Identificazione delle liasi della parete cellulare batterica

Christian Francesco Russo 1189343

Scopo del progetto

Lo scopo di questo progetto è quello di implementare attraverso l'utilizzo di Matlab dei nuovi metodi di codifica delle proteine in formato vettoriale numerico a partire dalla loro forma sequenziale, ossia da una rappresentazione della proteina come sequenza di amino-acidi.

I feature vectors ottenuti dai metodi implementati saranno utilizzati per classificare dataset di proteine allo scopo di identificare quali di queste siano liasi della parete cellulare batterica e quali no. Per testare e confrontare le prestazioni dei metodi sviluppati verrà utilizzato il codice di test fornito nel file *MainPhage.m*.

L'obiettivo è quello di provare a sviluppare un metodo di rappresentazione numerica delle proteine che sia più performante degli attuali metodi considerati allo stato dell'arte, per affrontare il caso sotto studio che è quello dell'identificazione delle liasi della parete cellulare batterica che è di fondamentale importanza per la ricerca di un rimedio alle malattie causate da agenti patogeni batterici.

Cenni sulle liasi della parete cellulare batterica

In questi ultimi anni, a causa dell'abuso di antibiotici, i batteri patogeni hanno sviluppato una sempre più crescente resistenza ai farmaci, per questo motivo si sta cercando di sviluppare rimedi più ragionevoli per risolvere il problema. In particolare ci si sta concentrando sullo studio e sullo sviluppo di liasi, in quanto queste sono in grado di distruggere la struttura cellulare del batterio infettivo e di ucciderlo. Per questo motivo le liasi della parete cellulare batterica sono candidati idonei come fonti di antibatterici e per studiarle meglio si vuole sviluppare un metodo computazionale veloce ed efficiente per identificarle.

Rappresentazione delle proteine

L'identificazione e la classificazione delle famiglie di proteine è uno dei principali problemi della bioinformatica e degli studi sulle proteine. Per questo negli anni sono state sviluppate alcune tecniche di sequenziamento che consentono ai ricercatori di identificare le somiglianze biologiche delle famiglie e delle funzioni proteiche. Tuttavia, determinare queste famiglie con applicazioni di sequenziamento richiede un'enorme quantità di tempo. Pertanto, è necessario un sistema di classificazione basato su computer e sull'intelligenza artificiale per risparmiare tempo ed evitare complessità nel processo di classificazione delle proteine.

La struttura primaria di una proteina è la sua composizione amino-acidica, perciò la proteina P viene rappresentata in forma sequenziale come una sequenza di amino-acidi:

$$P = R_1 R_2 R_3 R_4 \dots R_{L-1} R_L$$

dove la composizione di residui di amino-acido $R_1, R_2, \dots, R_L$ e la loro rispettiva posizione $1,2, \dots, L$ all'interno della sequenza identifica in modo univoco la proteina stessa.

Ogni amino-acido della proteina può appartenere a uno dei 20 tipi di amino-acidi noti. Ciascun amino-acido ha delle proprietà chimico-fisiche (in totale circa un centinaio), nella tabella di seguito ne sono riportate solo alcune:

Amino acids	Hydrophobicity	Hydrophilicity	Rigidity	Flexibility	Irreplaceability	Mass	pI	pK(α-COOH)	pK(α-NH_3^+)
A	0.62	−0.5	−1.338	−3.102	0.52	15	6.11	2.35	9.87
C	0.29	−1	−1.511	0.957	1.12	47	5.02	1.71	10.78
D	−0.9	3	−0.204	0.424	0.77	59	2.98	1.88	9.6
E	−0.74	3	−0.365	2.009	0.76	73	3.08	2.19	9.67
F	1.19	−2.5	2.877	−0.466	0.86	91	5.91	2.58	9.24
G	0.48	0	−1.097	−2.746	0.56	1	6.06	2.34	9.6
H	−0.4	−0.5	2.269	−0.223	0.94	82	7.64	1.78	8.97
I	1.38	−1.8	−1.741	0.424	0.65	57	6.04	2.32	9.76
K	−1.5	3	−1.822	3.950	0.81	73	9.47	2.2	8.9
L	1.06	−1.8	−1.741	0.424	0.58	57	6.04	2.36	9.6
M	0.64	−1.3	−1.741	2.484	1.25	75	5.74	2.28	9.21
N	−0.78	0.2	−0.204	0.424	0.79	58	10.76	2.18	9.09
P	0.12	0	1.979	−2.404	0.61	42	6.3	1.99	10.6
Q	−0.85	0.2	−0.365	2.009	0.86	72	5.65	2.17	9.13
R	−2.53	3	1.169	3.060	0.60	101	10.76	2.18	9.09
S	−0.18	0.3	−1.511	0.957	0.64	31	5.68	2.21	9.15
T	−0.05	−0.4	−1.641	−1.339	0.56	45	5.6	2.15	9.12
V	1.08	−1.5	−1.641	−1.339	0.54	43	6.02	2.29	9.74
W	0.81	−3.4	5.913	−1.000	1.82	130	5.88	2.38	9.39
Y	0.26	−2.3	2.714	−0.672	0.98	107	5.63	2.2	9.11

Tabella 1: Alcune proprietà chimico-fisiche degli amino-acidi

Per designare le famiglie di proteine con sistemi assistiti da computer, le sequenze proteiche devono essere convertite nelle rappresentazioni numeriche. Ad oggi sono stati sviluppate diverse metodologie per estrarre feature dalla rappresentazione sequenziale della proteina, nel seguito vediamo quelle principali, prima di andare a vedere i metodi che sono stati sviluppati per questo progetto.

Principali metodi di rappresentazione delle proteine esistenti

Amino acid composition (AS): è il metodo più semplice usato per estrarre features dalla rappresentazione sequenziale della proteina e si basa sul conteggio del numero di amino-acidi di ogni tipo da cui la proteina è composta, normalizzato per la lunghezza della proteina stessa:

$$AS(i) = \frac{h(i)}{N}, \quad i \in [1, \dots, 20]$$

dove $h(i)$ conta il numero di occorrenze dell'i-esimo amino-acido e N è la lunghezza della proteina. Dunque AS rappresenta ogni proteina con un feature vector di lunghezza 20.

2-Gram (2G): questo metodo rappresenta la proteina con un vettore di 20^2 features, ogni feature è ottenuta contando il numero di occorrenze di una data coppia di amino-acidi:

$$2G(k) = \left(\frac{h(i,j)}{N}\right), \quad i,j \in [1, \dots, 20], \quad k = j + 20(i-1)$$

dove $h(i,j)$ conta il numero di occorrenze di una coppia di amino-acidi (i,j) in una proteina di lunghezza N. Dunque 2G rappresenta ogni proteina con un feature vector di lunghezza 400.

Autocovariance approach (AC): questo metodo è più recente e complesso dei precedenti. Dati un parametro m che denota la massima distanza tra due amino-acidi considerati, una proteina $P = (p_1, p_2, \dots, p_n)$ e una proprietà fisicochimica d (proprietà dei singoli amino-acidi), ogni elemento del feature vector $AC^d(i)$ che rappresenta la proteina è definito come segue:

$$AC^d(i) = \begin{cases} \dfrac{h(i)}{N} & i \in [1, \dots, 20] \\ \displaystyle\sum_{k=1}^{N-i+20} \dfrac{(index(p_k, d) - \mu_d) \cdot (index(p_{k+1-20}, d) - \mu_d)}{\sigma_d \cdot (N - i + 20)} & i \in [21, \dots, 20 + m] \end{cases}$$

dove i primi 20 numeri sono la composizione aminoacidica AS vista prima, mentre gli altri m sono definiti da una sommatoria dove: p_k è il k-esimo amino-acido, μ_d e σ_d sono rispettivamente la media e la varianza della data proprietà fisico-chimica di tutti gli aminoacidi e rappresentano i fattori di normalizzazione che servono a normalizzare la somma e $index(i, d)$ ritorna il valore della proprietà fisico-chimica d dell'amino-acido i-esimo. Dunque AC rappresenta ogni proteina con un feature vector di $20 + m$, dove m dipende dal numero di proprietà fisico-chimiche che si vogliono considerare.

Metodi di rappresentazione delle proteine sviluppati

Per questo progetto sono stati sviluppati 10 metodi per rappresentare la proteina in formato vettoriale numerico e sono nel seguito descritti. I primi metodi sono volutamente semplici e banali, questo ci è utile per avere un punto di riferimento e di confronto su cui sviluppare i metodi successivi che sono più complessi e articolati, mentre gli ultimi metodi sono una composizione dei metodi precedenti.

Dato il relativamente basso numero di pattern nel training set a disposizione si è cercato di non creare feature vectors con dimensionalità troppo alta, questo per evitare overfitting sul training set durante l'addestramento, e quindi uno sfalsamento delle misure di prestazioni sul test set.

Di seguito utilizzeremo la notazione $f(i)$ per rappresentare la feature i-esima del feature vector $v = [f(1), f(2), \dots, f(N)]$, di dimensione N.

Metodo 1: questo metodo è il più basilare tra quelli implementati e altro non è altro che una semplificazione del metodo AS prima descritto, in particolare esso è così definito:

$$f(i) = h(i), \quad i \in [1, \dots, 20]$$

dove $h(i)$ conta il numero di occorrenze dell'i-esimo amino-acido.

Metodo 2: questo metodo altro non è che l'implementazione esatta del metodo AS:

$$f(i) = AS(i) = \frac{h(i)}{N}, \quad i \in [1, \dots, 20]$$

Metodo 3: questo metodo è uguale al precedente, ma aggiunge come 21-esima feature la lunghezza $L = N$ della proteina:

$$f(i) = \begin{cases} AS(i) = \dfrac{h(i)}{N} & i \in [1, \dots, 20] \\ L & i = 21 \end{cases}$$

Notiamo che essendo la lunghezza L della proteina diversa per ogni diversa proteina, ci aspettiamo, con l'aggiunta di questa feature, che questo metodo sia in grado di discriminare meglio le diverse proteine tra loro rispetto al metodo precedente.

Metodo 4: questo metodo è abbastanza diverso dai precedenti ed è definito come segue:

$$f(i) = \begin{cases} \dfrac{A(i)}{W} \cdot \displaystyle\sum_{\substack{j=1\dots L \\ tc.\ A(j)=A(i)}} w(j) & i \in [1, \dots, 20] \\ L & i = 21 \end{cases}$$

dove $A(i)$ è l'amino-acido di tipo i-esimo tra quelli noti rappresentato in formato numerico in codifica ASCII, quindi $[A, C, D, \dots, Y] \rightarrow [65{,}67{,}68, \dots, 89]$, questo viene poi moltiplicato per un peso posizionale normalizzato così definito:

$$\frac{1}{W} \cdot \sum_{\substack{j=1\dots L \\ tc.\ A(j)=A(i)}} w(j)\,, \qquad \text{con} \;\; w(j) = \ln(j)\,, \qquad W = \sum_{j=1\dots L} w(j)$$

dove $w(j)$ è il peso posizionale in base logaritmica dell'amino-acido di tipo i nella posizione j-esima all'interno della proteina P, con $j = 1, \dots, L$ indice della sequenza di amino-acidi della proteina $P = R_1R_2R_3 \dots R_j \dots R_{L-1}R_L$ e W è il fattore di normalizzazione dei pesi.

La scelta di un peso posizionale logaritmico anziché esponenziale è dovuta al fatto che l'esponenziale cresce troppo velocemente e non va bene per rappresentare il peso di sequenze come la proteina che hanno qualche centinaio di elementi.

Metodo 5: questo metodo utilizza come features una serie di 16 statistiche di: misura di tendenza centrale, misura di diffusione e misura di forma, calcolate su tutta la sequenza di amino-acidi in formato ASCII di ogni proteina P, ogni feature è così definita (le funzioni di seguito riportate sono quelle della libreria standard MatLab):

$$f(1) = \text{mean}(P), f(2) = \text{median}(P), f(3) = \text{mode}(P), f(4) = \text{trimmean}(P),$$

$$f(5) = \text{geomean}(P), f(6) = \text{harmmean}(P), f(7) = \text{range}(P), f(8) = \text{std}(P),$$

$$f(9) = \text{var}(P), f(10) = \text{mad}(P), f(11) = \text{irq}(P), f(12) = \text{moment0}(P),$$

$$f(13) = \text{moment1}(P), f(14) = \text{moment2}(P), f(15) = \text{swewness}(P), f(16) = \text{kurtosis}(P)$$

Metodo 6: questo metodo utilizza le proprietà chimico-fisiche degli amino-acidi riportate nella *Tabella 1* ed è così definito:

$$f(i) = \frac{1}{W} \cdot \sum_{\substack{j=1\dots L \\ tc.\ A(j)=A(i)}} w(j) \cdot \sum_{k=1\dots n} p(j,k) \qquad i \in [1, \dots, 20]$$

dove come nel metodo 5 $w(j)$ è il peso posizionale in base logaritmica dell'amino-acido di tipo i nella posizione j-esima all'interno della proteina P, con $j = 1, \dots, L$ indice della sequenza di amino-acidi della proteina $P = R_1R_2R_3 \dots R_j \dots R_{L-1}R_L$ e W è il fattore di normalizzazione dei pesi.

Questo peso si va a moltiplicare alla somma di tutte le proprietà k-esime $p(j,k)$ dell'amino-acido di tipo i in posizione j all'interno della proteina e vanno a comporre una delle 20 features da cui il feature vector è composto.

Metodo 7: questo metodo sfrutta il seguente algoritmo per creare un vettore di lunghezza H, dove H è un iperparametro che dopo varie prove è stato fissato ad $H = 30$:

```
function Metodo7(protein)
    H ← 30;
    feature_vector ← zeros(H, 1);
    j ← 1;

    for ogni aminoacid a_i ∈ protein
        aminoacid ← atoi(a_i);
        feature_vector(j) ← feature_vector(j) * 100 + aminoacid; // append protein

        if(mod(j, H) == 0)
```

$j \leftarrow 1$; // carriage return
else
$j \leftarrow j + 1$;

return $feature\_vector$

In pratica questo algoritmo inizia inserendo ordinatamente ogni amino-acido i-esimo della proteina P, in formato ASCII, in posizione j nel feature vector, quando arriva in posizione $j = H$ ritorna alla posizione $j = 1$ e inizia ad impilare ordinatamente i nuovi amino-acidi dietro a quelli precedenti che sono stati inseriti il ciclo precedente. Questo viene fatto finché non si sono consumati tutti gli amino-acidi i-esimi della proteina P. Con questo metodo si cerca di preservare la struttura amino-acidica univoca di ogni proteina P convertendola da una rappresentazione a lunghezza variabile in una rappresentazione a lunghezza fissa H.

Metodo 8: questo metodo è l'unione del metodo 4 e del metodo 5, in particolare esso rappresenta la proteina come un vettore di 37 features, di cui le prime 21 sono quelle del metodo 4 e le ultime 16 sono quelle del metodo 5.

Metodo 9: questo metodo è l'unione del metodo 4 e del metodo 6, in particolare esso rappresenta la proteina come un vettore di 41 features, di cui le prime 21 sono quelle del metodo 4 e le ultime 20 sono quelle del metodo 6.

Metodo 10: questo metodo è l'unione del metodo 6 e del metodo 8, in particolare esso rappresenta la proteina come un vettore di 57 features, di cui le prime 20 sono quelle del metodo 6 e le ultime 37 sono quelle del metodo 8.

Analisi delle prestazioni dei metodi sviluppati

Di seguito è riportata una tabella contenente, per ognuno dei 10 metodi sviluppati: la dimensione del feature vector generato, una stima della complessità temporale impiegata dall'algoritmo di ogni metodo per generare il rispettivo feature vector e l'accuracy che è stata ottenuta come output dal file di test *MainPhage.m* che utilizza un classificatore SVM addestrato sul dataset *Phage.mat* i cui pattern sono rappresentati con i rispettivi metodi implementati. La classificazione consiste nel stabilire quali delle proteine del dataset è liasi e quale no, in particolare le label assumono i seguenti valori:

$$label(i) = \begin{cases} 1 & se\ pattern(i)\ \text{è}\ liasi \\ 2 & altrimenti \end{cases}$$

Tabella delle analisi delle prestazioni dei metodi sviluppati:

Metodo	Size	Complessità	Accuracy (%)
1	20	$O(N)$	81.8182
2	20	$O(N)$	83.4225
3	21	$O(N)$	85.0267
4	21	$O(N)$	85.5615
5	16	$O(nN) \sim (N)$	82.8877
6	20	$O(kN) \sim (N)$	84.4920
7	$H = 30$	$O(N)$	81.8182
8	37	$O(nN) \sim (N)$	86.0963
9	41	$O(kN) \sim (N)$	87.1658
10	57	$O((k+n)N) \sim (N)$	87.1658

dove N è lunghezza della proteina, n è la lunghezza del feature vector e k è numero di proprietà per ogni amino-acido.

Nota: l'analisi della complessità temporale di ogni metodo è approssimativa e non tiene conto del costo delle funzioni fornite da Matlab come length(), find(), sum(), ecc. che nel calcolo vengono considerate tempo costante, altrimenti l'analisi risulterebbe troppo complessa.

Oltre ai test dei diversi metodi sul file di test *MainPhage.m* sono stati effettuati altri test con tre tipi di classificatori diversi: K-NN (con K=5), SVM (con kernel function RBF) e binary tree classification, ciascuno addestrato sullo stesso dataset *Phage.mat*.

Per ogni classificatore addestrato sul dataset *Phage.mat,* i cui pattern sono rappresentati con ciascuno dei 10 metodi, sono state misurate: accuracy, precision e recall:

$$Accuracy = \frac{pattern\ classificati\ correttamente}{pattern\ classificati}$$

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN}$$

questi ci permettono di confrontare la bontà delle diverse rappresentazioni ottenute con i diversi metodi nel nostro caso di studio, dandoci rispettivamente le seguenti informazioni: numero di pattern classificati correttamente su numero di pattern totali, quanto il sistema è accurato e quanto il sistema è selettivo.

Metodo 1			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	84.492	86.9048	95.4248
SVM	82.3529	82.2581	100
Tree search	79.6791	84.0237	92.8105

Metodo 2			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	81.8182	84.3931	95.4248
SVM	85.0267	84.5304	100
Tree search	78.0749	84.1463	90.1961

Metodo 3			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	77.5401	82.0809	92.8105
SVM	83.9572	83.9779	99.3464
Tree search	78.0749	84.1463	90.1961

Metodo 4			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	78.6096	82.659	93.4641
SVM	83.4225	83.8889	98.6928
Tree search	73.262	84.5638	82.3529

Metodo 5			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	79.1444	81.6667	96.0784
SVM	82.3529	82.967	98.6928
Tree search	71.123	82.3529	82.3529

Metodo 6			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	79.6791	84.4311	92.1569
SVM	83.9572	84.3575	98.6928
Tree search	77.5401	85.3503	87.5817

Metodo 7			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	77.5401	82.0809	92.8105
SVM	81.8182	81.8182	100
Tree search	79.1444	84.7561	90.8497

Metodo 8			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	80.7487	83.4286	95.4248
SVM	83.4225	83.5165	99.3464
Tree search	72.7273	84.4595	81.6993

Metodo 9			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	78.6096	82.659	93.4641
SVM	84.492	84.4444	99.3464
Tree search	73.262	82.8025	84.9673

Metodo 10			
Classificatore	Accuracy (%)	Precision (%)	Recall (%)
K-NN	80.7487	83.4286	95.4248
SVM	82.8877	82.8877	82.8877
Tree search	74.3316	83.0189	86.2745

Conclusioni sui risultati ottenuti

Stando a quanto riportato nel documento [2] lo stato dell'arte delle prestazioni medie di classificazione di liasi è stato ottenuto dal server *Lypred* che utilizza classificatori SVM e una tecnica di estrazione delle features basata su ANOVA, in particolare i dati riportati sono: 84.82% di accuratezza (accuracy), 76.47% di sensibilità (precision) e 93.16% di specificità (recall).

I valori di prestazioni che sono stati trovati per i vari metodi sviluppati in questo progetto e che sono riportati nelle tabelle prima viste non sono direttamente confrontabili con quelli del sistema *Lypred*, in quanto calcolati con classificatori e dataset diversi, ma possiamo tuttavia fare un analisi tra i vari metodi sviluppati.

Stando a quanto misurato attraverso i test eseguiti sul file *MainPhage.m* i metodi che hanno ottenuto prestazioni migliori sono il metodo 9 e il metodo 10 che hanno ottenuto un accuracy del 87.1658%, riuscendo a classificare correttamente 163 pattern su 187; il metodo 9 è tuttavia preferibile al metodo 10 in quanto i feature vectors da esso generati hanno dimensionalità più bassa e un costo computazionale (non asintotico) inferiore.

Confrontando invece i test effettuati con gli altri 3 tipi di classificatori le prestazioni migliori sono state ottenute con il metodo 2 che ricordiamo essere l'implementazione del metodo AS con cui abbiamo ottenuto le seguenti prestazioni medie: 81.64% di accuracy, 84,36% di precision e 95.21% di recall.

Bibliografia

[1] An Empirical Study of Different Approaches for Protein Classification, Loris Nanni, Alessandra Lumini and Sheryl Brahnam

[2] Identification of Bacterial Cell Wall Lyases via Pseudo Amino Acid Composition, Xin-Xin Chen, Hua Tang, Wen-Chao Li, Hao Wu, Wei Chen, Hui Ding and Hao Lin

[3] A novel Fibonacci hash method for protein family identi_cation by using recurrent neural networks, Talha Burak ALAKUS, Ibrahim TURKOGLU