Fabrizio Luccio. Informatica per le Biotecnologie

8-2 Altri problemi di carattere biologico

Il problema dei pesi

Nel paragrafo precedente abbiamo adottato i pesi +1, -1, e -1 per match, mismatch e carattere-spazio. Tuttavia, in dipendenza dal problema trattato, questi pesi possono non essere i più significativi e deve essere il biologo a indicare i pesi da scegliere, notando che non sono importanti i loro valori assoluti ma la differenza tra di essi, perché è questa differenza a influenzare il risultato dell'algoritmo di confronto.

Due punti particolarmente rilevanti sono il peso di differenti situazioni di mismatch e il peso associato alla corrispondenza carattere-spazio rispetto al mismatch (che finora abbiamo posto entrambi uguali a -1). Per quanto riguarda i mismatch si deve notare che una sostituzione tra purine (A e G) o tra pirimidine (T e C) è in genere più probabile che una sostituzione tra una purina e una pirimidina e il peso negativo del mismatch dovrebbe essere maggiore in valore assoluto nel secondo caso. Una regola ragionevole per i pesi di match e mismatch è espressa nella seguente tabella che è frequentemente adottata e non dovrebbe richiedere spiegazioni, ma ancora una volta sarà il biologo a stabilire i valori da inserirvi in relazione al problema trattato:

	A	T	C	G
A	+2	-2	-2	-1
T	-2	+2	-1	-2
C	-2	-1	+2	-2
G	-1	-2	-2	+2

Quanto al peso degli spazi la cosa è un po' più complicata non solo per il confronto col peso di mismatch (-1 o -2 secondo la tabella precedente) ma anche per l'eventuale trattamento dei ***gap***, ovvero di sequenze di spazi adiacenti in una sequenza. Per esempio

```
Y:   . . . G T - - - - G A . . .
X:   . . . A T A A T G G T . . .
```

è un gap di lunghezza quattro sulla Y. Scelto il peso da dare a uno spazio rispetto al mismatch, come al solito in dipendenza dal particolare problema biologico (qui poniamo sia -1 e si adotti la tabella sopra) la similarità tra i due tratti di sequenza indicati sarebbe data dalla somma -1 +2 -1 -1 -1 -1 +2 -2 = -3. Tuttavia in

caso di gap spesso si assegna una penalizzazione minore a ciascuno spazio che segue il primo perché la perdita di k caratteri consecutivi è più frequente e meno significativa di k perdite di un solo carattere in posizioni non contigue. Esistono varie funzioni matematiche per trattare i gap, la più comunemente usata detta *costo affine* è espressa come

$$c(k) = - a - (k-1)b$$

ove il costo $c(k)$ di un gap di k caratteri è dato dal costo a del primo carattere più il costo $(k-1)b$ dei caratteri successivi se presenti, con $b < a$ costo di ciascuno di essi. Nell'esempio precedente, ponendo $a = -1$ e $b = -1/4$ avremmo costo del gap $-1\ -3/4 = -1{,}75$ anziché -4.

L'assunzione della nuova tabella di pesi per match-mismatch non altera gli algoritmi visti, salvo l'applicazione dei nuovi valori del peso per il calcolo degli elementi M[i,j] della matrice di programmazione dinamica. L'introduzione del costo affine per i gap introduce invece una sostanziale variazione. Infatti, senza entrare in dettaglio, per calcolare il costo di uno spostamento orizzontale sulla matrice (gap sulla sequenza X) si deve sapere se nella cella di provenienza M[i,j-1] è contenuto un valore determinato da un gap orizzontale già iniziato, e in questo caso si sottrae b, altrimenti si sottrae a. Questo richiede di esaminare ulteriori caselle che precedono M[i,j-1] nella riga. Una situazione perfettamente simmetrica si ha per uno spostamento verticale (gap sulla sequenza Y). Abbiamo citato il problema senza addentrarci ulteriormente su di esso.

Individuazione di un gene in una sequenza di DNA

Un altro problema assai importante riguarda la ricerca di un gene, di cui si è determinata la sequenza S, all'interno di una sequenza già nota T del DNA di un intero organismo. Scopo dell'operazione è per esempio stabilire se la S compare (approssimativamente) come sotto-sequenza U di T, o controllare quali mutazioni presenta la S rispetto alla U. Questo è il problema di pattern comparison già trattato nella dispensa 7, cui ora si applicheranno pesi significativi per il problema biologico con eventuale trattamento dei gap.

Però, se per esempio la S ha una lunghezza |S| di migliaia di basi perché rappresenta il gene di un eucariote che contiene esoni e introni, e la T ha lunghezza |T| di centinaia di milioni di basi perché è la sequenza di un cromosoma umano, l'algoritmo quadratico di programmazione dinamica richiede un numero di operazioni di ordine $|S|\cdot|T|$ che è elevatissimo anche se polinomiale. Il meccanismo di ricerca si può organizzare allora in un modo

differente che spiegheremo senza presentare l'algoritmo relativo che è piuttosto complicato.

Come sappiamo il risultato che S è contenuta in T viene accettato se il numero di errori *e* nell'allineamento ottimo non supera un limite prefissato, poniamo *e* = 49 errori in totale. Se un tale allineamento esiste, dividendo S in *e* + 1 = 50 tratti consecutivi di |S|/50 basi ciascuno, almeno uno di questi tratti deve essere privo di errori: si noti che ne esiste uno solo nel caso in cui gli errori siano ugualmente distanziati, altrimenti possono esistere molti tratti senza errori.

Nel ricco mondo degli algoritmi classici su sequenze ne esiste uno detto **KMP** (dalle iniziali di Knuth, Morris e Pratt che lo hanno proposto) che permette di determinare la presenza **esatta** (cioè senza errori) di un pattern S in un testo T in tempo di ordine |T|+|S|, lineare nella dimensione dell'input. Per tornare al nostro esempio questo ordine è |T|, perché |T| è molto maggiore di |S|. Affrontiamo dunque il problema applicando 50 volte l'algoritmo KMP per ricercare senza errori in T ognuno dei tratti di S. Due risultati sono possibili:

1) nessuno di questi tratti appare in T, dunque $ non appare in T senza superare il numero massimo di errori richiesto;

2) uno di questi tratti, diciamo W, appare in T: dunque la zona Z di T dove dovrebbe trovarsi la S si sviluppa attorno a W e ha lunghezza di ordine |S| (può essere maggiore per la presenza di basi in T che non appartengono a S e danno luogo a spazi in S nell'allineamento). Si confrontano quindi S e Z con l'algoritmo di programmazione dinamica.

Il tempo richiesto dal caso 1) è di ordine *e*|T| per la ricerca degli *e*+1 tratti in T. Il tempo richiesto dal caso 2) è anch'esso di ordine *e*|T| per la stessa ricerca (anche se questa si ferma appena si trova W) più il tempo $|S|^2$ richiesto dall'algoritmo di programmazione dinamica applicato a due sequenze lunghe circa |S|; complessivamente il tempo rimane di ordine *e*|T| se $|T|>|S|^2$ come nel nostro esempio. L'applicazione diretta dell'algoritmo di programmazione dinamica a S e T avrebbe invece richiesto un tempo |S|•|T| >> *e*|T| (tipicamente 100 volte superiore nel nostro esempio).

Determinazione della sequenza di DNA di un organismo

La tecnica di confronto approssimato attraverso la ricerca esatta di un pattern in un testo è anche utilizzata, con metodo praticamente identico a quanto appena visto, quando si tratta di sequenziare l'intero DNA di un organismo di cui già si conosce il

genoma di riferimento memorizzato in una base di dati: per esempio sequenziare il genoma **G** di un uomo particolare, che avrà circa 1 milione di basi sconcordanti con quelli del genoma umano **R** di riferimento sequenziato a suo tempo (questo è l'ordine di grandezza medio del numero di differenze: 1 Mega su un totale di oltre 3 Giga basi, cioè circa 0.3 millesimi).

Partendo da un insieme di frammenti di G ottenuti come per il suo sequenziamento, anziché assemblarli come visto sopra si cercano questi frammenti in R col metodo detto sopra per stabilire in che zona di G si dovranno collocare e quindi con quali altri frammenti collegarli. Il procedimento è naturalmente molto più complicato di così, ma l'idea permette di capire come sequenziare un genoma ex novo con questo metodo richieda un numero di operazioni estremamente inferiore a quelle richieste da un completo fragment assembly.

Qualche considerazione conclusiva.

Quanto discusso sopra ci permette di fare una considerazione generale sulle proprietà degli algoritmi, a commento di quanto esposto in tutte queste dispense. L'apparizione esatta di un pattern in un testo può essere determinata sia con l'algoritmo KMP, sia con quello di programmazione dinamica per l'apparizione approssimata: infatti in questo caso le apparizioni esatte del pattern corrispondono alle celle della matrice che riportano un errore zero. Il primo algoritmo è molto più complicato ma molto più efficiente del secondo, e mostra che il problema è (in gergo) *molto facile* perché ha complessità lineare, che è anche un limite inferiore in quanto è proporzionale al tempo necessario a esaminare tutti i dati (quindi KMP è un algoritmo *ottimo*). Nel gergo degli algoritmi la *facilità* di un problema è legata alla sua natura che gli permette di risolverlo in modo efficiente, non alla facilità con cui si scopre e si progetta un algoritmo di soluzione.

Esistono infine molti problemi su sequenze biologiche che appartengono alla classe dei problemi **NP-completi**, per la cui soluzione si deve inevitabilmente ricorrere a algoritmi approssimati. Questi studi non trovano spazio in queste note: per essi rimandiamo a qualsiasi buon teso di Biologia Computazionale.