Relazione progetto BOLD

Autori:

Valerio Cestarelli
Andrea Iuliano

Tutor:
Emanuel Weitschek

Introduzione

Lo scopo del lavoro è effettuare un'analisi su campioni di sottosequenze di DNA, detti **barcode**, con il fine di classificare e predire la specie di appartenenza a partire da una specifica sequenza genomica. In particolare, è stata sperimentata l'analisi basata sul concetto di multi-locus, paragonandone i risultati con le ricerche mono-locus già sperimentate in passato [1].
Il lavoro è stato basato sui dati presenti nel database BOLD. Nello specifico, sono state prese in considerazione le specie relative alla famiglia delle piante e dei funghi.

Descrizione del progetto

Il progetto consiste nella realizzazione di un software che sia in grado di prelevare i diversi file contenenti le sequenze genomiche, generando per ognuno di essi una matrice, tale da rappresentare con quale frequenza le finestre di basi azotate compaiano nella sequenza relativa ad una specifica famiglia. Una finestra è una sequenza di basi azotate di lunghezza prefissata, che compare all'interno della stringa rappresentante la sequenza genomica. Ad esempio "TTGA"
nella stringa "TTTGATCAGGGGG" compare una volta, mentre la finestra "GGGG" compare due volte: la prima nella posizione (8-11) "TTTGATCA**GGGG**G", la seconda nella posizione (9-12) "TTTGATCAG**GGGG**", considerando l'indice $\in [0,12]$.
Queste "finestre", chiamate k-meri possono essere di una lunghezza predeterminata. Alcuni studi hanno dimostrato che k=4 restituisce dei buoni risultati, in particolare i risultati migliori nel range [3,6] [2].
L'utilizzo dei K-meri è adoperato sulla base di una tecnica "alignment-free", ovvero una tecnica che non utilizza l'allineamento delle sequenze geniche.
Vengono, invece, conteggiati k-meri e poi vengono calcolate le frequenze di questi rispetto al numero totale di finestre trovate nella sequenza genica [2].
Il formato restituito dal database BOLD è il *fasta*, che codifica le informazioni su coppie di righe: la prima presenta le informazioni specifiche del campione in questione (*id* | *specie di appartenenza* | *tipo di bar code*[1]), la seconda indica la sequenza di basi azotate presa in considerazione. In particolare BOLD ritorna un file *fasta* per ogni famiglia. È quindi possibile che diversi campioni compaiano su righe diverse del medesimo file.
Per la memorizzazione della matrice è stato scelto il comune formato CSV, memorizzando la matrice sopra citata. La struttura è così definita:

- la prima colonna identifica l'id del campione

[1] Il tipo di barcode indica specifiche regioni geniche (rbcL e matK per le piante e ITS per i funghi).

- l'ultima colonna indica la specie di appartenenza
- la *i-esima* colonna indica la particolare finestra[2] di geni presa in considerazione
- ogni riga rappresenta le informazioni relative ad un particolare soggetto (id, frequenza della *i-esima* finestra, specie)

La *Tabella 1* riporta un esempio di matrice generata.

Tabella 1

Matrice in formato csv, rappresentante la sequenza genomica in termini di frequenza di finestre di basi azotate.

id	AAAA	AAAC	...	TGCA	TTAA	species
ARG043-08	276.79	158.17	...	158.16	79.08	Mycena_pura
ARG060-08	355.87	276.79	...	276.79	79.08	Inocybe_sp.
BASC001-12	1169.59	888.89	...	397.66	678.36	Basidioascus_magus
...	...	...	...	...	...	...

Le colonne indicano la frequenza della finestra presa in considerazione. Tali frequenze sono state moltiplicate per 10^5 in modo da ottenere dei numeri più leggibili.

Le informazioni relative ai dati presi in considerazione, sia di input che di output, sono mostrati nella *Tabelle 2* e *3*. È da sottolineare che il numero di matrici prodotte è pari al numero di famiglie prese in considerazione e che il numero di righe di ogni matrice è pari al numero di id per famiglia.

Tabella 2

Tabella contenente statistiche di base relative ai dati in input utilizzati per le famiglie di piante.

Piante	# Campioni	MB Input	MB Output
Bryophyta	2535	2.9	5.1
Chlorophyta	4640	4.7	9.5
Lycopodiophyta	441	0.5	0.9
Magnoliophyta	176159	170.2	305.4
Pinophyta	5292	6.0	9.2
Pteridophyta	6503	8.0	15.2

[2] L'analisi è stata effettuata su finestre, ossia k-grammi, di 4 basi azotate, ma il sistema è stato reso parametrico.

Rhodophyta	24104	22.3	46.3

Tabella 3

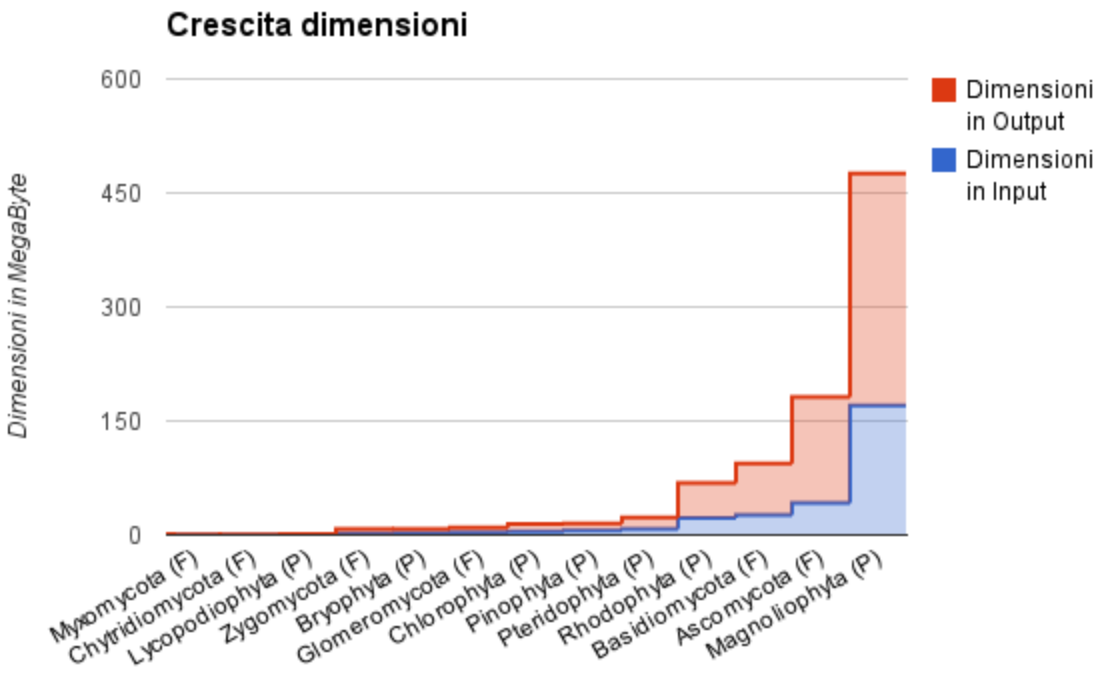
Tabella contenente statistiche di base relative ai dati in input utilizzati per le famiglie di funghi.

Funghi	# Campioni	MB Input	MB Output
Ascomycota	61712	42.2	139.7
Basidiomycota	30463	26.1	68.0
Chytridiomycota	231	0.2	0.5
Glomeromycota	3043	3.0	7.0
Myxomycota	232	0.15	0.5
Zygomycota	2591	1.8	5.8

Viene invece mostrato, in *Figura 1*, come la crescita di dati in input sia equivalente ad una crescita dei dati in output e come questi dati siano mediamente 2.3 volte più grandi dei dati in input.

Figura 1

Crescita delle dimensioni dell'output in funzione della specie di input.



Descrizione dei passi prefissati

Per la realizzazione del progetto sono stati identificati 5 diversi step, risolti con strumenti e tecnologie differenti a seconda dei sottoproblemi individuati. L'approccio scelto in fase di progettazione è stato quello modulare: sono stati quindi sviluppati differenti moduli/script per la realizzazione di un singolo sottoproblema, coordinati tra loro attraverso l'uso di uno script esterno.
Il tutto è riassumibile nelle seguenti fasi:

- Pre-processamento: da ogni famiglia vengono generati tanti file quanti sono i campioni;
- Processamento:
    - vengono estratte e contate le occorrenze delle sottostringhe per ogni campione;
    - preparate le righe parziali per la matrice della famiglia;
- Post-processamento: le singole righe parziali vengono inserite all'interno della matrice completa.

Di seguito, un *FlowChart* che descrive i passi effettuati:

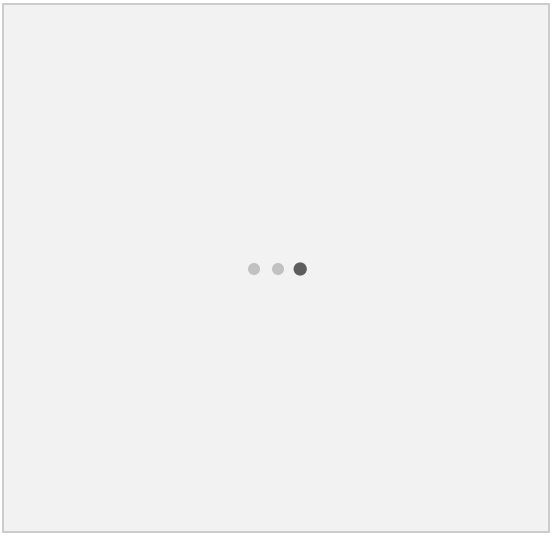


Figura 2. Flowchart

Fase 1: Preprocessamento

Il primo problema è dovuto dalla scelta di chi gestisce il database di BOLD, in quanto i file prelevati sono organizzati per famiglie e racchiudono quindi campioni diversi. Per i nostri scopi è invece necessario che il singolo file contenga esclusivamente le informazioni relative ad un singolo campione. Questo poiché Jellyfish non è in grado di distinguere i diversi campioni inseriti nello stesso file. È stato quindi realizzato uno script in Bash che, per ogni file in ingresso, genera un file per ogni campione in esso contenuto. Per far ciò è stato tenuto conto dei seguenti problemi:

- le informazioni sono memorizzate a coppie di righe
- uno stesso campione potrebbe occorrere più volte e non consecutivamente

Per tener traccia di ogni famiglia, lo split dei file è generato all'interno di una cartella recante il nome della famiglia stessa. Tale operazione sarà utile per la generazione della matrice.

Fase 2: Processamento

Come accennato all'inizio del capitolo, la fase di processamento è stata suddivisa in due parti in fase di lavorazione e sarà pertanto argomentata nei seguenti due sottocapitoli.

Fase 2.1: Jellyfish

La prima fase di processamento consiste nell'individuare quali sono le frequenze delle sottostringhe all'interno della sequenza genomica, contando contestualmente la loro occorrenza all'interno della sequenza stessa. Questo lavoro è stato effettuato tramite l'uso di un software apposito, denominato Jellyfish, il quale attua entrambi i task sopra definiti. Sono stati quindi preparati gli input per il programma all'interno della fase di preprocessamento, avviandolo successivamente per ognuno di essi.
L'uso di un software di terze parti non ci ha permesso di controllare l'output in uscita, che riporta la coppia (*finestra,occorrenza)*, pertanto siamo dovuti ricorrere a differenti strategie per non perdere informazioni utili, quali nome della specie e la lunghezza della stringa. Queste informazioni, insieme alla dimensione della finestra genomica presa in considerazione, sono state codificate all'interno del nome del file seguendo la seguente sintassi:
"*campione_id[parametro1=valore1][parametro2=valore2]....[parametroN=valoreN].occ*"

Fase 2.2: Map-Reduce

Per realizzare le righe della matrice è stato utilizzato Map-Reduce. In input vengono presi i file generati dal passo precedente, in particolare viene passata una cartella (rappresentante la famiglia) contentente i vari file .occ ed in output viene ritornata la riga parziale della matrice, contenente le seugenti informazioni:

- id campione
- specie di appartenenza
- coppie *(finestra, frequenza)*

Per la lavorazione è stato creato un singolo job così composto:

- Mapper: prende in input una riga composta dalla coppia *(finestra, occorrenza)* e ritorna una coppia *(id_campione specie, finestra frequenza)*, ove le informazioni relative all'id del campione e la specie sono prese dal nome del file, oltre alle informazioni relative al calcolo della frequenza. Inoltre, per questioni di leggibilità, i valori di frequenza sono stati moltiplicati per un fattore 10^5
- Reducer: si limita a convertire la lista di *"finestra frequenza"* in una stringa, riportandola come coppia insieme alla chiave "*id_campione specie"*.

Fase 3: Post-processamento

Terminata la fase di processamento si hanno in input le righe parziali della matrice, ossia ogni riga contiene esclusivamente le finestre che occorrono all'interno della sequenza genomica associata al campione di riferimento. Tramite l'uso di uno script in python è stata prima raccolta la lista di tutte le finestre relative ad una famiglia e successivamente sono state riempite tutte le righe con le finestre mancanti, memorizzando il tutto all'interno di un file csv.

Prima di avviare lo script in python, è stato realizzato un piccolo script in bash, che raccoglie i dati prodotti da Map-Reduce e li unisce all'interno di un unico file.

Per descrivere in dettaglio il codice sviluppato è stato allegato un **readme** a questo documento.

Problemi riscontrati

Durante il progetto è stata posta l'attenzione sui seguenti punti:

- la grande quantità di dati
- la numerosità dei file
- le capacità di calcolo richieste dal reducer, durante la fase map-reduce
- grandi capacità di calcolo richieste anche dalle operazioni di pre e post processamento

L'operazione di preprocessamento, che consiste nello splitting dei file di input, ha generato un elevato numero di file da elaborare. Durante la fase di map-reduce, il reducer ha dovuto operare su una quantità di coppie chiave-valore, esponenziale rispetto ai dati di input.
Le operazioni di preprocessamento e postprocessamento hanno richiesto un calcolatore con capacità di elaborazione molto alte. Ci è stato fornito, a tal proposito, un calcolatore del CNR dotato di un processore i7 e 24GB di RAM.
Data la mole di lavoro sottoposta a map-reduce si è reso necessario l'utilizzo di Amazon Web Services. Sono state lanciate tante istanze quante sono le famiglie[3], dotandole di cluster con calcolatori di capacità molto elevate (m2.4xLarge). Ogni istanza è stata lanciata con un numero di calcolatori proporzionato alla grandezza dell'input.

[3] Le due istanze più piccole sono state lanciate in locale in fase di sviluppo.

Tabella 3
Tempi di esecuzione

Famiglia	# calcolatori	Tempo Impiegato	Ore Normalizzate (Amazon)
Ascomycota	9	1h:04m	234
Basidiomycota	5	1h:02m	130
Chlorophyta	3	22m	78
Chytridiomycota	2	8m	52
Glomeromycota	3	18m	78
Lycopodiophyta	2	9m	52
Magnoliophyta	15	2h:28m	1170
Pinophyta	3	21m	78
Pteridophyta	4	22m	104
Rhodophyta	10	24m	260
Zygomycota	2	26m	52
Bryophyta	1(locale)	25m	--
Myxomycota	1(locale)	15m	--

Analisi

Sono state effettuate delle analisi di classificazione tramite Weka su tutte le matrici ottenute.
Weka, acronimo di "Waikato Environment for Knowledge Analysis", è un software Open Source per l'apprendimento automatico, sviluppato nell'università di Waikato in Nuova Zelanda [3].
Weka permette di prendere dati da file o database, filtrarli, clusterizzarli, di applicare ad essi tecniche di classificazione e predizione, e tecniche di apprendimento automatico [3].
Le istanze di Weka vengono lanciate tramite un software scritto in Java. Le matrici sono state precedentemente filtrate da un altro software, scritto nel medesimo linguaggio. Quest'ultimo restituisce la matrice eliminando quelle righe che hanno una specie che compare, all'interno della matrice stessa, un numero di volte al di sotto di una soglia N fissata.
In questo modo si cercano di eliminare quelle specie con poche istanze che "sporcano" il dataset.

L'analisi è stata effettuata con un sistema di predizione basato su regole, ovvero JRIP.
I risultati ottenuti (in allegato) tramite questa analisi si sono dimostrati molto interessanti.

La classificazione, sulla maggiorparte delle famiglie, ha restituito una buona accuratezza e delle altre metriche di valutazione elevate. Tre famiglie hanno invece restituito dei risultati peggiori (precision sotto al 70%).

Si riporta di seguito una tabella riassuntiva dei risultati ottenuti:

Tabella 4

Risultati delle analisi effettuate

Famiglia	# istanze	# istanze classificate correttamente	# istanze classificate incorrettamente	# regole trovate	Precision	Recall	F-Measure
Ascomycota	39265	30402 (77.4277 %)	8863 (22.5723 %)	1831	0.801	0.774	0.780
Basidiomycota	17954	12835 (71.4882 %)	5119 (28.5118 %)	904	0.748	0.715	0.721
Bryophyta	546	442 (80.9524 %)	104 (19.0476 %)	28	0.817	0.810	0.806
Chlorophyta	1915	1548 (80.8355 %)	367 (19.1645 %)	89	0.811	0.808	0.804
Chytridiomycota	109	107 (98.1651 %)	2 (1.8349 %)	3	0.983	0.982	0.982
Glomeromycota	2741	2212 (80.7005 %)	529 (19.2995 %)	92	0.808	0.807	0.806
Lycopodiophyta	110	95 (86.3636 %)	15 (13.6364 %)	5	0.887	0.864	0.870
Magnoliophyta	58861	37733 (64.1053 %)	21128 (35.8947 %)	3317	0.673	0.641	0.636
Myxomycota (4-filtered)	37	26 (70.2703 %)	11 (29.7297 %)	6	0.729	0.703	0.704
Pinophyta	2076	771 (37.1387 %)	1305 (62.8613 %)	154	0.422	0.371	0.364
Pteridophyta	1618	1146 (70.8282 %)	472 (29.1718 %)	97	0.691	0.708	0.688
Rhodophyta	17240	14342 (83.1903 %)	2898 (16.8097 %)	537	0.832	0.832	0.829
Zygomycota	1702	1292 (75.9107 %)	410 (24.0893 %)	98	0.764	0.759	0.753
Risultati Totali e Medie	**144174** (media: **11090**)	**102951** (71.4 %)	**41223** (28.6 %)	**7161** (media: **551**)	media: **0,7666**	media: **0,7518**	media: **0,7495**

I risultati ottenuti suggeriscono che la strada intrapresa può essere ulteriormente esplorata. In particolare è possibile studiare meglio il dataset e realizzare delle tecniche di preprocessing più efficaci, al fine di ottenere risultati migliori, soprattutto su quelle famiglie che hanno mostrato dei valori di accuratezza bassi.
L'utilizzo di una tecnica di predizione basata su regole si è dimostrata efficace con questa tipologia di dati. Un esempio di modello generato con questa tecnica è il seguente:

REGOLE:
(GTCA >= 1288.244767) and (CCTA <= 166.944908) => species=**Mycena_pura**
(CACC <= 158.165283) and (AAGG <= 197.706603) and (CTGC >= 237.247924) => species=**Inocybe_sp.**
(CAAC >= 2010.050251) and (AATT <= 167.785235) => species=**Basidioascus_magus**

Conclusioni e Sviluppi futuri

I risultati ottenuti sono stati soddisfacenti, dimostrando che l'analisi tramite multilocus è una strada percorribile per future ricerche.
È possibile comunque effettuare analisi più approfondite. Ad esempio, si possono eliminare, nella fase di filtraggio, le classi che hanno meno di 20 o più istanze, invece che solo di 8, come è stato fatto ora. In questo modo si ottengono risultati più precisi, eliminando dai dati, quelli che, essendo pochi, rendono il dataset più difficoltoso da predire. Altre operazioni di preprocessing si possono cercare di realizzare su quelle matrici che hanno restituito valori bassi di precisione, cercando di capire le motivazioni che hanno fatto restituire questi risultati, e migliorando così l'algoritmo generale.

Bibliografia

[1] Emanuel Weitschek, Giulia Fiscon and Giovanni Felici, Supervised DNA Barcodes species classification: analysis, comparisons and results, BioData Mining, 2014
[2] Emanuel Weitschek, Fabio Cunial and Giovanni Felici, Classifying bacterial genomes with compact logic formulas on k -mer frequencies
[3] http://it.wikipedia.org/wiki/Weka