי של Prediction Score של כל מקטע שיחשב כמקטע טרנס ממברנלי. ככל שהסכום המינימלי יהיה קטן יותר, כך יוחמצו פחות מקטעים טרנסממברנלים, אך דיוק אותם מקטעים יפחת בהתאם.

שיפורים והרחבות

כדי לשפר את הדיוק והחוסן של המודל, ניתן לחקור את השיפורים הבאים:

1. **שילוב תכונות נוספות**: הכללת תכונות נוספות כגון הידרופוביות, מטען ומידע על מבנה משני יכולה לשפר את דיוק המודל.

2. **מודלים מתקדמים של למידת מכונה**: יישום מודלים מתקדמים יותר של למידת מכונה, כגון טכניקות למידה עמוקה, יכול לשפר את התחזיות.

3. **אימות צולב**: שימוש בטכניקות אימות צולב לכיוון הפרמטרים של המודל, כגון גודל החלון וגודל חלון הממוצע, יכול לעזור במציאת ההגדרות האופטימליות.

4. **מערך נתונים גדול יותר**: הרחבת מערך הנתונים לכלול יותר חלבונים טרנסממברנליים יכולה לשפר את החוסן והכלליות של המודל.

5. **שימוש ב-Convolution**: אלגוריתם היכול לשמש כתחליף לאלגוריתם החלון הממוצע שכתבנו. אלגוריתם החלון הממוצע הגיע לתוצאות לא אופטימליות ולכן הפרמטר האופטימלי לגודל החלון הוא 1. אנו חושבים ששימוש באלגוריתם ה-Convolution יוכל לעזור להשגת תוצאות אופטימליות יותר.

סיכום

שיפור ידע בנושאים

- **pandas -** במהלך העבודה השתמשנו רבות בספריית pandas. למרות שהשתמשנו בה כבר בעבר, למדנו לעבוד איתה יותר טוב ושיפרנו את הבנתנו לגבי מנגנוניה.
- **חלבונים טרנסממברנליים -** שיפרנו את הבנתנו לגבי חלבונים טרנסממברנליים. למדנו על המקטעים הטרנסממברנליים ולמדנו על תכונותיהם.
- **שימוש ב- Uniprot REST-API -** המיידע מ-Uniprot הוא בסיס העבודה שלנו, לכן היה חשוב לקבל את המידע דרך קוד פעמים רבות.
- **Jupyter Notebook -** למדנו כיצד לעבוד עם כלי זה כדי להראות את תוכניתנו בצורה נוחה ואפקטיבית.

לסיכום, כפי שניתן לראות בעבודה, הצלחנו להגיע לתוצר שאנחנו גאים בו מאוד, אשר ביצועיו קרובים מאוד לכלים דומים לשלנו. במהלך העבודה התנסנו בספריות שונות והרחבנו את הידע שלנו במגוון רחב של נושאים וחיזקנו את הידע שלנו בנושאים המוכרים לנו.