BIOINFORMATICA

MFN0951 - BIOINFORMATICA

A CURA DI CHIARA TUMMINELLI

PROF. BECCUTI, BOTTA, CORDERO

BIOINFORMATICA

a cura di Chiara Tumminelli

BIOINFORMATICA PAG. 1

- COS'È LA BIOINFORMATICA? PAG. 1
- METODO DI SANGER PAG. 1
- PROGETTO GENOMA UMANO PAG. 2
- CICLO DELLA RICERCA PAG. 2
- BIOLOGIA COMPUTAZIONALE PAG. 2
- BIOLOGIA MOLECOLARE PAG. 3
- GENI PAG. 3
- DNA PAG. 3
- RNA PAG. 4
- PROTEINE PAG. 4
- SPLICING PAG. 4
- EPIGENETICA PAG. 5
- COVERAGE PAG. 5
- READS PAG. 5
- ACCURATEZZA PAG. 6
- TECNOLOGIE PAG. 6
- APPLICAZIONI PAG. 6

PATTERN MATCHING PAG. 8

- COS'È IL PATTERN MATCHING? PAG. 8
- METODO NAIF PAG. 8
- PREPROCESSING E RICERCA PAG. 8
- ALGORITMO Z PAG. 8
- ALGORITMO DI BOYER-MOORE PAG. 9
- ALGORITMO DI KNUTH-MORRIS-PRATT PAG. 9

MOTIF FINDING PAG. 10

- COS'E' IL MOTIF FINDING? PAG. 10
- MOTIF LOGO PAG. 10
- STRINGA DI CONSENSO PAG. 10
- CALCOLO DELLO SCORE PAG. 10
- DISTANZA DI HAMMING PAG. 11
- STRINGA MEDIANA PAG. 11
- FUNZIONE DI BYPASS PAG. 12
- ALGORITMO GREEDY PAG. 12
- ENTROPIA PAG. 13
- MOTIF IDENTIFICATION PAG. 13

PARTIAL DIGEST PROBLEM PAG. 14

- COS'È IL PARTIAL DIGEST PROBLEM? PAG. 14
- ALGORITMO BRUTE FORCE PAG. 14
- ALGORITMO BRANCH AND BOUND PAG. 14
- ALGORITMO DI DIGEST PARZIALE PAG. 15

ASSEMBLAGGIO GENOMICO PAG. 16
COS'E' L'ASSEMBLAGGIO GENOMICO? PAG. 16
SINGLE-END SEQUENCING E PAIRED-END SEQUENCING PAG. 16
GRAFO DI SOVRAPPOSIZIONE NEL SINGLE-END SEQUENCING PAG. 17
GRAFO DI DE BRUIJN NEL SINGLE-END SEQUENCING PAG. 17
GRAFO DI SOVRAPPOSIZIONE NEL PAIRED-END SEQUENCING PAG. 18
GRAFO DI DE BRUIJN NEL PAIRED-END SEQUENCING PAG. 18
PROBLEMA DELLA STRINGA UNIVERSALE PAG. 18
TOOL VELVET PAG. 19
PROGETTO SUS-MIRRI.IT PAG. 19

CLUSTERING PAG. 20
COS'È IL CLUSTERING? PAG. 20
BONTÀ DEL CLUSTERING PAG. 20
CLUSTERING PARTIZIONALE PAG. 21
CLUSTERING GERARCHICO PAG. 23
CLUSTERING BASATO SULLA DENSITÀ PAG. 25
CLUSTERING BASATO SU GRIGLIE PAG. 26
CLUSTERING BASATO SU MODELLI PAG. 27
TIPI DI GRAFI PAG. 27
MICROARRAY PAG. 28
VALUTAZIONE DEL CLUSTERING PAG. 28
CO-CLUSTERING PAG. 29
CO-CLUSTERING CON VINCOLI PAG. 30

ALBERI FILOGENETICI PAG. 31
RICOSTRUZIONE DELL'ALBERO EVOLUTIVO PAG. 31
FILOGENESI BASATA SULLA DISTANZA PAG. 31
FILOGENESI BASATA SUI CARATTERI PAG. 34
ALBERO DI COPERTURA MINIMO PAG. 35
MODELLI DI RETI E REGOLAZIONE GENICA PAG. 35

PREDIZIONE STRUTTURALE PROTEICA PAG. 37
COS'È LA PREDIZIONE STRUTTURALE PROTEICA? PAG. 37
PROPENSIONI STRUTTURALI PAG. 37
EVOLUZIONE DELLA PREDIZIONE PAG. 38

ALLINEAMENTO DI SEQUENZE PAG. 39
COS'È' L'ALLINEAMENTO DI SEQUENZE? PAG. 39
LONGEST COMMON SUBSEQUENCE PAG. 39
PROGRAMMAZIONE DINAMICA E BACKTRACKING PAG. 40
GRAFO DI ALLINEAMENTO PAG. 40

RICERCA COMBINATORIA DI PATTERN PAG. 42
PATTERN MATCHING MULTIPLO PAG. 42
PREFIX TRIE PAG. 42
SUFFIX TRIE PAG. 42
TRASFORMATA DI BURROWS-WHEELER PAG. 42

BIOINFORMATICA

COS'È LA BIOINFORMATICA?

La bioinformatica e' una disciplina che combina biologia, informatica e statistica per analizzare e interpretare i dati biologici. Grazie alle nuove tecnologie di sequenziamento (NGS), oggi e' possibile ottenere enormi quantita' di dati genomici e biologici. Tuttavia, la loro complessita' e vastita' rende indispensabile l'uso di strumenti computazionali avanzati per la loro gestione e analisi.

La bioinformatica fornisce software e algoritmi per elaborare questi dati, permettendo di prevedere la funzione biochimica di geni e proteine direttamente dalla loro sequenza, senza dover passare attraverso la sperimentazione tradizionale. Questo approccio consente di accelerare la ricerca e di ottenere informazioni cruciali sulla funzione biologica di molecole e organismi.

Le principali applicazioni sono:

1. Comparazione genomica: confronto tra genomi per identificare somiglianze e differenze evolutive;
2. Analisi filogenetica: studio delle relazioni evolutive tra specie;
3. Annotazione del genoma: identificazione e caratterizzazione di geni e altre sequenze funzionali;
4. Bioinformatica strutturale: analisi delle strutture tridimensionali di proteine per comprenderne la funzione.

Le applicazioni secondarie sono:

1. Biologia dei sistemi: studio delle interazioni tra le componenti biologiche per comprendere il funzionamento complessivo di un organismo;
2. Medicina personalizzata: utilizzo dei dati genomici per sviluppare trattamenti su misura per i pazienti;
3. Drug Design computazionale: identificazione di nuovi target terapeutici e sviluppo di farmaci innovativi;
4. Studio dei microbiomi: analisi delle comunita' microbiche per comprenderle il ruolo nella salute umana e nell'ambiente.

Grazie alla bioinformatica, oggi possiamo sfruttare la potenza dell'informatica per estrarre conoscenze dai dati biologici, accelerando le scoperte scientifiche e le applicazioni in ambito medico e biotecnologico.

METODO DI SANGER

Il metodo di Sanger, sviluppato nel 1977 da Frederick Sanger, e' una tecnica di sequenziamento del DNA basata sulla terminazione della catena. Sanger e' stato l'unico scienziato a ricevere due premi Nobel per la chimica: il primo nel 1958 per aver determinato la sequenza amminoacidica dell'insulina e il secondo nel 1980, insieme a Walter Gilbert e a Paul Berg, per il suo contributo al sequenziamento del DNA.

Il metodo si basa sulla sintesi controllata di un nuovo filamento di DNA utilizzando un filamento stampo. Dopo la denaturazione del DNA, viene aggiunto un primer, che e' un breve oligonucleotide complementare che serve per inizializzare la sintesi del nuovo filamento. Questo primer permette l'inizio della reazione, fornendo un'estremita' libera sulla quale la DNA polimerasi, un enzima che catalizza la sintesi del DNA, puo' aggiungere nuovi nucleotidi. Alla miscela di reazione vengono aggiunti anche i nucleotidi normali (dNTPs) e i nucleotidi modificati (ddNTPs). La sintesi del DNA e' una reazione chimica di polimerizzazione, in cui i nucleotidi vengono legati tra loro per formare un filamento di DNA. I dNTPs (deossinucleotidi trifosfato) sono formati da una base azotata (adenina - A, timina - T, citosina - C o guanina - G), uno zucchero (desossiribosio) e tre gruppi fosfato. Le basi azotate formano legami a idrogeno tra loro in specifici accoppiamenti: adenina si lega a timina tramite due legami a idrogeno, mentre citosina si lega a guanina tramite tre legami a idrogeno. I ddNTPs (dideossinucleotidi trifosfato) sono simili, ma mancano del gruppo ossidrile (-OH) in posizione 3' del desossiribosio, che e' essenziale per la formazione del legame fosfodiesterico con il nucleotide successivo. Senza questo gruppo, la DNA polimerasi non puo' proseguire l'allungamento della catena, causando l'arresto della sintesi. Questo genera frammenti di lunghezza diversa, ciascuno terminato con un nucleotide marcato con un fluoroforo.

I frammenti di DNA generati vengono separati tramite elettroforesi capillare, una tecnica in cui un capillare e' riempito con un gel che funge da matrice per la separazione dei frammenti. Quando una corrente elettrica viene applicata, i frammenti migrano nel capillare in base alla loro lunghezza: i frammenti piu' corti migrano piu' velocemente, mentre quelli piu' lunghi si spostano piu' lentamente. I frammenti, ciascuno contrassegnato da un fluoroforo, vengono rilevati in tempo reale da un sensore ottico che registra il segnale emesso dal marcatore fluorescente. In questo modo, e' possibile determinare la sequenza del DNA analizzando l'ordine dei segnali fluorescenti.

Le principali applicazioni del metodo di Sanger sono:

1. Diagnosi precoce di malattie genetiche (ad esempio, fibrosi cistica o distrofia muscolare di Duchenne);
2. Studio di virus e batteri (ad esempio, HIV o epatite C).

Queste applicazioni sono fondamentali per comprendere le cause genetiche di malattie e per sviluppare terapie mirate e personalizzate.

Il metodo di Sanger presenta alcune limitazioni, tra cui una capacita' di sequenziamento limitata, che rende difficili sequenziale grandi genomi in modo efficiente. Inoltre, il processo ha tempi di esecuzione lunghi e costruisce sequenze relativamente corte (tipicamente fino a circa 800-1000 basi). Questi limiti fanno si che il metodo di Sanger sia meno adatto per il sequenziamento di interi genomi o per applicazioni che richiedono un'analisi rapida e ad alta capacita'.

Nonostante queste limitazioni, il metodo di Sanger viene ancora utilizzato quando e' necessaria la massima accuratezza, per il sequenziamento di brevi tratti di DNA o per la conferma di risultati ottenuti con altri metodi, cioe' e' usato come

gold standard. La sua precisione e affidabilita' lo rendono uno strumento fondamentale in contesti specifici, come la validazione di varianti genetiche o per il sequenziamento di singoli geni.

PROGETTO GENOMA UMANO

Il metodo di Sanger ha avuto un impatto fondamentale nelle prime fasi del Progetto Genoma Umano, iniziato nel 1990 e concluso nel 2003, con l'obiettivo di mappare l'intero patrimonio genetico umano, pari a circa 3 miliardi di coppie di basi. Tuttavia, nel corso del progetto, il sequenziamento di Sanger e' stato progressivamente affiancato da tecniche piu' efficienti per accelerare il lavoro. Un ruolo cruciale e' stato svolto dalla bioinformatica, che ha fornito strumenti informatici avanzati per analizzare e gestire enormi quantita' di dati generati.

Determinante e' stata anche la cooperazione internazionale, con il coinvolgimento di laboratori, scienziati e organizzazioni di tutto il mondo, tra cui il National Institutes of Health (NIH) e il Wellcome Trust. Questo progetto ha rappresentato un modello di collaborazione scientifica globale, un approccio che abbiamo visto ripetersi in situazioni successive, come durante la pandemia di COVID-19.

Dopo il completamento del progetto genoma umano, sono emerse nuove tecnologie di sequenziamento, come le Next Generation Sequencing (NGS), tra cui il sequenziamento massivo parallelo e il sequenziamento a singola molecola. Il sequenziamento massivo parallelo consente di analizzare simultaneamente milioni di frammenti di DNA, mentre il sequenziamento a singola molecola permette di leggere direttamente il DNA senza necessita' di amplificazione, riducendo cosi il rischio di errori.

L'avvento delle NGS ha portato ad una drastica riduzione dei costi: mentre inizialmente sequenziare un intero genoma costava circa 100 miliardi di dollari, oggi il costo e' sceso sotto i 1000$. Questo progresso segue la legge di Moore, secondo cui la potenza computazionale raddoppia ogni due anni, favorendo continui avanzamenti nel campo della genomica e accelerando l'innovazione delle biotecnologie.

Grazie a questa riduzione dei costi, il sequenziamento del DNA e' diventato parte della routine clinica, con sempre piu' persone che vi si sottopongono per scopi diagnostici e terapeutici. In particolare, il sequenziamento genomico viene utilizzato per la diagnosi precoce di malattie genetiche e tumori, per la terapia personalizzata e per lo sviluppo della medicina di precisione. Questo approccio consente di selezionare i trattamenti piu' efficaci in base al profilo genetico del paziente, aumentando le probabilita' di successo della terapia e riducendo gli effetti collaterali.

CICLO DELLA RICERCA

Il ciclo della ricerca in bioinformatica e' un processo iterativo che inizia con l'osservazione, basata sulla conoscenza biologica esistente e sulla revisione della letteratura scientifica. La lettura di articoli e studi precedenti permette di identificare lacune nella conoscenza e di formulare nuove ipotesi, che vengono poi testate attraverso esperimenti. Questi esperimenti possono essere di due tipi: wet e dry. Gli esperimenti wet si svolgono in laboratorio, utilizzando tecniche sperimentali per verificare le ipotesi a livello molecolare, cellulare o organismico; gli esperimenti dry, invece, sono computazionali e sfruttano algoritmi, simulazioni e modelli matematici per generare previsioni sui dati biologici.

Le previsioni ottenute vengono poi confrontate con i risultati sperimentali nella fase di analisi dei dati. Se le previsioni corrispondono ai dati reali, il modello viene validato e si puo' affinare la conoscenza esistente formulando nuove domande, dando cosi inizio ad un nuovo ciclo di ricerca. Se invece i risultati non confermano le ipotesi, si formulano nuove osservazioni e ipotesi, riprendendo comunque il ciclo. Questo processo continuo permette di affinare progressivamente le conoscenze in bioinformatica, migliorando le capacita' predittive e applicative della disciplina.

BIOLOGIA COMPUTAZIONALE

La biologia computazionale e' la disciplina che applica strumenti matematici e informatici allo studio dei sistemi biologici. Si divide in quattro discipline principali:

1. Genomica: studio dei geni e del DNA;
2. Proteomica: studio delle proteine e delle loro funzioni;
3. Genomica strutturale: analisi della struttura e organizzazione del genoma;
4. Bioinformatica: sviluppo di strumenti per l'analisi di dati biologici.

Queste discipline fanno parte della biologia dei sistemi, che studia le interazioni tra i vari componenti (geni, proteine, metaboliti) di una cellula o di un organismo, e rientrano nella biologia molecolare computazionale.

Il campo sfrutta tecnologie come machine learning, intelligenza artificiale, algoritmi, teoria dei grafi, statistica e probabilita', robotica, database e teoria dell'informazione per analizzare dati complessi e migliorare la comprensione dei processi biologici.

Un esempio concreto e' il lavoro del team di Francesca Miccio, che ha applicato tecniche di biologia computazionale per ottimizzare i trattamenti contro il glioblastoma, un tumore cerebrale particolarmente aggressivo. Il team ha scoperto che il trattamento con radiazioni fisse per cinque giorni consecutivi era poco efficace e che un regime con radiazioni differite, non somministrate ogni giorno, risultava piu' vantaggioso. Questo approccio e' stato testato in modelli murini in vivo, dimostrando che la sopravvivenza dei topi raddoppiava, come confermato dalla curva di Kaplan-Meier.

BIOLOGIA MOLECOLARE

La biologia molecolare studia i processi fondamentali della vita a livello delle molecole biologiche, con il dogma centrale come principio guida. Il dogma centrale prevede che il DNA, che contiene informazioni genetiche di un organismo, venga copiato in RNA tramite la trascrizione. L'RNA trasporta queste informazioni dal nucleo al citoplasma, dove avviene la sintesi proteica tramite la traduzione. Le proteine, prodotte a partire dall'RNA, sono essenziali per tutte le funzioni vitali dell'organismo.

Oltre a questo ciclo centrale, la biologia molecolare segue un ciclo continuo che collega il DNA, l'RNA, le proteine e il fenotipo, cioe' l'insieme delle caratteristiche osservabili di un organismo. Il fenotipo e' determinato sia da eventi genetici che dall'ambiente circostante. Successivamente, interviene la selezione naturale, per cui gli individui con tratti fenotipici vantaggiosi in un determinato ambiente hanno maggiori probabilita' di sopravvivere e riprodursi. Questo processo porta all'evoluzione, che comporta il cambiamento delle caratteristiche di una popolazione nel tempo.

Il ciclo si chiude perche' l'evoluzione modifica il DNA stesso attraverso l'accumulo di mutazioni e la ricombinazione genetica. Le mutazioni possono essere casuali (errori di replicazione, mutagenesi) o guidate da pressioni selettive. Se una variazione genetica conferisce un vantaggio in un determinato ambiente, si diffonde nella popolazione attraverso la selezione naturale. Nel tempo, queste modifiche diventano parte stabile del genoma, influenzando i futuri cicli di trascrizione, traduzione e trasformazione del fenotipo.

In questo schema, l'individuo si colloca nel punto in cui il fenotipo viene selezionato dall'ambiente, mentre la popolazione si colloca nella fase dell'evoluzione, in cui i cambiamenti nei tratti si diffondono e si stabilizzano nel tempo. Questo ciclo si ripete continuamente, guidando la diversita' e l'adattamento degli organismi viventi.

GENI

Il gene e' l'unita' funzionale del DNA e rappresenta l'elemento fondamentale dell'ereditarieta'. Ogni gene e' composto da diverse regioni:

1. Regione promotrice: punto di avvio della trascrizione;
2. Regione terminatrice: segnale di fine trascrizione per l'RNA polimerasi;
3. Sequenza trascritta: include sia esoni (regioni codificanti) che introni (regioni non codificanti, rimossi nello splicing);
4. Sequenza codificante: composta dagli esoni, ovvero le parti del gene che verranno tradotte in proteine.
5. Sequenza non codificante: regolano l'espressione igienica, come le sequenze UTR, e comprendono anche gli RNA funzionali (microRNA, tRNA, rRNA).

Le sequenze non codificanti si suddividono in sequenze pseudogeniche, che derivano da geni ancestrali che hanno perso la loro funzione a causa di mutazioni o promotori inattivi, e sequenze intergeniche, situate tra un gene e l'altro, che un tempo erano considerate junk DNA ma oggi sono riconosciute come elementi regolatori. Una parte delle sequenze intergeniche e' costituita da genome-wide repeats, cioe' sequenze di DNA ripetute distribuite in tutto il genoma, che si distinguono in Interspersed repeats, distribuite in modo sparso nel genoma, ad esempio Long Interspersed Nuclear Elements (LINEs), Short Interspersed Nuclear Elements (SINEs), Long Terminal Repeats (LTR) e trasposoni; e Tandem Repeats, cioe' sequenze ripetute una accanto all'altra, ad esempio microsatelliti.

L'espressione genica, infine, e' regolata su due livelli: per quanto riguarda la regolazione temporale, i geni vengono attivati o repressi in momenti specifici, ad esempio durante lo sviluppo embrionale, per garantire la corretta formazione di organi e tessuti; la regolazione spaziale, invece, prevede che i geni vengano espressi solo in determinati tessuti o aree dell'organismo. Un esempio e' il gene bicoide, essenziale per la formazione della testa negli embrioni, che viene espresso solo in una regione specifica.

DNA

Il DNA (acido desossiribonucleico) e' la molecola che contiene le informazioni genetiche. La sua struttura e' formata da due catene di nucleotidi che si avvolgono a spirale per formare una doppia elica antiparallela. Ogni nucleotide e' composto da uno zucchero (desossiribosio), un gruppo fosfato e una base azotata (adenina, timina, citosina e guanina). Lo zucchero e il gruppo fosfato formano il backbone, ovvero lo scheletro della molecola di DNA, mentre le basi azotate si legano tra loro attraverso legami a idrogeno: l'adenina si lega con la timina tramite due legami a idrogeno, mentre la citosina si lega con la guanina tramite tre legami a idrogeno. I legami tra citosina e guanina sono piu' resistenti alla denaturazione termica, quindi richiedono piu' energia per essere separati.

Le catene di DNA sono lette in direzione 5' → 3', dove 5' si riferisce al carbonio 5 del desossiribosio, che si lega ad un gruppo fosfato, mentre il 3' si riferisce al carbonio 3 del desossiribosio, che si lega ad un gruppo ossidrile.

Le mutazioni sono cambiamenti nella sequenza del DNA e possono verificarsi per diversi motivi, come errori durante la replicazione del DNA o a causa di fattori esterni, come radiazioni o sostanze chimiche. I tipi principali di mutazioni includono:

1. Mutazioni puntiformi: sostituzione di una singola base;
2. Inserzioni: aggiunta di basi;
3. Delezioni: perdita di basi;
4. Riorganizzazioni cromosomiche: inversioni, traslocazioni e duplicazioni;
5. Modifiche su larga scala: variazioni su grandi porzioni del genoma, come amplificazioni o perdite di segmenti cromosomici.

Un esempio di mutazione genetica e' quella che riguarda il gene FGFR3 (Fibroblast Growth Factor Receptor 3), che e' responsabile della regolazione della crescita delle ossa. Mutazioni in questo gene sono associate a disturbi dello sviluppo scheletrico, come la sindrome di Crouzon e altre forme di nanismo.

RNA

L'RNA (acido ribonucleico) e' una molecola formata da una catena singola di nucleotidi. Ogni nucleotide e' composto da uno zucchero (ribosio), un gruppo fosfato e una base azotata (adenina, uracile, citosina, guanina). Lo zucchero e il gruppo fosfato formano il backbone, ovvero lo scheletro della molecola di RNA, mentre le basi azotate si legano tra loro attraverso legami a idrogeno: l'adenina si lega all'uracile tramite due legami a idrogeno, mentre la citosina si lega con la guanina tramite tre legami a idrogeno. I legami tra citosina e guanina sono piu' resistenti alla denaturazione termica, quindi richiedono piu' energia per essere separati.

L'RNA e' una molecola temporanea, destinata a svolgere la sua funzione e poi essere degradata. La sua instabilita' e' dovuta alla presenza del ribosio, che ha un gruppo ossidrile (-OH) in posizione 2', e dell'uracile, che rispetto alla timina manca di un gruppo metile, rendendolo piu' suscettibile alla degradazione.

Le sequenze di RNA sono lette in direzione 5'⟶3', dove 5' si riferisce al carbonio 5 del ribosio, che si lega ad un gruppo fosfato, mentre il 3' si riferisce al carbonio 3 del ribosio, che si lega ad un gruppo ossidrile.

Esistono diversi tipi di RNA con funzioni specifiche:

1. mRNA (RNA messaggero): copia temporanea di un gene che serve da modello per la sintesi proteica;
2. tRNA (RNA di trasporto): traduce il linguaggio nucleotidico in linguaggio proteico, trasportando gli amminoacidi ai ribosomi;
3. rRNA (RNA ribosomiale): componente strutturale dei ribosomi, facilita la formazione dei legami peptidici tra gli amminoacidi durante la traduzione.

L'RNA ha diverse funzioni fondamentali, tra cui trasportare le informazioni genetiche dal DNA ai ribosomi per la sintesi proteica, catalizzare reazioni chimiche come fanno alcuni RNA enzimatici, e regolare l'espressione genica controllando quali geni vengono attivati o disattivati in una cellula.

PROTEINE

Le proteine sono molecole fondamentali per tutte le funzioni dell'organismo e sono soggette ad un continuo turnover, cioe' un processo di degradazione e sostituzione. La velocita' di questo processo varia: alcune proteine durano a lungo, mentre altre vengono rapidamente rimosse e rimpiazzate.

Dal punto di vista strutturale, le proteine sono composte da lunghe catene di amminoacidi uniti da legami peptidici. Esistono 20 tipi di amminoacidi, ognuno caratterizzato da un gruppo amminico ($-NH_2$), un gruppo carbossilico (-COOH) e una catena laterale (R), che ne determina le proprieta' chimiche e il comportamento nella struttura della proteina.

Le proteine possono assumere diverse strutture:

1. Primaria: sequenza lineare di amminoacidi;
2. Secondaria: ripiegamenti locali che formano α-eliche (strutture a spirale) e foglietti β (strutture piatte ripiegate su se stesse);
3. Terziaria: combinazione di α-eliche e foglietti β, che porta alla conformazione tridimensionale finale della proteina e ne definisce la funzione biologica.

Il ripiegamento delle proteine non e' casuale, ma segue precise regole biochimiche. Tuttavia, tradurre queste regole in modelli computazionali in grado di prevedere correttamente la struttura tridimensionale di una proteina partendo dalla sua sequenza primaria e' una sfida complessa.

Un grande passo avanti in questo campo e' stato fatto con AlphaFold, una rete neurale sviluppata da DeepMind, che e' una divisione di Google. AlphaFold utilizza algoritmi di machine learning addestrati su enormi set di dati per comprendere come le forze fisiche e chimiche determinano la struttura finale di una proteina. Questo modello ha raggiunto un'accuratezza superiore al 97%, dimostrando una capacita' predittiva straordinaria. Tuttavia, le reti neurali come AlphaFold hanno una scarsa interpretabilita', cioe' non sempre forniscono spiegazioni chiare su come arrivano ad una determinata previsione.

Ogni due anni, i progressi nella predizione della struttura delle proteine vengono valutati nella conferenza CASP (Critical Assessment of Structure Prediction), un punto di riferimento internazionale per confrontare i modelli computazionali. Durante le edizioni di CASP, AlphaFold ha ottenuto risultati eccezionali, confermandosi come uno degli strumenti piu' avanzati nel campo della biologia strutturale.

SPLICING

Lo splicing e' un processo fondamentale che avviene negli eucarioti e consiste nella rimozione degli introni, cioe' delle sequenze non codificanti, del pre-mRNA per ottenere un mRNA maturo pronto per la traduzione in proteina. Questo processo e' necessario perche' il gene eucariotico e' composto da esoni e introni, ma solo gli esoni contengono le informazioni utili per la sintesi proteica.

Nello splicing tradizionale, gli introni vengono rimossi e gli esoni vengono uniti tra loro in un ordine preciso, producendo un mRNA maturo che corrisponde esattamente alla sequenza del gene da cui deriva.

Lo splicing alternativo, invece, e' un meccanismo che permette di combinare gli esoni in modi diversi, generando piu' varianti di mRNA a partire dallo stesso gene. Questo significa che un solo gene puo' produrre diverse proteine con funzioni diverse, aumentando la complessita' del proteoma senza aumentare il numero totale di geni. Lo splicing alternativo e' regolato da proteine specifiche che determinano quali esoni vengono inclusi o esclusi nell'mRNA maturo.

Se gli introni non venissero rimossi, l'mRNA risultante sarebbe difettoso e la sintesi proteica verrebbe compromessa. Durante la traduzione, i ribosomi leggerebbero anche le sequenze non codificanti, il che potrebbe portare a proteine non funzionali o tronche a causa della presenza di codoni di stop prematuri negli introni. Inoltre, l'mRNA non modificato potrebbe non essere riconosciuto correttamente dai ribosomi, essere instabile e degradato rapidamente nel citoplasma o causare malattie genetiche, poiche' le proteine risultanti potrebbero non svolgere la loro funzione correttamente o addirittura essere dannose per la cellula.

EPIGENETICA

L'epigenetica e' lo studio di come l'espressione dei geni possa cambiare senza modificare la sequenza del DNA. Questo significa che, pur avendo lo stesso DNA, le cellule possono attivare o disattivare certi geni in base a segnali interni o esterni.

Questi cambiamenti avvengono attraverso modifiche chimiche al DNA o alle proteine che lo avvolgono, come la metilazione o le modifiche degli istoni. Queste modifiche possono essere influenzate da fattori esterni come alimentazione, ambiente e stress. Ad esempio, lo stress puo' alterare l'epigenetica di un individuo e contribuire allo sviluppo della depressione, influenzando i meccanismi che regolano l'attivazione dei geni.

L'epigenetica ha un ruolo fondamentale in diversi processi biologici: sviluppo e differenziamento cellulare, perche' permette alle cellule di specializzarsi durante lo sviluppo embrionale attivando solo i geni necessari per ogni tipo di cellula; adattamento all'ambiente, perche' le cellule possono regolare i loro geni in modo dinamico per rispondere a stimoli esterni; malattie, ad esempio, in alcuni tipi di cancro, i geni che normalmente bloccano la crescita tumorale possono essere spenti tramite la metilazione, favorendo cosi la proliferazione incontrollata delle cellule maligne.

Un aspetto interessante e' che alcune modifiche epigenetiche possono essere ereditate, cioe' trasmesse alle generazioni successive. Questo significa che l'ambiente e lo stile di vita di un individuo potrebbero influenzare anche la regolazione genica della sua progenie.

COVERAGE

Il coverage e' il numero di volte in cui una specifica regione del DNA viene letta durante il sequenziamento. In altre parole, indica quante volte una particolare sequenza e' stata copiata e analizzata, fornendo un'idea della profondita' e dell'affidabilita' dei dati ottenuti.

Un coverage alto significa che la stessa sequenza e' stata letta molte volte, aumentando l'accuratezza del sequenziamento e riducendo la probabilita' di errori casuali. Questo e' particolarmente importante in applicazioni cliniche e di ricerca, dove e' fondamentale avere dati affidabili per individuare mutazioni genetiche o variazioni tra individui.

Un coverage basso, invece, implica che ogni base del DNA e' stata letta poche volte. Questo puo' portare ad una maggiore possibilita' di errori e a una perdita di informazioni, rendendo difficile identificare con sicurezza le variazioni genetiche. Se il coverage e' troppo basso, alcune mutazioni potrebbero non essere rilevate, compromettendo l'affidabilita' dell'analisi.

Tuttavia, anche un coverage troppo alto puo' avere delle conseguenze negative. Un numero eccessivo di letture aumenta i costi e la complessita' computazionale dell'analisi, senza necessariamente migliorare la qualita' dei risultati. Inoltre, un coverage eccessivo puo' introdurre errori sistematici o artefatti, rendendo piu' difficile distinguere le reali varianti genetiche dal rumore di fondo.

Quindi, il coverage deve essere bilanciato: troppo basso porta a dati poco affidabili, mentre troppo alto puo' essere inefficiente e problematico. Il livello ideale dipende dall'applicazione specifica, ad esempio per il sequenziamento di un genoma umano si usa spesso un coverage intorno a 30x, mentre per analisi piu' dettagliate, come sequenziamento di tumori, si possono superare i 100x.

READS

Le read sono le sequenze di DNA ottenute dal sequenziamento. In pratica, rappresentano i frammenti di DNA che vengono letti dalle macchine di sequenziamento e successivamente assemblati per ricostruire la sequenza completa di un genoma o di una regione specifica.

Le short read sono sequenze di DNA relativamente corte, generalmente tra 50 e 300 basi di lunghezza. Vengono prodotte principalmente dalle tecnologie di Next Generation Sequencing (NGS), come Illumina. Sono molto accurate e permettono di sequenziare il DNA a un costo relativamente basso e con un alto throughput, ma hanno il limite di non riuscire a leggere lunghe regioni continue del genoma, rendendo difficile assemblare aree molto ripetitive o complesse.

Le long read, invece, sono sequenze molto piu' lunghe, che possono arrivare fino a 10.000-100.000 basi o piu'. Sono generate da tecnologie di Third Generation Sequencing (TGS), come PacBio e Oxford Nanopore. Il grande vantaggio delle long read e' la loro capacita' di coprire intere regioni genomiche senza interruzioni, facilitando l'assemblaggio e l'analisi di strutture complesse. Tuttavia, hanno spesso un tasso di errore piu' alto rispetto alle short read.

Esiste anche un approccio ibrido, che combina short read e long read per ottenere il meglio di entrambi i metodi. Le short read forniscono alta accuratezza, mentre le long read aiutano a risolvere regioni complesse del genoma. Questo metodo e' spesso utilizzato nell'assemblaggio de novo, dove si cerca di ricostruire un genoma senza riferimenti preesistenti.

Le read, sia corte che lunghe, possono essere ulteriormente suddivise in segmenti piu' piccoli chiamati k-mer, che sono frammenti di lunghezza fissa (di solito tra 15 e 31 basi). I k-mer sono fondamentali per diversi algoritmi di analisi genomica, come l'assemblaggio del genoma e rilevamento di mutazioni, poiche' permettono di confrontare rapidamente grandi quantita' di dati genomici e identificare somiglianze tra diverse sequenze.

ACCURATEZZA

Il tasso di accuratezza nel sequenziamento del DNA e' un parametro cruciale in bioinformatica, poiche' determina la correttezza con cui una base nucleotidica viene identificata. Un'alta accuratezza significa che gli errori di lettura sono minimi, rendendo i dati affidabili per la ricerca e la diagnostica. Questo permette di identificare con precisione le mutazioni genetiche, garantire un'assemblaggio corretto del genoma e ridurre la necessita' di ripetere il sequenziamento, abbattendo tempi e costi.

Al contrario, una bassa accuratezza porta a errori nella lettura delle basi, compromettendo l'interpretazione dei dati e aumentando il rischio di identificare mutazioni inesistenti o di non rilevare quelle reali. Questo puo' influenzare negativamente sia la ricerca scientifica che le applicazioni cliniche, obbligando a ulteriori controlli e aumentando i costi.

Le tecnologie di sequenziamento cercano di bilanciare accuratezza, velocita' e costi. Le short read, come quelle di Illumina, offrono una lettura molto precisa ma possono avere difficolta' nell'assemblaggio di regioni ripetute, mentre le long read, come quelle di PacBio o Oxford Nanopore, forniscono una visione piu' ampia della sequenza ma con un tasso di errore inizialmente piu' elevato. Per migliorare l'accuratezza si utilizzano strategie come l'approccio ibrido, che combina le due tecnologie, oppure l'aumento della profondita' di sequenziamento, che permette di ridurre l'impatto degli errori, ripetendo la lettura delle stesse regioni piu' volte.

TECNOLOGIE

Le tecnologie di sequenziamento del DNA si dividono principalmente in due categorie: microarray e next generation sequencing (NGS). I microarray si basano sull'ibridazione del DNA con sonde specifiche e permettono di analizzare varianti genetiche note in modo rapido ed economico. Tuttavia, non consentono di identificare nuove mutazioni o di ottenere la sequenza completa di un genoma. Il next generation sequencing (NGS), invece, sfrutta il sequenziamento massivo parallelo per leggere milioni di frammenti di DNA contemporaneamente, offrendo un'analisi molto piu' dettagliata e adatta alla scoperta di nuove varianti.

Tra le tecnologie piu' utilizzate nel campo del NGS, Illumina e' una delle piu' diffuse grazie alla sua elevata accuratezza e al basso costo per base sequenziata. Il suo metodo, basato sul sequencing-by-synthetis, prevede l'incorporazione ciclica di nucleotidi fluorescenti durante la sintesi del DNA, permettendo di ottenere letture brevi ma estremamente precise. Tuttavia, l'uso di short read, che in genere non superano le 300 basi, rende piu' complesso il riassemblaggio di regioni ripetute o altamente strutturate del genoma.

A differenza di Illumina, IonTorrent utilizza il rilevamento di variazioni di pH anziche' la fluorescenza, rendendo il processo di sequenziamento piu' veloce e meno costoso. Anche questa tecnologia produce short read, ma presenta una minore accuratezza, in particolare nella lettura di omopolimeri, ossia sequenze con ripetizioni della stessa base.

Le tecnologie di sequenziamento a long read, come PacBio e Oxford Nanopore, affrontano il problema delle regioni ripetute e delle varianti strutturali. PacBio utilizza la tecnologia SMRT (Single Molecule Real-Time Sequencing), che permette di leggere lunghe sequenze di DNA senza necessita' di amplificazione, superando anche le 10.000 basi per lettura. Questa capacita' e' particolarmente utile per assemblaggi di genomi complessi e per studi epigenetici, poiche' consente di rilevare direttamente le modifiche chimiche del DNA. Inizialmente, il tasso di errore di PacBio era piuttosto elevato, ma con l'introduzione delle HiFi reads l'accuratezza e' migliorata notevolmente. Nonostante i vantaggi, il costo e la complessita' della tecnologia rimangono elevati, rendendola meno accessibile rispetto ad altri metodi.

Oxford Nanopore, invece, utilizza nanopori proteici attraverso cui il DNA viene fatto passare, generando variazioni di corrente elettrica che vengono interpretate per determinare la sequenza. Questa tecnologia e' rivoluzionaria perche' permette di ottenere letture estremamente lunghe, anche superiori a 100.000 basi, e offre il vantaggio del sequenziamento in tempo reale. Inoltre, gli strumenti sono portatili e relativamente economici rispetto ad altre piattaforme di sequenziamento. Tuttavia, presenta un tasso di errore piu' alto rispetto ad altre tecnologie, con una precisione che varia dal 5 al 15%, richiedendo algoritmi di correzione avanzati per ottenere dati affidabili.

La scelta della tecnologia piu' adatta dipende dall'applicazione specifica. Le short read di Illumina e IonTorrent sono ideali per il sequenziamento di RNA e per studi in cui e' necessaria un'elevata accuratezza. D'altra parte, le long read di PacBio e Oxford Nanopore sono fondamentali per il sequenziamento di genomi complessi, l'identificazione di varianti strutturali e la metagenomica. La complessita' tecnologica varia notevolmente: Illumina e PacBio richiedono laboratori ben attrezzati e infrastrutture avanzate, mentre Oxford Nanopore offre una soluzione piu' accessibile, anche se con sfide nell'accuratezza dei dati.

APPLICAZIONI

Negli ultimi decenni, il progresso delle tecnologie di sequenziamento del DNA ha permesso di avviare progetti su larga scala per comprendere la variabilita' genetica umana e il funzionamento del genoma. In questo contesto, iniziative come l'International HapMap Project, ENCODE e 1000 Genomes Project hanno avuto un ruolo cruciale nell'ampliare la conoscenza

sulla struttura e sulla regolazione del genoma, con importanti applicazioni per la medicina e la ricerca scientifica. Questi studi hanno fornito strumenti essenziali per identificare varianti genetiche associate a malattie, mappare gli elementi regolatori del DNA e analizzare la diversita' genetica tra popolazioni, contribuendo cosi allo sviluppo della genetica moderna e della medicina di precisione.

L'International HapMap Project e' stato avviato per creare una mappa delle variazioni genetiche umane, in particolare degli aplotipi, ovvero combinazioni di varianti genetiche ereditate insieme. Questo progetto ha premesso di identificare regioni del genoma associate a malattie complesse e ha facilitato lo sviluppo di geni di associazione genome-wide (GWAS).

L'ENCODE (Encyclopedia of DNA Elements) ha avuto l'obiettivo di identificare tutte le regioni funzionali del genoma umano, andando oltre la semplice sequenza del DNA per studiare elementi regolatori, come promotori e enhancers, e il loro ruolo nell'espressione genica. Grazie a questo progetto, e' emerso che gran parte del DNA precedentemente considerato spazzatura svolge invece funzioni regolatorie essenziali.

Il 1000 Genomes Project ha mirato a catalogare la diversita' genetica umana su scala globale, sequenziando il genoma di individui di diversa popolazione per creare un database di varianti comuni e rare. Questo ha migliorato la nostra comprensione della variazione genetica tra popolazioni e ha fornito dati fondamentali per la ricerca medica e la genetica delle malattie.

PATTERN MATCHING

COS'È IL PATTERN MATCHING?

Il pattern matching e' il processo per trovare tutte le posizioni in cui una stringa piu' corta (pattern) appare all'interno di una stringa piu' lunga (testo).

Questa tecnica ha molte applicazioni pratiche. Nella ricerca di informazioni testuali, viene utilizzato per trovare parole o frasi all'interno dei documenti. Nei cataloghi digitali, aiuta a cercare libri in una biblioteca online. Nei browser Internet permette di individuare parole chiave in una pagina web, mentre nei dizionari online facilita la ricerca di significati e sinonimi. In bioinformatica, il pattern matching e' fondamentale per confrontare sequenze di DNA, identificare cancellazioni e replicazioni, riconoscere sequenze simili che possono causare errori durante la replicazione del DNA e diagnosticare malattie genetiche, individuando specifici pattern di mutazioni nei genomi dei pazienti. Inoltre, il pattern matching viene utilizzato per individuare sequenze ripetute nel DNA, note come repeats. Queste ripetizioni possono essere di diversi tipi, come le tandem repeats, in cui una sequenza si ripete consecutivamente, o le interspersed repeats, distribuite in diverse parti del genoma. L'identificazione di queste ripetizioni e' importante per lo studio delle mutazioni, la comprensione di malattie genetiche e l'analisi evolutiva delle specie.

Quando si esegue il pattern matching, possiamo adottare due approcci principali: il pattern matching esatto e il pattern matching approssimato. Il pattern matching esatto richiede una corrispondenza precisa tra il pattern e il testo, ed e' semplice ed efficiente, ma non tollera variazioni e puo' risultare inefficiente con testi grandi. Il pattern matching approssimato, invece, permette piccole discrepanze tra il pattern e il testo, rendendolo piu' flessibile, ma piu' complesso e lento e i risultati possono essere meno deterministici, con il rischio di falsi positivi o negativi.

METODO NAIF

Il metodo naif per il pattern matching e' un algoritmo semplice che cerca un pattern all'interno di un testo confrontandolo carattere per carattere in tutte le posizioni possibili.

Il suo funzionamento e' diretto: si prende il pattern e si prova ad allinearlo con ogni posizione del testo, confrontando uno a uno i caratteri. Se tutti i caratteri del pattern corrispondono a quelli nel testo in una data posizione, si ha un match. Se c'e' una differenza, si passa alla posizione successiva e si ripete il processo.

Il vantaggio principale e' la sua semplicita' e facilita' di implementazione, poiche' non richiede pre-elaborazioni o strutture dati complesse. Tuttavia, il principale svantaggio e' l'inefficienza, specialmente per testi lunghi, perche' ogni possibile allineamento viene testato separatamente, risultando in molte operazioni ripetitive.

La complessita' nel caso peggiore e' O(nm), dove n e' la lunghezza del testo e m la lunghezza del pattern. Questo significa che, se il testo e' molto lungo e il pattern e' relativamente corto, il metodo puo' diventare lento rispetto ad algoritmi piu' avanzati come Boyer-Moore o Knuth-Morris-Pratt (KMP).

PREPROCESSING E RICERCA

Un metodo di pattern matching che utilizza sia preprocessing che ricerca e' piu' efficiente rispetto al metodo naif, perche' riduce il numero di confronti necessari.

Nella fase di preprocessing, si analizza il pattern prima della ricerca per estrarre informazioni utili, come schemi ripetuti o regole che permettono di saltare i confronti inutili. Questo permette di evitare di ricontrollare caratteri che sappiamo gia' non poter portare ad un match.

Nella fase di ricerca, il pattern viene confrontato con il testo sfruttando le informazioni ottenute nel preprocessing. Questo permette di spostare il pattern in avanti in modo intelligente, senza controllare tutte le posizioni una per una, migliorando così l'efficienza dell'algoritmo.

L'uso del preprocessing rende questi metodi piu' veloci rispetto al naif, soprattutto quando si lavora con testi lunghi o con pattern ripetitivi, riducendo la complessita' della ricerca rispetto ad un confronto diretto carattere per carattere.

ALGORITMO Z

L'algoritmo Z e' un metodo efficiente per il pattern matching che sfrutta una fase di preprocessing per accelerare la ricerca del pattern nel testo. L'idea centrale e' calcolare l'array Z, che contiene, per ogni posizione di una stringa, la lunghezza della sottostringa piu' lunga che coincide con il prefisso iniziale della stringa stessa. Questo permette di individuare rapidamente le aree del testo che potrebbero contenere il pattern senza dover ripetere confronti inutili.

Nella fase di preprocessing, si costruisce l'array Z per il pattern concatenato al testo, separati da un carattere speciale. L'array Z viene calcolato analizzando le Z-box, intervalli in cui sappiamo gia' che esistono corrispondenze con il prefisso del pattern. Grazie a queste informazioni, possiamo determinare in anticipo dove il pattern potrebbe combaciare, evitando confronti ridondanti.

Nella fase di ricerca, l'array Z viene sfruttato per individuare rapidamente le posizioni in cui il pattern appare nel testo. Se in una certa posizione l'array Z ha un valore pari alla lunghezza del pattern, significa che in quella posizione c'e' una corrispondenza esatta. Questo permette di trovare tutte le occorrenze del pattern in tempo lineare, senza dover esaminare ogni carattere del testo singolarmente.

Uno dei principali vantaggi dell'algoritmo Z e' la sua efficienza, poiche' permette di eseguire il pattern in tempo lineare O(n+m), grazie alla combinazione di preprocessing e ricerca ottimizzata. Inoltre, e' semplice da implementare e non

richiede strutture dati complesse. Tuttavia, uno svantaggio e' che il preprocessing, pur essendo veloce, introduce un costo aggiuntivo rispetto a metodi piu' semplici come il naif, il che potrebbe non essere giustificato per testi molto brevi o ricerche singole.

ALGORITMO DI BOYER-MOORE

L'algoritmo di Boyer-Moore e' un metodo efficiente per il pattern matching che accelera la ricerca sfruttando il preprocessing e una scansione ottimizzata del testo. A differenza di altri algoritmi che leggono il pattern da sinistra a destra, Boyer-Moore esegue la scansione da destra a sinistra, permettendo di effettuare grandi salti nel testo quando trova caratteri che non corrispondono.

Nella fase di preprocessing, vengono costruite due tabelle fondamentali: la bad character table e la good suffix table. La regola del bad character memorizza, per ogni carattere del pattern, l'ultima posizione in cui appare. Se durante la ricerca si incontra un carattere nel testo che non corrisponde a quello atteso nel pattern, si puo' spostare il pattern in avanti saltando piu' posizioni invece di procedere carattere per carattere. Un'estensione di questa regola, detta bad character rule estesa, permette di gestire in modo piu' efficiente i caratteri che non appaiono affatto nel pattern, massimizzando lo spostamento. La good suffix rule si basa sul riconoscimento delle porzioni finali del pattern che hanno gia' corrisposto al testo. Se un confronto fallisce, si puo' fare avanzare il pattern fino alla prossima occorrenza di quel suffisso nel pattern stesso, evitando di ricontrollare i caratteri inutilmente. La strong good suffix rule migliora ulteriormente questo principio, promettendo di fare salti ancora piu' grandi, sfruttando le corrispondenze parziali all'interno del pattern.

Nella fase di ricerca, il pattern viene allineato con il testo e confrontato da destra verso sinistra. Se si verifica un mismatch, si utilizzano le tabelle preprocessate per determinare il massimo spostamento possibile, riducendo cosi il numero di confronti. Se invece si trova un match completo, si registra la posizione e si sposta al pattern in base alla regola che garantisce il salto piu' lungo.

Uno dei principali vantaggi di Boyer-Moore e' la sua elevata efficienza, specialmente con testi lunghi e alfabeti grandi, poiche' promette di saltare intere sezioni del testo invece di controllare ogni carattere. Nel caso medio, l'algoritmo ha una complessita' di O(n/m), rendendolo estremamente veloce. Tuttavia, nei casi peggiori, ad esempio con pattern molto ripetitivi o alfabeti piccoli, la complessita' puo' arrivare a O(nm), rendendolo meno efficace rispetto ad altri metodi ottimizzati come Knuth-Morris-Pratt. Un altro svantaggio e' la complessita' del preprocessing, che richiede un'elaborazione aggiuntiva per costruire le tabelle di spostamento, rendendolo meno conveniente per pattern molto brevi o per ricerche una tantum.

ALGORITMO DI KNUTH-MORRIS-PRATT

L'algoritmo di Knut-Morris-Pratt (KMP) e' un metodo efficiente per il pattern matching che evita confronti ridondanti sfruttando un'analisi preliminare del pattern. A differenza dell'algoritmo naif, che ripete i confronti per ogni possibile allineamento, KMP utilizza una strategia intelligente per spostare il pattern senza ricontrollare i caratteri gia' verificati.

Nella fase di preprocessing, viene costruito l'array pi (o LPS, Longest Prefix Suffix), che memorizza, per ogni posizione del pattern, la lunghezza del prefisso piu' lungo che e' anche un suffisso della sottostringa fino a quella posizione. Questo permette di determinare di quanto far avanzare il pattern nel testo in caso di mismatch, senza dover ricominciare da zero.

Nella fase di ricerca, il pattern viene confrontato con il testo da sinistra a destra. Se i caratteri corrispondono, il confronto procede normalmente. Se si verifica un mismatch, invece di tornare indietro nel testo, l'algoritmo utilizza l'array pi per calcolare direttamente il prossimo allineamento utile, riducendo cosi il numero di operazioni necessarie. Questo approccio elimina confronti inutili e garantisce che ogni carattere del testo venga esaminato al massimo una volta.

Uno dei principali vantaggi di KMP e' la sua efficienza garantita, con una complessita' di O(n+m), dove n e' la lunghezza del testo e m la lunghezza del pattern. Questo lo rende particolarmente utile per testi lunghi, perche' assicura che il numero di confronti non cresca in modo esponenziale. Inoltre, il preprocessing e' relativamente semplice e permette di risparmiare tempo durante la ricerca. Tuttavia, uno svantaggio rispetto a Boyer-Moore e' che non sfrutta salti lunghi nel testo, poiche' lavora in modo piu' sistematico, risultando meno efficiente in alcuni casi pratici, specialmente con pattern lunghi e alfabeti grandi.

MOTIF FINDING

COS'È IL MOTIF FINDING?

Il motif finding e' un problema tipico della bioinformatica che consiste nell'individuare sequenze conservate (motif) all'interno di un insieme di sequenze biologiche, come DNA o proteine, senza sapere esattamente quale sia il pattern da cercare. Questi motif spesso rappresentano regioni funzionali importanti, come siti di legame per proteine regolatrici o elementi conservati tra specie diverse. Poiche' i motif biologici possono variare leggermente tra le sequenze, il motif finding non si basa su una corrispondenza esatta, ma utilizza modelli probabilistici o algoritmi statistici per identificare regioni ricorrenti con somiglianze significative.

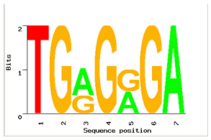
La differenza rispetto al pattern matching sta nel fatto che nel pattern matching si cerca un pattern noto all'interno di un testo o di una sequenza biologica, spesso in modo esatto o con una tolleranza agli errori definita. Nel motif finding, invece, il pattern da trovare non e' noto a priori e deve essere scoperto analizzando le sequenze. Il pattern matching e' un problema deterministico, mentre il motif finding e' spesso un problema non supervisionato, che richiede tecniche di apprendimento automatico o statistico per identificare i pattern nascosti.

MOTIF LOGO

Il motif logo e' una rappresentazione grafica di un motivo biologico, in particolare un motivo di sequenza di DNA, RNA o proteine, che mostra la frequenza e la distribuzione di ogni carattere (nucleotide o amminoacido) in ogni posizione di un set di sequenze allineate. Viene utilizzato per visualizzare i motifs, ossia sequenze ricorrenti che potrebbero avere una funzione biologica, come siti di legame per proteine o altre regioni regolatorie.

Il logo e' costruito con barre che rappresentano ciascuna posizione nel motivo, e la loro altezza riflette la frequenza relativa dei nucleotidi o amminoacidi in quella posizione. Piu' alta e' la barra, piu' comune e' quel carattere in quella posizione. Inoltre, ogni simbolo all'interno del logo puo' essere rappresentato da lettere (per il DNA o RNA) o da abbreviazioni per gli amminoacidi (per le sequenze proteiche). I colori o la larghezza delle barre possono essere utilizzati per rappresentare la variabilita' del carattere in quella posizione o l'energia di legame associata.

Un motif logo fornisce quindi una visualizzazione immediata e intuitiva di come un motivo e' composto e di quanto ogni carattere sia conservato o variabile, il che e' particolarmente utile per interpretare e comprendere le funzioni biologiche di sequenze simili.



STRINGA DI CONSENSO

La stringa di consenso e' una sequenza rappresentativa che viene ottenuta analizzando un insieme di sequenze simili, come nel caso di motivi biologici, dove si cerca di identificare una sequenza comune che sintetizzi le informazioni principali di un gruppo di sequenze. In altre parole, la stringa di consenso e' una sequenza che riflette i caratteri piu' frequenti in ogni posizione nelle sequenze allineate. La sua costruzione e' un passo fondamentale nel motif finding, in quanto aiuta a identificare la struttura comune di una famiglia di sequenze.

Uno dei principali vantaggi della stringa di consenso e' che offre una rappresentazione semplificata delle sequenze, concentrandosi sugli elementi piu' conservati rendendo piu' facile l'interpretazione. In questo modo, aiuta a individuare rapidamente i motivi biologici, come siti di legame o regioni regolatorie. Tuttavia, un suo svantaggio e' che puo' perdere informazioni sulle variazioni tra le sequenze. Questo e' un problema soprattutto quando le sequenze presentano molte divergenze, poiche' la stringa di consenso potrebbe non rappresentare adeguatamente la complessita' delle sequenze originali. Inoltre, non tiene conto delle mutazioni o delle possibili variazioni che potrebbero essere biologicamente rilevanti.

Per quanto riguarda la complessita', la costruzione della stringa di consenso richiede l'analisi di ogni posizione nelle sequenze, e quindi la sua complessita' e' proporzionale al numero di sequenze e alla loro lunghezza. Ad esempio, se si ha un insieme di 10 sequenze di lunghezza 100, l'algoritmo deve esaminare ogni posizione per determinare il carattere piu' frequente, portando ad una complessita' che cresce all'aumentare delle sequenze e della loro lunghezza.

AGCT
ATCT
GGCT
AGGT

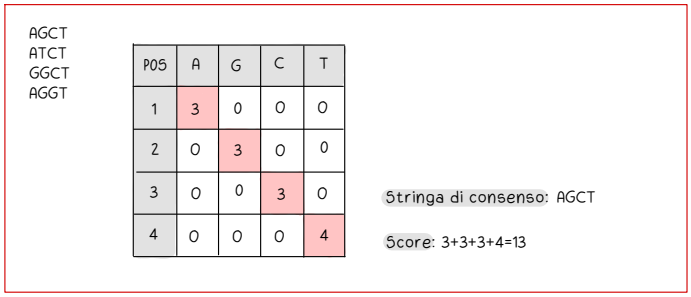
POS	A	G	C	T
1	3	1	0	0
2	0	3	0	1
3	0	1	3	0
4	0	0	0	4

Stringa di consenso: AGCT

CALCOLO DELLO SCORE

Per identificare un motif migliore in un insieme di sequenze, bisogna cercare di distinguere tra un motivo che ha una forte rilevanza biologica (motivo buono) e uno che e' casuale o poco informativo (motivo cattivo). Una delle modalita' per fare questa distinzione e' l'uso di uno score, che fornisce una misura quantitativa di quanto il motivo sia rappresentativo e conservato tra le sequenze. Piu' alto e' lo score, maggiore e' la probabilita' che il motivo trovato sia significativo.

Un tipo comune di score e' quello basato sulla Position Weight Matrix (PWM), che rappresenta la probabilita' di ciascun carattere (nucleotide o amminoacido) in una determinata posizione di un motivo. La PWM viene costruita analizzando le sequenze allineate e calcolando la probabilita' di ciascun carattere (A, T, C, G per il DNA) per ogni posizione del motivo. Le probabilita' vengono poi trasformate in punteggi, che indicano quanto ciascun carattere in una posizione sia rappresentativo del motivo. Piu' le posizioni di un motivo sono conservate e corrispondenti alla PWM, maggiore sara' lo score.



Lo score per una sequenza candidata che corrisponde a questa matrice puo' essere calcolato sommando i valori corrispondenti nelle celle della PWM. Un punteggio alto indica una buona corrispondenza con il motivo trovato.

Un motivo buono avra' una PWM con colonne altamente conservate, cioe' con caratteri che appaiono frequentemente nelle stesse posizioni tra le sequenze. Questo indica che il motivo e' rilevante e probabilmente ha una funzione biologica. Al contrario, un motivo cattivo avra' una PWM con caratteri distribuiti in modo casuale, con punteggi bassi per ogni posizione, indicando che il motivo e' poco conservato e quindi poco significativo.

DISTANZA DI HAMMING

La distanza di Hamming e' una misura utilizzata per quantificare le differenze tra due stringhe di uguale lunghezza, contando il numero di posizioni in cui i caratteri corrispondenti sono diversi. Nel contesto del motif finding, questa distanza puo' essere impiegata per affinare l'identificazione di un motivo, confrontando la stringa di consenso (che rappresenta la sequenza piu' comune in un set di sequenze) con altre possibili sequenze. L'idea e' di trovare le sequenze che si adattano meglio al motivo, ossia quelle che hanno una distanza di Hamming minima dalla stringa di consenso. In altre parole, piu' basso e' il valore della distanza di Hamming tra una sequenza candidata e la stringa di consenso, maggiore e' la somiglianza tra di esse e, quindi, maggiore e' la sua probabilita' di rappresentare correttamente il motivo.

Ad esempio, se la stringa di consenso e' "AGCT" e si desidera confrontarla con una serie di sequenze candidate, come "AGCT", "AGGT", "TGCT", "AGAT" e "CGCT", la distanza di Hamming tra la stringa di consenso e le altre sequenze sarebbe calcolata come segue: tra "AGCT" e "AGCT" la distanza di Hamming e' 0 perche' la stringa e' uguale; tra "AGCT" e "AGGT" la distanza di Hamming e' 1 perche' la quarta posizione differisce; tra "AGCT" e "TGCT" la distanza di Hamming e' 1 perche' la prima posizione differisce; tra "AGCT" e "AGAT" la distanza di Hamming e' 1 perche' la terza posizione differisce; tra "AGCT" e "CGCT" la distanza di Hamming e' 1 perche' la prima posizione differisce. In questo caso, "AGCT" e' la sequenza che si adatta perfettamente al motivo, con una distanza di Hamming pari a 0. Le altre sequenze differiscono nella stringa di consenso in una sola posizione, quindi hanno una distanza di Hamming di 1. Questo esempio mostra che la distanza di Hamming puo' essere utile per confrontare sequenze e determinare quali si adattano meglio a un motivo, ma se tutte le sequenze hanno distanze simili, sara' necessario considerare altri fattori per affinare la selezione, come la frequenza di occorrenza o altre caratteristiche biologiche.

Quando le sequenze hanno lunghezze diverse e si vuole calcolare una misura di similarita' come la distanza di Hamming, un approccio efficace consiste nell'utilizzare prima un allineamento multiplo delle sequenze. Questo processo, noto come Multiple Sequence Alignment (MSA), e' fondamentale per ottenere un confronto significativo tra sequenze di lunghezza variabile. L'allineamento consente di allineare le sequenze in modo che le posizioni corrispondenti possano essere confrontate direttamente, anche se le sequenze hanno lunghezze diverse. Il risultato dell'allineamento e' una rappresentazione delle sequenze in cui le differenze tra le varie posizioni sono piu' evidenti e dove le regioni di gap (differenze di lunghezza tra le sequenze) vengono trattate come vuote o simboli speciali. In questo modo, si ottiene una sequenza di consenso che rappresenta al meglio la posizione di ciascun residuo nelle sequenze, rendendo possibile il calcolo delle distanze tra sequenze allineate in modo coerente.

Nel contesto di sequenze di lunghezze diverse, potrebbe essere utile anche considerare altre misure di similarita', come la distanza di Levenshtein. Questa misura, a differenza della distanza di Hamming, non richiede che le sequenze siano della stessa lunghezza, ma permette di calcolare il numero minimo di operazioni (inserimenti, cancellazioni e sostituzioni) necessarie per trasformare una sequenza nell'altra. Pertanto, se la distanza di Hamming non e' applicabile direttamente in caso di lunghezze differenti, la distanza di Levenshtein puo' rappresentare un'alternativa per ottenere una misura di dissimilarita' tra sequenze.

STRINGA MEDIANA

La stringa mediana e' una sequenza che minimizza la distanza totale da tutte le altre sequenze in un insieme, cercando di rappresentare il motivo centrale che possiede la minima variazione rispetto alle sequenze analizzate. Questo approccio e' utile in contesti come il motif finding, dove si cerca di identificare un motivo che rappresenti sequenze con piccole mutazioni. Per calcolare la stringa mediana, si misurano le distanze tra tutte le sequenze dell'insieme,

solitamente, con la distanza di Hamming. La stringa mediana rappresenta quindi il miglior compromesso, catturando le caratteristiche comuni delle sequenze con la minima variabilita'.

FUNZIONE DI BYPASS

La funzione di bypass e' una tecnica utilizzata in vari ambiti di ricerca, tra cui il motif finding, per migliorare l'efficienza del processo di identificazione di motivi. In particolare, permette di saltare passaggi non rilevanti o esplorare soluzioni alternative in modo rapido, ottimizzando il calcolo complesso. La sua funzione principale e' ridurre il numero di operazioni necessarie per arrivare alla soluzione migliore, risparmiando tempo e risorse computazionali. Ad esempio, durante la ricerca di un motivo, puo' evitare di esaminare sequenze che sicuramente non forniranno un buon punteggio, concentrandosi solo sulle combinazioni piu' promettenti.

Il funzionamento della funzione di bypass si basa sulla capacita' di saltare fasi che altrimenti richiederebbero un esame dettagliato, come l'esclusione di sequenze che non soddisfano determinati criteri preliminari o che hanno gia' un punteggio basso. In questo modo e' possibile accelerare il processo di ricerca e migliorare l'efficienza computazionale.

Per esempio, se stiamo cercando di identificare un motivo in un insieme di sequenze di DNA, una funzione di bypass potrebbe escludere velocemente le sequenze che non mostrano somiglianze evidenti con il motivo cercato, concentrandosi invece su quelle che potrebbero essere piu' promettenti. Questo approccio consente di ottimizzare il tempo di esecuzione del calcolo e ridurre il numero di iterazioni necessarie per trovare una soluzione ottimale.

Uno dei principali vantaggi della funzione di bypass e' l'aumento dell'efficienza. Riducendo il numero di calcoli necessari e saltando passaggi non utili, permette di ottenere risultati in tempi piu' brevi. Questo e' particolarmente utile quando si lavora con grandi dataset o quando la ricerca dei motivi richiede un'analisi intensiva delle sequenze. Inoltre, grazie alla possibilita' di concentrarsi solo su soluzioni promettenti, il processo diventa piu' mirato, evitando sprechi di risorse computazionali. Il principale svantaggio di una funzione di bypass e' che, a volte, potrebbe escludere involontariamente sequenze che avrebbero potuto portare ad un risultato migliore. Poiche' la funzione si concentra su sequenze piu' promettenti, c'e' il rischio di non considerare alcune combinazioni che, pur avendo una probabilita' inferiore di successo, potrebbero comunque contenere motivi rilevanti. Questo potrebbe ridurre la completezza del risultato finale. Inoltre, l'implementazione di una funzione di bypass richiede una buona comprensione del dominio del problema per evitare errori di selezione.

Dal punto di vista della complessita' computazionale, la funzione di bypass puo' ridurre significativamente il tempo di esecuzione di un algoritmo. Tuttavia, dipende molto da come viene implementata e dalle specifiche condizioni in cui viene utilizzata. Se la funzione e' progettata per saltare solo passaggi molto costosi in termini computazionali, la sua efficacia aumenta, portando ad un miglioramento della complessita' temporale. In generale, riducendo il numero di operazioni necessarie, una funzione di bypass puo' trasformare un algoritmo che altrimenti avrebbe una complessita' elevata in uno piu' efficiente. Tuttavia, bisogna fare attenzione a non sacrificare la qualita' del risultato a favore della velocita', quindi la sua applicazione deve essere ben calibrata in base al contesto.

ALGORITMO GREEDY

L'algoritmo Greedy per il motif search problem e' un approccio turistico utilizzato per identificare motivi ricorrenti all'interno di un insieme di sequenze, cercando di ottimizzare localmente la ricerca del miglior motivo, passo dopo passo. La caratteristica fondamentale che le sequenze devono avere in input all'algoritmo e che devono essere simili tra loro; infatti, l'algoritmo cerca di identificare una sequenza che rappresenti il motivo comune che appare frequentemente nelle sequenze, il che implica che ci deve essere una certa omogeneita' nei dati.

L'algoritmo Greedy per il motif search inizia selezionando un segmento di una sequenza come motivo iniziale. A partire da questa scelta iniziale, l'algoritmo intera attraverso le altre sequenze, cercando i segmenti che piu' somigliano al motivo gia' trovato. Ad ogni interazione, l'algoritmo aggiusta il motivo attuale, cercando il segmento che massimizza la corrispondenza con le sequenze rimanenti. Questo processo continua fino a quando non si raggiunge la condizione di arresto, come un numero prefissato di interazioni o il raggiungimento di una buona corrispondenza tra il motivo e le sequenze.

Il passo chiave in un algoritmo Greedy e' la sua scelta locale ottimale: ad ogni passaggio, se seleziona la miglior sequenza che si adatta al motivo corrente, senza considerare globalmente le implicazioni delle scelte fatte in precedenza.sebbene questo approccio possa portare a soluzioni efficienti e veloci, non garantisce che la soluzione finale sia la migliore possibile a livello globale.

Uno dei principali vantaggi dell'algoritmo Greedy e la velocita' e l'efficienza computazionale. Poiche' l'algoritmo non esplora tutte le possibili combinazioni (come altri approcci piu' esaustivi), ma si concentra solo sulle scelte locali migliori, puo' produrre una soluzione in tempi relativamente brevi. Questo e' particolarmente utile quando si lavora con grandi dataset, dove la ricerca dei motivi potrebbe richiedere enormi risorse computazionali. il principale svantaggio, invece, e' che non sempre garantisce la soluzione ottimale globale. Poiche' si concentra solo sulla scelta migliore ad ogni passo senza considerar l'intero spazio delle soluzioni, e' possibile che l'algoritmo finisca per convergere su una soluzione subottimale, che non rappresenta il miglior motivo possibile rispetto a tutte le sequenze. Inoltre, se le sequenze di input sono molto diverse tra loro, l'algoritmo potrebbero non trovare un motivo comune significativo.

La complessita' computazionale dell'algoritmo Greedy dipende principalmente dal numero di sequenze e dalla loro lunghezza. In generale, l'algoritmo esplora ogni sequenza per trovare il migliore allineamento possibile ad ogni iterazione. Pertanto, la complessita' temporale dell'algoritmo e' generalmente proporzionale al numero di sequenze e alla lunghezza di ciascuna, ossia O(mn), dove m e' il numero di sequenze e n e' la lunghezza di ciascuna sequenza. Sebbene piu' veloce

rispetto a metodi esaustivi, come la ricerca di tutte le possibili sotto sequenze, l'algoritmo Greedy non e' sempre il piu' preciso e potrebbe essere soggetto a errori locali che lo portano ad una soluzione meno ottimale.

ENTROPIA

L'entropia nel motif finding e' una misura della variabilita' o dell'incertezza di una posizione in un motivo.un motivo e' una sequenza di simboli (nucleotidi nel caso del DNA o amminoacidi nel caso delle proteine) che appare frequentemente in un insieme di sequenze. L'idea dell'entropia e' che, se una posizione del motivo ha una bassa entropia, significa che un nucleotide (o amminoacido) specifico e' molto conservato in quella posizione. Se invece ha un'alta entropia, significa che ci sono molte varianti per quella posizione, e quindi e' meno conservata.

Nel contesto del motif finding, l'entropia aiuta a identificare le posizioni piu' informative di un motivo. In altre parole, le posizioni con bassa entropia sono quelle dove i nucleotidi o amminoacidi sono piu' simili tra loro nelle sequenze analizzate, il che le rende piu' significative nel determinare il motivo.

La PWM (Position Weight Matrix) e' una rappresentazione numerica di un motivo.ogni colonna della PWM rappresenta una posizione del motivo, mentre le righe rappresentano i vari nucleotidi (A, C, G, T per il DNA). il contenuto di ciascuna cella della matrice e' un punteggio che riflette la frequenza o probabilita' di un determinato nucleotide (o amminoacido) in cui la posizione attraverso tutte le sequenze analizzate. Questi punteggi sono usati per indicare la forza del motivo in una determinata posizione e aiutano a identificareil motivo piu' rappresentativo.

Quindi, l'entropia misura quanto una posizione di un motivo e' conservata o variabile, e la PWM aiuta a rappresentare questa variabilita', con i valori nelle celle che indicano la prevalenza di un nucleotide in ogni posizione del motivo.

MOTIF IDENTIFICATION

I termini "motif finding" e "motif identification" vengono talvolta usati in modo intercambiabile, ma in realta' si riferiscono a concetti differenti.

Il motif finding il processo di ricerca de novo (senza conoscenza preliminare) di sequenze ricorrenti in un insieme di dati, spesso in DNA, RNA o proteine. L'obiettivo e' scoprire i motivi conservati che potrebbero avere una funzione biologica, come siti di legame per fattori di trascrizione.gli algoritmi di motif finding cercano di individuare questi schemi senza informazioni a priori sulla loro struttura o posizione.

Il motif identification, invece, si riferisce al riconoscimento di un motivo noto in nuove sequenze. In questo caso, si parte da un modello gia' conosciuto e si verifica se e dove appare in una nuova sequenza. Questo approccio e' spesso usato per identificare la presenza di specifici siti funzionali in un genoma o per confermare risultati precedenti.

La principale differenza sta quindi nel punto di partenza: nel motif finding cercano nuovi motivi, mentre nel motif identification si verifica la presenza di motivi gia' noti in nuovi dati.

PARTIAL DIGEST PROBLEM

COS'È IL PARTIAL DIGEST PROBLEM?

Il Partial Digest Problem (PDP) e' un problema che riguarda il tentativo di ricostruire una sequenza di posizioni, come se fossero dei punti su una retta, partendo da un insieme di distanze tra di essi. Si puo' immaginare di avere un gruppo di punti su una linea, ma solo le distanze tra alcune coppie di questi conti sono conosciute, e il compito e' determinare la posizione esatta di ciascun punto.

Questo problema e' particolarmente utile in campi come la biologia computazionale, dove si cerca di dedurre l'ordine dei geni in un cromosoma basandosi sulle distanze tra di essi. Un esempio comune e' quando si hanno dei marcatori genetici e si conoscono le distanze tra di loro, ma non la posizione esatta di ciascun marcatore sulla sequenza del DNA. Risolvere il PDP significa riuscire a ricostruire l'ordine originale di questi marcatori, a partire solo dalle informazioni sulle distanze.

Il funzionamento del problema implica trovare una disposizione che rispetti tutte le distanze date. Cio' significa che, dato un insieme di distanze tra i punti, bisogna trovare una configurazione delle loro posizioni che soddisfi tutte queste distanze.non si parte da una sequenza prestabilita, ma si cerca di trovare una disposizione che, pur rispettando le distanze, faccia sensato il problema nel suo complesso.

Tuttavia, risolvere il PDP non e' un compito facile. La principale difficolta' sta nel fatto che le soluzioni possibili crescono molto rapidamente con l'aumentare del numero di punti e distanze.questo rende il problema difficile da risolvere in modo efficiente per insiemi di dati di dimensioni elevate.molti degli approcci utilizzati per risolvere il PDP, come la forza bruta, provano tutte le combinazioni possibili di posizioni, ma questo e' molto lento e poco pratico per problemi complessi.altri metodi piu' avanzati, come il branch and bound, cercano di migliorare l'efficienza limitando lo spazio di ricerca, ma anche questi non risolvono il problema in tempi rapidi per grandi dimensioni.

La complessita' del problema e' una delle sue caratteristiche piu' critiche. Il PDP e' un problema NP-completo, il che significa che non esistono algoritmi noti che possono risolverlo in tempi rapidi per tutti i casi possibili. Cio' implica che, sebbene si possano trovare soluzioni per piccole istanze del problema, man mano che la dimensione del problema aumenta, il tempo necessario per risolverlo cresce in modo esponenziale, rendendo praticamente impossibile trovare una soluzione per grandi quantita' di dati in tempi utili.

ALGORITMO BRUTE FORCE

Il brute force per il PDP e' un approccio che cerca di risolvere il problema esplorando tutte le possibili soluzioni, senza alcun tipo di ottimizzazione. In pratica, questo metodo consiste nel provare ogni possibile disposizione dei punti sulla retta e verificare se le distanze tra i punti corrispondono esattamente a quelle date nel problema.

Il funzionamento del brute force e' relativamente semplice: dato un insieme di distanze tra punti, il metodo genera tutte le possibili configurazioni dei punti sulla retta. Per ogni configurazione generata, calcola le distanze tra tutte le coppie di punti e le confronta con l'insieme delle distanze fornite dal problema. Se una configurazione soddisfa tutte le distanze, allora la soluzione e' corretta. Se no, si prova con un altro disposizione dei punti.questo processo continua fino a quando si trovano una soluzione che rispetta tutte le distanze, oppure fino a quando tutte le possibili disposizioni sono state esplorate senza successo.

Il principale vantaggio del metodo brute force e che e' garantito trovare la soluzione, se esiste una soluzione valida. Non richiede alcuna conoscenza preliminare delle distanze o delle caratteristiche dei dati, quindi e' molto semplice ed implementare. Inoltre, poiche' esplora ogni possibile combinazione, non c'e' il rischio di saltare alcuna soluzione possibile. Il principale svantaggio del brute force, invece, e' la sua inefficienza. Se il numero di punti e' grande, il numero di permutazioni possibili cresce molto velocemente, portando ad un aumento esponenziale del tempo necessario per esplorare tutte le combinazioni. Questo rende il metodo in impraticabile per problemi di grande dimensione, dove potrebbe essere necessario un tempo computazionale enorme per trovare una soluzione. In pratica, il brute force e' adatto solo a problemi di piccole dimensioni, dove il numero di combinazioni da esplorare e gestibile.

La complessita' del brute force per il PDP e' molto alta, poiche' deve esplorare tutte le possibili permutazioni di n punti. La quantita' di permutazioni di N oggetti e' n! (fattoriale di n), che cresce in modo esponenziale al crescere di n. Questo significa che, con l'aumento del numero di punti, il tempo richiesto per risolvere il problema aumenta drasticamente. Se ci sono n punti, il numero di combinazioni da esplorare e' di ordine O(n!), che e' molto grande anche per valori moderati di n.

ALGORITMO BRANCH AND BOUND

Il Branch and Bound e' un metodo di ottimizzazione che puo' essere utilizzato per risolvere il PDP in modo piu' efficiente rispetto al brute force. Questo approccio riduce lo spazio di ricerca eliminando in anticipo le configurazioni che non possono portare a una soluzione valida, evitando cosi di esplorare tutte le possibili combinazioni.

L'idea alla base di Branch and Bound e' quella di costruire progressivamente la soluzione, mantenendo il controllo sulle distanze generate. Si parte da una soluzione parziale e si aggiungono punti uno alla volta, verificando se le distanze calcolate fino a quel momento sono compatibili con il set di distanze dato. Se a un certo punto ci si accorge che la soluzione parziale non potra' mai portare ad una soluzione completa valida, si interrompe immediatamente l'esplorazione di quel ramo.

Per fare questo, l'algoritmo segue due principi:

1. Branching (ramificazione)
2. Bounding (potatura)

Il Branching (ramificazione) divide il problema in sotto problemi più piccoli. Nel caso del PDP, questo significa provare diverse possibilita' per la posizione di un punto e costruire progressivamente la soluzione.

Il Bounding (potatura) prevede che, se una soluzione parziale non puo' piu' generare una soluzione valida, ad esempio se le distanze generate fino a quel momento non corrispondono a quelle date, il ramo viene scartato, evitando di esplorare ulteriormente combinazioni inutili.

Questo meccanismo riduce drasticamente il numero di combinazioni da esplorare, rendendo il metodo piu' efficiente rispetto alla ricerca esaustiva del brute force.

Uno dei principali vantaggi del Branch and Bound e che riduce significativamente il numero di configurazioni da esplorare, rendendo la ricerca molto piu' veloce rispetto al brute force. Inoltre, garantisce comunque di trovare la soluzione corretta, se esiste, perche' non elimina mai prematuramente soluzioni valide. Invece, il principale svantaggio e' che, nonostante le ottimizzazioni, il problema rimane complesso e, in alcuni casi, Branch and Bound puo' comunque richiedere molto tempo per essere risolto, soprattutto quando il numero di punti e' elevato e la potatura non riesce a ridurre efficacemente lo spazio di ricerca. Inoltre, la sua efficienza dipende fortemente dalla qualita' dell'strategia di bounding: se i limiti scelti non sono stretti, l'algoritmo potrebbe comunque esplorare molte combinazioni superflue.

La complessita' esatta di Branch and Bound per il PDP dipende da quanto efficacemente si riesce a ridurre lo spazio di ricerca, ma in generale e' molto piu' efficiente di O(n!) del brute force. In alcuni casi, puo' avvicinarsi ad una complessita' esponenziale piu' contenuta o addirittura polinomiale per istanze particolari del problema. Tuttavia, nei casi peggiori, puo' comunque degenerare in una ricerca quasi completa, rendendo difficile una stima precisa della sua efficienza.

ALGORITMO DI DIGEST PARZIALE

L'algoritmo di digest parziale e' un metodo specifico per risolvere il PDP in modo piu' efficiente rispetto ad un approccio brute force. L'idea di base e' sfruttare la struttura del problema per costruire la soluzione passo dopo passo, evitando di esplorare combinazioni inutili.

Si parte identificando i punti piu' distanti tra loro, perche' la distanza massima fornita e' sempre quella tra gli estremi della sequenza. Una volta fissati questi punti, si procede aggiungendo progressivamente gli altri, scegliendo da un insieme di distanze rimanenti e verificando che la loro aggiunta sia compatibile con il problema. Se un punto a aggiunto genera insieme di distanze che non corrisponde a quelle fornite, viene scartato e si prova con un'altra possibilita'. Questo processo continua finche' tutti i punti non sono stati assegnati correttamente oppure finche' non si esauriscono le possibilita' valide.

Il vantaggio principale di questo approccio e' che evita di esplorare tutte le possibili disposizioni di punti, riducendo il numero di combinazioni da verificare rispetto al brute force. Questo rende l'algoritmo piu' efficiente e applicabile a insiemi di dati piu' grandi. Tuttavia, il metodo non e' sempre perfetto: se il numero di punti e' molto elevato e le distanze sono numerose, la ricerca puo' comunque diventare complessa, soprattutto se la strategia di selezione dei punti non e' ottimizzata.

ASSEMBLAGGIO GENOMICO

COS'È L'ASSEMBLAGGIO GENOMICO?

L'assemblaggio genomico e' il processo di ricostruzione del genoma di un organismo a partire dalle sequenze di DNA frammentate, utilizzando algoritmi di assemblaggio per ordinare e unire questi frammenti in una sequenza completa.

Le principali applicazioni dell'assemblaggio genomico sono:

1. Ricerca biomedica: studiare varianti genetiche legate a malattie complesse;
2. Agricoltura: ottimizzare piante e colture attraverso l'analisi del DNA;
3. Analisi filogenetica: ricostruire l'evoluzione e le relazioni tra specie.

Le tecnologie di sequenziamento non possono leggere un intero genoma in una sola volta per motivi legati alla grandezza del genoma, alle limitazioni fisiche delle macchine, agli errori di lettura e alla necessita' di trattare il DNA in frammenti piu' piccoli. Quindi, prima di essere sequenziato, il DNA viene frammentato fisicamente o chimicamente in piccoli pezzi per poter essere manipolato e sequenziato dalle macchine, ad esempio usando ultrasuoni o enzimi di restrizione.

Dopo che il DNA e' stato frammentato, si preparano le librerie di sequenza, che sono insiemi di frammenti di DNA con specifici marcatori e adattatori. Questi frammenti vengono poi caricati nelle macchine di sequenziamento. A questo punto, le macchine di sequenziamento leggono ciascun frammento e producono le reads, che sono sequenze di DNA piu' corte. Esistono due tipi principali di reads: le short reads, che sono sequenze corte circa 100-300 basi, e le long reads, che sono sequenze piu' lunghe fino a decine di migliaia di basi. Le tecnologie di sequenziamento come Illumina, PacBio, IonTorrent o Oxford Nanopore leggono quindi questi frammenti e generano la sequenza di ciascuna read, ma non frammentano ulteriormente il DNA: si limitano a leggere e registrare la sequenza di ciascun frammento.

Una volta che il DNA e' stato sequenziato, un computer prende le reads generate dalla sequenziatrice e cerca di ricostruire il genoma completo. Questo processo di assemblaggio si basa sul principio di trovare sovrapposizioni tra i frammenti letti e unire le reads che si sovrappongono, in modo da ricostruire la sequenza originale. Per facilitare questo processo, le reads vengono trasformate in k-mers, cioe' sottosequenze di lunghezza k, generate da software di assemblaggio come Velvet.

SINGLE-END SEQUENCING E PAIRED-END SEQUENCING

Il sequenziamento puo' essere effettuato con due approcci principali: single-end sequencing e paired-end sequencing.

Il single-end sequencing e' un metodo di sequenziamento in cui ogni frammento di DNA viene letto solo da un estremo. Questo significa che, per ogni frammento generato durante il processo di frammentazione del DNA, la macchina di sequenziamento produce una singola sequenza nucleotidica continua, senza fornire informazioni sulla distanza tra le reads.

Il processo inizia con la frammentazione del DNA in pezzi piu' piccoli, che vengono poi adattati alla tecnologia di sequenziamento scelta. Successivamente, la macchina di sequenziamento legge le basi nucleotidiche di ciascun frammento partendo da un solo estremo, generando una read. Questo metodo e' relativamente semplice e veloce, perche' non richiede il ricollegamento di due estremita' dello stesso frammento.

Il single-end sequencing viene utilizzato quando si devono analizzare genomi semplici, senza molte ripetizioni o regioni altamente complesse. E' particolarmente utile per studi che richiedono un'elevata velocita' di sequenziamento e costi ridotti, come il sequenziamento di piccoli genomi batterici o per esperimenti di RNA-Seq in cui e' importante quantificare i livelli di espressione genica piu' che ricostruire l'intera sequenza genomica.

Tuttavia, ci sono situazioni in cui il single-end sequencing potrebbe non essere la scelta migliore. Se il genoma da assemblare contiene molte sequenze ripetitive o strutture complesse, il metodo puo' risultare inefficace perche' non fornisce abbastanza informazioni per distinguere tra regioni simili. Senza una lettura dell'altro estremo del frammento, puo' essere difficile determinare con certezza la posizione relativa di una read rispetto alle altre, aumentando la probabilita' di errori nell'assemblaggio.

Uno dei principali vantaggi del single-end sequencing e' la sua rapidita' e il minor costo rispetto ad altri metodi di sequenziamento. Essendo un processo piu' semplice, richiede meno risorse computazionali per analizzare i dati e puo' essere applicato su larga scala. Tuttavia, lo svantaggio principale e' la mancanza di informazioni sulla struttura del genoma, il che puo' rendere l'assemblaggio meno accurato e aumentare il numero di ambiguita'. Per questo motivo, quando e' necessario ottenere un'alta precisione, soprattutto in genomi complessi, e' preferibile utilizzare il paired-end sequencing.

Il paired-end sequencing e' un metodo di sequenziamento in cui ogni frammento di DNA viene letto da entrambe le estremita'. A differenza del single-end sequencing, che legge solo un'estremita' del frammento, il paired-end sequencing fornisce due reads per ogni frammento, separate da una regione interna di lunghezza nota ma non sequenziata direttamente. Questo approccio migliora notevolmente la qualita' e l'accuratezza dell'assemblaggio genomico.

Il processo inizia con la frammentazione del DNA, ottenuta tramite ultrasuoni o enzimi di restrizione, per generare frammenti di lunghezza adeguata alla tecnologia di sequenziamento utilizzata. Successivamente, a ciascun frammento vengono aggiunti adattatori alle due estremita', che permettono alla macchina di sequenziamento di leggere entrambe le estremita' del frammento in direzioni opposte.

Il paired-end sequencing viene utilizzato quando e' necessario un assemblaggio piu' accurato, in particolare per genomi complessi contenenti regioni ripetitive o difficili da distinguere. Questo metodo e' fondamentale per il de novo assembly, l'identificazione di varianti strutturali, lo studio di genomi eucariotici e per l'analisi metagenomica. Inoltre, e' molto utile in ambito clinico per individuare riarrangiamenti cromosomici e mutazioni legate a malattie genetiche.

Tuttavia, il paired-end sequencing ha alcuni svantaggi. E' piu' costoso rispetto al single-end sequencing, perche' richiede il doppio del tempo di lettura e una maggiore capacita' computazionale per elaborare i dati. Inoltre, la preparazione della libreria di DNA e' piu' complessa, perche' implica l'aggiunta di adattatori specifici per consentire la lettura bidirezionale.

Uno dei principali vantaggi del paired-end sequencing e' la sua maggiore accuratezza nell'assemblaggio genomico. Le due reads provenienti dallo stesso frammento forniscono informazioni sulla distanza tra di loro, aiutando a risolvere ambiguita' e a migliorare l'assemblaggio in regioni ripetitive. Inoltre, riduce il numero di errori e migliora l'allineamento delle reads al genoma di riferimento. Tuttavia, e' piu' costoso e richiede piu' risorse computazionali rispetto al single-end sequencing, quindi e' preferibile utilizzarlo solo quando e' necessario un elevato livello di precisione.

GRAFO DI SOVRAPPOSIZIONE NEL SINGLE-END SEQUENCING

Il grafo di sovrapposizione e' una struttura grafica utilizzata nell'assemblaggio genomico per ricostruire la sequenza di DNA a partire da frammenti letti, conosciuti come reads. In questo grafo, i nodi sono rappresentati dai k-mers, cioe' sottosequenze di lunghezza fissa, estratte dalle reads. Gli archi tra i nodi rappresentano le sovrapposizioni tra i k-mers: due k-mers sono connessi da un arco se il suffisso di uno di essi corrisponde al prefisso di un altro. Le sovrapposizioni sono fondamentali per stabilire la relazione tra i frammenti letti e per ricostruire la sequenza originale.

Il grafo di sovrapposizione e' usato principalmente per assemblare sequenze di long reads. Questo perche' le long reads sono sequenze relativamente lunghe, e quindi, la possibilita' di trovare sovrapposizioni tra i k-mers e' piu' alta e meno problematica. Le long reads permettono di costruire un grafo di sovrapposizione piu' chiaro, con meno ambiguita' nelle sovrapposizioni.

Sebbene il grafo di sovrapposizione sia piu' indicato per le long reads, puo' essere usato anche per short reads, anche se con maggiore difficolta'. Le short reads sono sequenze molto brevi, il che significa che le sovrapposizioni tra i k-mers sono meno evidenti e quindi l'assemblaggio puo' risultare complesso. Inoltre, con le short reads, il rischio di errori e' maggiore, specialmente quando si incontrano sequenze ripetute nel genoma.

Nel processo di assemblaggio genomico tramite grafo di sovrapposizione, la ricostruzione della sequenza completa del genoma puo' essere vista come la ricerca di un cammino hamiltoniano nel grafo. Un cammino hamiltoniano e' un percorso che visita ogni nodo (k-mer) esattamente una volta, e la sua costruzione rappresenta il processo di assemblaggio delle reads nella sequenza corretta.

In alcuni casi, potrebbe esserci la necessita' di trovare un ciclo hamiltoniano, ovvero un percorso che parte e termina nello stesso nodo, visitando ogni nodo esattamente una volta. La presenza di cicli e' legata a situazioni particolari in cui si tratta di sequenze cicliche ripetute nel genoma.

Il problema di trovare il cammino hamiltoniano o il ciclo hamiltoniano in un grafo di sovrapposizione e' un problema NP-completo. Cio' significa che non esiste un algoritmo noto che possa risolvere il problema in modo efficiente per tutte le possibili istanze, e la complessita' cresce rapidamente con l'aumentare del numero di k-mers e delle loro sovrapposizioni. Questo implica che l'assemblaggio genomico attraverso il grafo di sovrapposizione puo' diventare estremamente impegnativo dal punto di vista computazionale, soprattutto quando si lavora con genomi di grandi dimensioni.

Quindi, il grafo di sovrapposizione e' molto utile quando si lavora con long reads, poiche' le sovrapposizioni tra i k-mers sono piu' evidenti e facilmente identificabili. Tuttavia, quando si lavora con short reads, l'assemblaggio attraverso un grafo di sovrapposizione puo' risultare meno preciso e piu' complesso. In questi casi, si preferisce utilizzare il grafo di De Bruijn, che e' piu' efficiente per gestire un gran numero di k-mers derivanti da short reads.

GRAFO DI DE BRUIJN NEL SINGLE-END SEQUENCING

Il grafo di De Bruijn e' una rappresentazione grafica utilizzata nell'assemblaggio genomico, in particolare per le short reads. Il suo obiettivo principale e' quello di rappresentare le sovrapposizioni tra i frammenti di DNA sequenziato in un modo che permetta di ricostruire il genoma completo. La caratteristica distintiva di questo grafo e' che i nodi rappresentano i k-mers, mentre gli archi rappresentano le sovrapposizioni.

Esso e' particolarmente adatto per short reads perche' con letture brevi la dimensione di ogni k-mer e' contenuta e quindi la costruzione del grafo e' relativamente semplice. Le long reads, invece, sono piu' lunghe e contengono piu' informazioni, il che rende la rappresentazione grafica piu' complessa. Le long reads non necessitano tanto di un grafo di De Bruijn per l'assemblaggio, perche' possono essere utilizzate in modo piu' diretto con altri approcci, come i grafi di sovrapposizione, che sono piu' adatti per sequenze piu' lunghe e piu' continue. Tuttavia, il grafo di De Bruijn puo' comunque essere usato anche per long reads, ma si tratta di una scelta meno comune.

Nel grafo di De Bruijn, per ricostruire una sequenza completa di DNA, si cerca un cammino euleriano. Un cammino euleriano e' un percorso che attraversa ogni arco del grafo esattamente una volta. Questo cammino e' utile perche', attraverso di esso, e' possibile risolvere il problema dell'assemblaggio genomico, unendo i frammenti di DNA in un'unica

sequenza piu' lunga.

Un ciclo euleriano e' un percorso che attraversa ogni arco del grafo esattamente una volta e ritorna al punto di partenza, formando un ciclo chiuso. La possibilita' di avere un ciclo euleriano dipende dalla struttura del grafo, ovvero dalla presenza di un percorso che attraversa tutti gli archi in un ciclo chiuso senza ripetizioni.

Il teorema di Eulero afferma che, affinche' esista un cammino euleriano in un grafo, ogni nodo deve essere bilanciato. Questo significa che, per ogni nodo del grafo, il numero di archi entranti deve essere uguale al numero di archi uscenti. Inoltre, un grafo puo' avere al massimo due nodi con grado dispari, ossia con un numero dispari di archi entranti o uscenti. Se ci sono piu' di due nodi con grado dispari, il grafo non ha un cammino euleriano. Nel contesto del grafo di De Bruijn, questa condizione e' particolarmente importante, perche' se non vengono soddisfatte queste regole, l'assemblaggio genomico non sara' possibile attraverso un cammino euleriano. Il grafo deve essere fortemente connesso, cioe' ci deve essere un percorso che collega ogni nodo con tutti gli altri nodi del grafo.

Il problema di trovare un cammino euleriano nel grafo di De Bruijn e' un problema NP-completo, il che significa che e' molto difficile trovare una soluzione ottimale in tempo polinomiale quando il grafo e' molto grande, come accade nell'assemblaggio genomico. La difficolta' aumenta con la dimensione del grafo, perche' il numero di k-mers e di archi da analizzare cresce esponenzialmente con la lunghezza delle reads e con la complessita' del genoma da assemblare.

Dunque, il grafo di De Bruijn e' utilizzato principalmente per short reads e quando si hanno grandi quantita' di dati. E' vantaggioso soprattutto quando si vuole rappresentare in modo compatto tutte le sovrapposizioni tra i k-mers e utilizzare questi dati per assemblare il genoma. Invece, e' meno adatto per long reads.

GRAFO DI SOVRAPPOSIZIONE NEL PAIRED-END SEQUENCING

Per il grafo di sovrapposizione, non c'e' una versione equivalente specifica per il paired-end sequencing. Il grafo di sovrapposizione di per se' e' gia' un tipo di grafo che si adatta abbastanza bene a entrambi i tipi di sequenziamento, single-end e paired-end. Tuttavia, quando si lavora con paired-end sequencing, l'uso di coppie di letture non cambia radicalmente la struttura del grafo di sovrapposizione, ma puo' migliorare l'accuratezza della ricostruzione del genoma. Questo accade perche' le informazioni provenienti da due reads, che rappresentano le estremita' opposte di un frammento di DNA, possono essere utilizzate per confermare le sovrapposizioni, ridurre gli errori e risolvere le ambiguita' nell'assemblaggio.

In pratica, con il paired-end sequencing, si ha un vantaggio nell'assemblaggio attraverso la combinazione delle due reads che forniscono informazioni aggiuntive sulla distanza tra di loro, aiutando a risolvere alcune problematiche di sovrapposizione, specialmente quando si trattano sequenze piu' lunghe o regioni piu' difficili da assemblare. Tuttavia, non esiste una versione dedicata o strutturalmente diversa del grafico di sovrapposizione, come esiste per il grafo di De Bruijn il Pair De Bruijn Graph.

GRAFO DI DE BRUIJN NEL PAIRED-END SEQUENCING

Il Pair De Bruijn Graph e' un'estensione del grafo di De Bruijn progettata per sfruttare le informazioni aggiuntive fornite dal paired-end sequencing, una tecnica in cui le reads vengono sequenziate in coppie con una distanza nota tra di loro. Questa informazione consente di migliorare l'accuratezza dell'assemblaggio genomico, specialmente in presenza di regioni ripetute o sequenze complesse che potrebbero confondere il metodo tradizionale basato sui singoli k-mers.

Nel Pair De Bruijn Graph, invece di costruire un grafo basato sui singoli k-mers, vengono generati k-mers pairs, ovvero coppie di k-mers separati da una distanza fissa, chiamata insert size, che corrisponde alla distanza tra i due frammenti letti durante il sequenziamento. Ogni nodo del grafo rappresenta una coppia di k-mers, mentre gli archi collegano coppie di k-mers che condividono una sovrapposizione coerente con la distanza tra di loro.

L'uso del Pair De Bruijn Graph offre diversi vantaggi rispetto al De Bruijn Graph tradizionale. Innanzitutto, la conoscenza della distanza tra i k-mers permette di risolvere piu' facilmente regioni ripetute nel genoma, poiche' le informazioni sulla separazione tra le coppie riducono le ambiguita' nel percorso di assemblaggio. Inoltre, l'uso di coppie di k-mers aiuta a collegare regioni che altrimenti sarebbero frammentate a causa della mancanza di sovrapposizioni dirette tra le reads.

Dal punto di vista computazionale, il Paired De Bruijn Graph introduce una maggiore complessita' rispetto al De Bruijn Graph classico, poiche' il numero di nodi e archi puo' aumentare notevolmente. Tuttavia, questo incremento di complessita' e' compensato dalla maggiore robustezza dell'assemblaggio, soprattutto nei casi in cui il grafo di De Bruijn tradizionale produrrebbe molteplici contigs disconnessi. Il Pair De Bruijn Graph consente quindi di ottenere un assemblaggio piu' accurato e continuo, riducendo la frammentazione del genoma ricostruito e migliorando la risoluzione delle regioni difficili da assemblare con il metodo standard.

PROBLEMA DELLA STRINGA UNIVERSALE

La stringa universale e' cio' che si desidera ottenere attraverso l'assemblaggio genomico, sia utilizzando il grafo di sovrapposizione che il grafo di De Bruijn. Essa rappresenta una sequenza di DNA che contiene tutte le possibili sottosequenze, o k-mers, che si potrebbero estrarre dalle reads sequenziate.

TOOL VELVET

Velvet e' un software di assemblaggio de novo progettato per ricostruire sequenze genomiche a partire da brevi reads, particolarmente adatto per il sequenziamento di nuova generazione. Il suo funzionamento si basa sulla creazione di un grafo di De Bruijn, in cui i nodi rappresentano i k-mers derivati dalle reads, mentre gli archi collegano i k-mers che si sovrappongono. Il processo di assemblaggio in Velvet e' indirizzato a risolvere alcuni problemi tipici dei dati di sequenziamento, come la gestione degli errori nelle reads, la risoluzione delle "bubble", cioe' delle situazioni in cui il grafo ha piu' possibili percorsi paralleli, e l'assemblaggio dei contig.

Una delle principali sfide che Velvet affronta e' la rimozione degli errori nel grafo di De Bruijn, che puo' essere causato da piccole mutazioni o errori di sequenziamento. Velvet gestisce questi errori identificando e filtrando i k-mers non affidabili, riducendo cosi' la possibilita' che errori possano influenzare l'assemblaggio finale. Un altro aspetto importante e' la gestione delle bubble, che possono sorgere in sequenze genomiche complesse, dove diverse varianti della sequenza si sovrappongono. Velvet risolve questo problema, cercando di identificare il percorso piu' probabile e ottimizzando l'assemblaggio in base ai dati disponibili.

Tuttavia, Velvet ha anche dei limiti. Ad esempio, l'assemblaggio puo' essere influenzato dalla scelta del valore di k, che puo' avere un impatto significativo sulla qualita' dell'assemblaggio finale. Se k e' troppo piccolo, potrebbe esserci una maggiore quantita' di rumore nel grafo; se e' troppo grande, alcune informazioni potrebbero andare perse. Inoltre, Velvet puo' faticare con genomi molto complessi o altamente ripetitivi, poiche' il grafo di De Bruijn puo' diventare intricato e difficile da risolvere. Infine, l'uso di Velvet e' limitato principalmente alle short reads, rendendolo meno efficace rispetto ad altri assemblatori per dati provenienti da tecnologie di sequenziamento long reads, come Oxford Nanopore o PacBio. Anche le risorse computazionali richieste possono diventare significative quando si lavora con dataset molto grandi.

Nonostante questi limiti, Velvet e' ampiamente utilizzato per l'assemblaggio di genomi batterici, virali e di piccoli eucarioti grazie alla sua efficienza nella gestione di short reads e alla capacita' di produrre contig di alta qualita'. E' un tool particolarmente utile per chi lavora con dati di sequenziamento Illumina, poiche' permette di ottenere un'assemblaggio affidabile partendo da dati rumorosi e frammentati.

PROGETTO SUS-MIRRI.IT

Il progetto SUS-MIRRI.IT mira a rafforzare l'infrastruttura di ricerca italiana nel campo delle risorse microbiche, promuovendo la bioscienza e la bioeconomia sostenibili. Finanziato con circa 17 milioni di euro dal Piano Nazionale di Ripresa e Resilienza (PNRR) nell'ambito del programma NextGenerationEU, il progetto e' coordinato dall'Universita' di Torino e coinvolge 15 istituzioni con 24 unita' operative.

Esso si concentra sull'analisi e la caratterizzazione dei microrganismi presenti nelle biobanche, con l'obiettivo di comprendere meglio il loro ruolo nel mantenimento della salute umana e nell'insorgenza di malattie. I microrganismi hanno un impatto fondamentale sulla nostra salute, ad esempio attraverso il microbioma intestinale, che puo' influenzare vari aspetti, dalla digestione alla risposta immunitaria.

Inoltre, studiando queste risorse biologiche, il progetto mira anche a scoprire nuove applicazioni biotecnologiche, come la produzione di farmaci, enzimi e altri composti utili per la medicina e altre industrie. Quindi, oltre a contribuire alla ricerca medica, il progetto sostiene anche la bioeconomia sostenibile.

CLUSTERING

COS'È IL CLUSTERING?

Prima di dare una definizione di clustering, occorre capire la differenza tra apprendimento supervisionato e apprendimento non supervisionato.

Nell'apprendimento supervisionato ogni esempio del dataset e' associato ad un'etichetta predefinita che indica la categoria o il valore corretto dell'output. L'obiettivo e' addestrare un modello a riconoscere schemi nei dati per poi classificare correttamente nuove istanze. Alcuni esempi tipici sono la classificazione di immagini e la previsione di valori numerici.

Invece, nell'apprendimento non supervisionato, i dati non sono etichettati e l'algoritmo deve autonomamente identificare strutture o schemi nascosti. Il clustering e' una delle tecniche principali di questo approccio: il modello suddivide i dati in gruppi in base alla loro similarita', senza una conoscenza a priori delle categorie esistenti. Questo e' utile in contesti in cui le classi non sono note in anticipo, come la segmentazione di clienti in base ai loro comportamenti o l'identificazione di pattern in dati biologici.

La principale differenza tra i due approcci sta dunque nel tipo di guida fornita al modello: nell'apprendimento supervisionato il sistema impara da esempi con risposte corrette, mentre nell'apprendimento non supervisionato esplora autonomamente la struttura dei dati senza indicazioni esplicite.

Dunque, il clustering e' una tecnica di apprendimento non supervisionato utilizzata per raggruppare insiemi di dati in sottogruppi, chiamati cluster, in modo che gli elementi all'interno di ciascun gruppo siano piu' simili tra loro rispetto agli elementi appartenenti a cluster diversi. A differenza della classificazione supervisionata,che utilizza etichette predefinite per addestrare un modello, il clustering identifica autonomamente e strutture nei dati senza una conoscenza preliminare delle categorie.

Questa tecnica ha molte applicazioni nel mondo reale.viene utilizzata nel marketing segmentare i clienti in base ai loro comportamenti d'acquisto, nell'analisi dei social media per individuare comunita' di utenti con interessi simili e nella rilevazione di anomalie per identificare attivita' sospette, come le frodi finanziarie. nell'ambito della bioinformatica, il clustering e' fondamentale per l'analisi dei dati biologici, come la classificazione di geni con funzioni simili, il raggruppamento di cellule in studi di espressione genica e la scoperta di nuove malattie basandosi su dati clinici e molecolari.

Esistono diversi approcci al clustering, ognuno con caratteristiche specifiche:

1. Clustering partizionale: suddivide i dati in un numero fisso di cluster;
2. Clustering gerarchico: crea una struttura ad albero di cluster senza predefinire il numero;
3. Clustering basato sulla densita': identifica cluster in base alla densita' dei punti;
4. Clustering basato su griglie: raggruppa i dati suddividendo lo spazio in celle regolari;
5. Clustering basato su modelli: assegna i dati a cluster basati su distribuzioni probabilistiche.

Una distinzione importante e' tra single clustering e co-clustering. Nel primo caso, gli oggetti vengono raggruppati rispetto ad una singola dimensione dei dati. Invece, nel co-clustering, si considerano contemporaneamente due insiemi, ad esempio campioni biologici e geni, creando una struttura a blocchi che puo' rivelare interazioni nascoste. Esiste anche il co-clustering con vincoli, che introduce condizioni aggiuntive, come conoscenze a priori, per migliorare la qualita' della suddivisione.

In generale, il clustering offre diversi vantaggi, tra cui la capacita' di scoprire strutture nascoste nei dati senza bisogno di etichette predefinite. E' utile per segmentare dati complessi, identificare pattern e ridurre la dimensionalita', trovando gruppi naturali in grandi insiemi di informazioni. Inoltre, permette di analizzare dati non strutturati in modo flessibile e adattabile a diversi contesti, come il marketing, la bioinformatica e la rilevazione di anomalie.

Tuttavia, presenta anche diverse sfide. La scelta del numero di cluster e' spesso difficile e puo' influenzare i risultati.i I metodi di clustering sono sensibili alla metrica di distanza utilizzata, alla scala dei dati e alla presenza di outliers, cioe' dati anomali che si discostano significativamente dagli altri valori, potenzialmente indicando errori, variazioni rare o informazioni utili. Inoltre, alcuni algoritmi possono essere computazionalmente costosi o non adatti a dataset di grandi dimensioni. Infine, valutare la qualita' del clustering e' complesso, poiche' non esiste un criterio unico e universale per misurarne l'efficacia.

BONTÀ DEL CLUSTERING

Un buon clustering si distingue per la compattezza e la inseparabilita' dei cluster: i punti all'interno di un cluster devono essere molto simili tra loro (alta coesione), mentre i cluster devono essere ben distinti l'uno dall'altro (alta separabilita'). Inoltre, deve essere stabile, ovvero riproducibile con piccole variazioni nei dati, e significativo, riflettendo strutture reali e utili per l'analisi. infine, deve gestire correttamente gli outliers, isolandoli o assegnandoli in modo appropriato senza distorcere la struttura complessiva dei cluster.

Un cattivo clustering, invece, produce gruppi poco coerenti, con punti mal assegnati o cluster sovrapposti. Puo' essere influenzato da scelte errate di parametri, come il numero di cluster, o da un metodo non adatto alla distribuzione dei dati. Un clustering inefficace puo' anche essere instabile, variando troppo a fronte di piccole modifiche nei dati, o privo di interpretabilita', senza un valore pratico per l'analisi.

CLUSTERING PARTIZIONALE

Il clustering partizionale e' una tecnica di analisi dei dati utilizzata per suddividere un insieme di oggetti in gruppi, chiamati cluster, in modo che gli oggetti all'interno di ciascun cluster siano simili tra loro e differenti da quelli appartenenti ad altri cluster. In pratica, questa tecnica cerca di trovare una struttura nascosta nei dati, organizzandoli in gruppi omogenei. Ogni oggetto e' assegnato a uno e un solo cluster, e il numero di cluster ed definito in anticipo, oppure determinato in base alla configurazione del modello.

Il funzionamento del clustering partizionale si basa sull'idea di minimizzare una funzione obiettivo, che misura la coesione all'interno dei cluster e la separazione tra i cluster stessi. Piu' precisamente, l'obiettivo e' ridurre la distanza intra-cluster, cioe' la distanza tra gli oggetti all'interno dello stesso cluster, e aumentare la distanza inter-cluster, cioe' la distanza tra oggetti appartenenti a cluster diversi.

Il k-means e' uno degli algoritmi di clustering piu' popolari e funziona suddividendo un insieme di dati in k cluster, dove k e' un numero scelto dall'utente. Il processo inizia con la selezione casuale di k centroidi, che rappresentano il centro di ciascun cluster. Ogni punto del dataset viene assegnato al centroide piu' vicino, basandosi solitamente sulla distanza euclidea. Dopo che tutti i punti sono stati assegnati ad un cluster, i centroidi vengono aggiornati come la media dei punti del rispettivo cluster. Il processo di assegnazione e aggiornamento continua fino a quando i centroidi non cambiano piu' significativamente, indicandone la convergenza.

Tra i vantaggi di k-means ci sono la sua semplicita', la velocita' e l'efficienza computazionale, che lo rendono particolarmente adatto per dataset di grandi dimensioni. E' anche relativamente facile da implementare e ha una buona interpretabilita', poiche' ogni punto appartiene ad un solo cluster e ogni cluster e' definito dal suo centroide. Tuttavia, ha alcuni svantaggi: prima di tutto, e' insensibile all'inizializzazione dei centroidi, che puo' portare a risultati diversi in base alla scelta iniziale. Inoltre, k-means richiede che il numero di cluster k sia definito a priori, il che puo' essere difficile senza una conoscenza preliminare dei dati. Un altro limite e' che l'algoritmo e' sensibile agli outliers, che possono influenzare notevolmente la posizione dei centroide e quindi la qualita' del clustering.

La complessita' computazionale di k-means e' O(nkid), dove n e' il numero di punti, k e' il numero di cluster, i e' il numero di iterazioni e d e' la dimensione dei dati. Questo implica che l'algoritmo puo' essere relativamente veloce su grandi dataset, ma il numero di iterazioni dipende dalla distribuzione dei dati e dalla qualita' dell'inizializzazione, il che puo' aumentare il tempo di esecuzione.

Il k-modes e' una variante dell''algoritmo k-means progettata per lavorare con dati categorici. Mentre k-means e' adatto ai dati numerici e utilizza la media come centroide per rappresentare ciascun cluster, k-modes impiega la moda, cioe' il valore piu' frequente, per determinare il centro del cluster. Questo permette a k-modes di gestire correttamente variabili non numeriche, come categorie, etichette o altre caratteristiche discrete.

Il funzionamento di k-modes e' simile a quello di k-means. Si inizia con una scelta casuale di k centroidi (modi iniziali). Ogni punto del dataset viene assegnato al cluster in cui il centroide ha il valore della moda piu' simile. Dopo aver assegnato tutti i punti ai rispettivi cluster, k-modes ricalcola i centroide come la moda dei punti nel cluster, cioe' il valore piu' frequente per ogni attributo del cluster. Il processo continua iterativamente fino a che i centroide non cambiano piu' significativamente, segnalando la convergenza.

I vantaggi di k-modes sono evidenti quando si lavora con dati categorici, dove il calcolo della media (come in k-means) non avrebbe senso. k-modes e' quindi piu' appropriato e utile in situazioni in cui si desidera eseguire il clustering su variabili non numeriche. E' anche relativamente semplice da implementare e veloce, e ha una buona capacita' di separare i dati in base alle modalita' di ciascuna variabile. Tuttavia, ci sono anche svantaggi: uno dei principali e' che k-modes richiede comunque di specificare k, il numero di cluster, prima di eseguire l'algoritmo e questo puo' essere difficile senza una conoscenza preliminare dei dati. Inoltre, se i dati hanno molte modalita' diverse o se ci sono molti valori rari, k-modes puo' risultare meno efficace. k-modes e' anche sensibile agli outliers, anche se in misura minore rispetto a k-means, ma potrebbe comunque non essere il metodo migliore per dati molto rumorosi o con distribuzioni irregolari.

La complessita' computazionale di k-modes e simile a quella di k-means, ossia O(nkid), dove n e' il numero di punti, k e' il numero di cluster, i e' il numero di iterazioni e d e' il numero di variabili. Anche se la complessita' dipende dal numero di iterazioni, in pratica k-modes e' generalmente piu' veloce rispetto a metodi piu' complessi, ma comunque puo' essere influenzato dalla scelta di k e dalla distribuzione dei dati.

k-prototype e' un algoritmo di clustering che estende k-means e k-modes per gestire dataset misti, ossia dataset che contengono sia dati numerici che categorici. Mentre k-means e' adatto ai dati numerici e k-modes e' adatto ai dati categorici, k-prototype e' progettato per trattare entrambe le tipologie di variabili nello stesso insieme di dati.

Il funzionamento di k-prototype e' molto simile a quello di k-means e a quello di k-modes, ma per ogni cluster, invece di un singolo centroide, viene calcolato un prototipo che combina la media per le variabili numeriche e la moda per quelle categoriche. I punti vengono assegnati ai cluster in base alla distanza calcolata separatamente per le variabili numeriche e categoriche. Poi, i prototipi vengono aggiornati in base ai nuovi assegnamenti.

Il principale vantaggio di k-prototype e' la capacita' di gestire data set misti senza la necessita' di preprocessare i dati. Tuttavia, come k-means e k-modes, e' sensibile alla scelta del numero di cluster k e agli outliers. La sua complessita' computazionale e' $O(nki(d_n + d_c))$, dove n e' il numero di punti, k e' il numero di cluster, i e' il numero di iterazioni, d_n e' il numero di variabili numeriche e d_c e' il numero di variabili categoriche.

k-medoids e' un algoritmo di clustering che rappresenta una variante piu' robusta di k-means, particolarmente utile quando i dati contengono outliers o valori anomali. A differenza di k-means, che utilizza i centroidi (punti medi) come rappresentanti dei cluster, k-medoids utilizza punti reali del dataset, chiamati "medoids", come centri di ciascun cluster. In altre parole, un medoid e' un punto effettivo del dataset che meglio rappresenta il cluster, piuttosto che un punto calcolato come la media dei dati.

Il funzionamento di k-medoids e' simile a quello di k-means, ma con alcune differenze cruciali. Prima si sceglie un numero k di cluster e si inizializzano k medoids casualmente. Poi, ogni punto del dataset viene assegnato al medoid piu' vicino, basandosi su una misura di distanza, che solitamente e' la distanza euclidea o altre metriche di distanza. Dopo che i punti sono stati assegnati ai rispettivi cluster, i medoids vengono aggiornati: in ogni cluster, il medoid e' scelto come il punto che minimizza la somma delle distanze da tutti gli altri punti nel cluster. Questo processo viene ripetuto iterativamente fino a quando i medoids non cambiano piu' significativamente, raggiungendo la convergenza.

I vantaggi di k-medoids rispetto a k-means sono evidenti quando si lavora con dati che possono contenere outliers o valori estremi. Poiche' i medoids sono punti reali e non sono influenzati dalla media, l'algoritmo e' piu' robusto agli outliers. Inoltre, k-medoids puo' essere utilizzato con diverse metriche di distanza, inclusi metodi non euclidei, rendendolo adatto per un'ampia varieta' di applicazioni rispetto a k-means. Tuttavia, gli svantaggi di k-medoids sono legati principalmente alla sua maggiore complessita' computazionale rispetto a k-means. Poiche' il processo di aggiornamento dei medoids comporta il calcolo delle distanze tra tutti i punti e il medoid, k-medoids puo' essere molto piu' lento, specialmente con dataset di grandi dimensioni. Inoltre, come per k-means, e' necessario specificare in anticipo il numero k di cluster, e l'algoritmo e' sensibile alla scelta iniziale dei medoids, che puo' influenzare il risultato finale.

La complessita' computazionale di k-medoids e' generalmente $O(n_2ki)$, dove n e' il numero di punti, k e' il numero di cluster e i e' il numero di iterazioni. Questo lo rende meno scalabile di k-means, soprattutto per dataset molto grandi, poiche' il calcolo delle distanze tra tutti i punti e i medoids e' un'operazione costosa.

PAM (Partitioning Around Medoids) e' una variante specifica di k-medoids. E' stato progettato per migliorare l'algoritmo k-means, con il vantaggio di essere piu' robusto agli outliers e di utilizzare punti reali nel dataset come centri dei cluster, chiamati "medoids", invece dei centroidi, che sono punti calcolati come la media. PAM e' particolarmente utile in situazioni dove i dati possono contenere valori anomali o outliers che potrebbero influenzare il risultato del clastering se venissero utilizzati i centroidi.

Il funzionamento di PAM e' simile a quello di k-medoids. Prima di tutto, si sceglie il numero di cluster k e si inizializzano k medoids casualmente. Successivamente, ogni punto del dataset viene assegnato al medoid piu' vicino, utilizzando una metrica di distanza (di solito la distanza euclidea). Dopo aver assegnato tutti i punti ai rispettivi cluster, il passo successivo consiste nell'ottimizzare i medoids: per ogni cluster, viene scelto il punto del cluster che minimizza la somma delle distanze tra il medoid e gli altri punti del cluster. Questo processo continua iterativamente, aggiornando i medoids e riassegnando i punti ai nuovi medoids, fino a quando non si raggiunge la convergenza, cioe' quando i medoids non cambiano piu'.

Un vantaggio di PAM e' la sua robustezza rispetto agli outliers. Poiche' i medoids sono punti effettivi del dataset, l'algoritmo e' meno influenzato da valori estremi rispetto a k-means, utilizza la media dei punti come centroide. Inoltre, PAM puo' utilizzare diversi metriche di distanza, non limitandosi alla distanza euclidea, che lo rende piu' flessibile rispetto a K-means, che puo' essere applicato solo a dati numerici. Uno degli svantaggi principali di PAM e' la sua complessita' computazionale.l'algoritmo ha una complessita' di $O(n^2k)$ dove n e' il numero di punti nel dataset e k e' il numero di cluster.questo lo rende piuttosto lento, specialmente sudata set molto grandi, poiche' il calcolo delle distanze tra tutti i punti e i medoids puo' essere oneroso. Sebbene il miglioramento rispetto a k-means in termini di robustezza sia evidente, Pam non e' adatto per grandi dataset a causa della sua inefficienza computazionale. Per migliorare l'efficienza di Pam sudata set piu' grandi, sono stati sviluppati algoritmi come CLARA (Clustering LARge Applications) e CLARANS (Clustering LARge Applications based on Randomized Search).

CLARA (Clustering LARge Applications) e' una versione migliorata di PAM (Partitioning Around Medoids), progettata per affrontare la complessita' computazionale elevata di PAM quando si lavora con dataset molto grandi. L'algoritmo CLARA si concentra sull'efficienza, riducendo il carico computazionale senza compromettere troppo la qualita' del clustering.

CLARA utilizza una tecnica di campionamento per ridurre la quantita' di dati con cui l'algoritmo deve lavorare. Invece di eseguire l'algoritmo di clustering su tutto il dataset, CLARA seleziona casualmente un sottoinsieme di punti (un campione) da un dataset piu' grande, esegue l'algoritmo di clustering PAM su questo campione e quindi applica i medoids risultanti all'intero dataset per assegnare i punti ai cluster. Questo processo viene ripetuto piu' volte con campioni diversi per cercare la soluzione di clustering migliore. In pratica, CLARA esegue PAM su diversi campioni casuali e seleziona la configurazione di medoids che minimizza il costo totale, che e' basato sulla somma delle distanze tra i punti e i medoids.

Il vantaggio principale di CLARA e' che, riducendo la quantita' di dati su cui eseguire il clustering, l'algoritmo e' piu' veloce e scalabile rispetto a PAM, rendendolo adatto a dataset di dimensioni molto grandi. Inoltre, come PAM, CLARA mantiene la robustezza contro gli outliers, poiche' continua a usare i medoids, che sono punti reali del dataset. Gli svantaggi di CLARA sono legati al fatto che, sebbene il campionamento riduca il tempo di calcolo, puo' perdere precisione rispetto all'uso dell'intero dataset. Poiche' il clustering viene eseguito su sottoinsiemi casuali dei dati, la qualita' del clustering potrebbe essere influenzata dalla scelta dei campioni. Se il campione selezionato non e' rappresentativo dell'intero dataset, i risultati potrebbero non essere ottimali. Inoltre, come PAM, anche CLARA richiede che il numero di cluster k venga definito a priori, il che puo' essere un problema se non si ha una buona idea del numero di cluster.

La complessita' computazionale di CLARA e' molto piu' bassa rispetto a PAM, poiche' l'algoritmo lavora solo con campioni ridotti anziche' con l'intero dataset: $O(n*(s^2k+k*s*\log(s)))$, dove n e' il numero di campioni, s e' la dimensione del campione casuale, s^2k riguarda la fase di assegnazione dei punti ai medoids e k*s*log(s) riguarda l'aggiornamento dei medoids. Sebbene la complessita' di CLARA dipenda dal numero di campioni e dalle dimensioni di ciascun campione, e' generalmente molto piu' veloce e piu' scalabile rispetto a PAM. Questo lo rende adatto a grandi dataset, ma con il rischio che i risultati possano non essere completamente accurati se il campione non e' rappresentativo.

CLARANS (Clustering Large Applications based on RANdomized Search) e' un algoritmo di clustering sviluppato per migliorare l'efficienza del PAM e di CLARA, affrontando il problema della complessita' computazionale elevata quando si lavora con dataset molto grandi. Come PAM, CLARANS si basa sul concetto di medoids, ma utilizza una ricerca randomizzata per ottimizzare il processo di clustering e ridurre i tempi di calcolo.

Il funzionamento di CLARANS e' diverso rispetto a PAM e CLARA. Invece di utilizzare un campionamento casuale come in CLARA, CLARANS esegue una ricerca stocastica per trovare i migliori medoids. L'algoritmo inizia con una scelta casuale dei medoids iniziali, proprio come gli altri metodi. Tuttavia, invece di calcolare le distanze tra tutti i punti e i medoids in ogni iterazione (come fa PAM), CLARANS esplora casualmente un numero limitato di medoids vicini e aggiorna i medoids solo se la nuova configurazione migliora la qualita' del clustering. Questo processo riduce il numero di calcoli necessari, ma puo' comunque trovare una soluzione di clustering che e' abbastanza buona.

In pratica, l'algoritmo esplora in modo casuale un insieme di soluzioni vicine, scegliendo i migliori tra questi e continuando il processo iterativo. La ricerca e' limitata, quindi la complessita' computazionale e' piu' bassa rispetto a PAM, e la ricerca randomizzata aiuta a evitare di essere bloccati in minimi locali. La ricerca continua fino a quando non si raggiunge una soluzione stabile, dove i medoids non cambiano piu' significativamente.

I principali vantaggi di CLARANS sono la velocita' e la scalabilita', rendendolo adatto a dataset molto grandi. La ricerca randomizzata permette di evitare l'esplorazione completa dello spazio delle soluzioni, riducendo il numero di operazioni e accelerando il processo di clustering rispetto a PAM. Inoltre, CLARANS e' piu' robusto rispetto agli outliers, in quanto come PAM e k-medoids utilizza i medoids, che sono meno sensibili ai valori estremi rispetto ai centroidi. Gli svantaggi principali di CLARANS riguardano la qualita' del clustering. Poiche' la ricerca e' randomizzata, c'e' il rischio che l'algoritmo non trovi la soluzione ottimale, specialmente se il numero di vicini esplorati e' troppo basso. La scelta dei parametri, come il numero di vicini da esplorare e il numero di iterazioni, puo' influenzare notevolmente il risultato. Inoltre, come in PAM e CLARA, e' necessario specificare il numero di k cluster a priori, il che puo' essere una limitazione se non si ha una conoscenza preliminare dei dati.

La complessita' computazionale di CLARANS e' notevolmente ridotta rispetto a PAM, ed e' generalmente $O(kn(\log(n))i)$, dove k e' il numero di cluster, n e' il numero di punti e i e' il numero di iterazioni. La ricerca limitata dei vicini riduce drasticamente il numero di calcoli necessari, rendendo l'algoritmo molto piu' veloce rispetto a PAM e CLARA.

CLUSTERING GERARCHICO

Il clustering gerarchico e' una tecnica di clustering che mira a costruire una gerarchia di cluster, in cui i dati sono organizzati in una struttura ad albero, chiamata dendrogramma, che mostra come i cluster sono uniti o separati a livelli progressivi. Questa tecnica non richiede di specificare in anticipo il numero di cluster, a differenza del clustering partizionale come k-means.

Il clustering gerarchico puo' essere suddiviso in due principali approcci:

1. Approccio bottom-up: clustering agglomerativo;
2. Approccio top-down: clustering divisivo.

Nel clustering agglomerativo, che e' l'approccio piu' comune, si parte considerando ogni punto come un singolo cluster. Successivamente, in ogni iterazione, i due cluster piu' simili vengono uniti per formare un nuovo cluster. Questo processo continua fino a quando non tutti i punti sono raggruppati in un unico cluster, creando una gerarchia di unioni. La misura di similarita' tra i cluster puo' essere calcolata in vari modi, come la distanza minima (single linkage), la distanza massima (complete linkage), la distanza media (average linkage) o la distanza centrata sui centroidi (centroid linkage). L'agglomerazione avviene procedendo dalla base, ovvero dai singoli punti, e i cluster si uniscono via via che il processo prosegue. La costruzione del dendrogramma permette di visualizzare il clustering a vari livelli di granularita', consentendo di scegliere il numero di cluster in base al livello di taglio desiderato nell'albero.

Nel clustering divisivo, al contrario, si parte considerando tutti i dati come un singolo cluster. Da questo cluster iniziale, l'algoritmo cerca di suddividerlo in due o piu' cluster, scegliendo il miglior punto di divisione in base alla similarita' tra i dati. Ogni nuovo cluster formato puo' essere ulteriormente suddiviso, fino a quando tutti i punti sono separati in cluster distinti. Questo approccio e' meno comune rispetto a quello agglomerativo perche' tende ad essere piu' complesso e computazionalmente piu' costoso ma puo' essere utile in alcuni casi specifici dove e' necessario un controllo maggiore su come i dati vengono divisi in modo top-down.

AGNES (AGglomerative NESting) e' un algoritmo di clustering gerarchico che segue l'approccio agglomerativo (bottom-up) per costruire una gerarchia di cluster. In AGNES, si inizia considerando ogni dato come un singolo cluster, e successivamente i cluster vengono uniti iterativamente sulla base della somiglianza tra di essi, fino ad ottenere un unico cluster che include tutti i dati.

Il processo di funzionamento di AGNES inizia con ogni punto che costituisce un cluster separato. L'algoritmo calcola la distanza (o similitudine) tra ogni coppia di cluster. in base alla misura di distanza scelta, ad esempio la distanza euclidea, la distanza di Manhattan o altre metriche, i due cluster piu' vicini vengono uniti in un unico nuovo cluster. Questo processo continua iterativamente: ad ogni passaggio, i due cluster piu' vicini vengono fusi, e la distanza tra il nuovo cluster risultante e gli altri cluster viene aggiornata. L'algoritmo ripete questo processo finche' non viene creato un singolo cluster che contiene tutti i punti. La misura di distanza tra i cluster in AGNES puo' essere calcolata in vari modi. I metodi piu' comuni includono la distanza minima (single linkage), la distanza massima (complete linkage), la distanza media (average linkage) o la distanza centrata sui centroidi (centroid linkage).

Un aspetto distintivo di AGNES e' che, essendo un algoritmo bottom-up, non e' necessario specificare il numero di cluster in anticipo. L'algoritmo costruisce un dendrogramma, che e' una rappresentazione grafica della struttura gerarchica dei dati, dove ogni ramo rappresenta un'unione di cluster. La divisione dei dati in un numero predefinito di cluster puo' essere fatta tagliando il dendrogramma ad un livello desiderato, il che consente di scegliere il numero di cluster in base alla struttura gerarchica dei dati.

I principali vantaggi di AGNES includono la sua capacita' di gestire dati non lineari e cluster di forme diverse, poiche' non assume alcuna struttura predefinita come nel caso di k-means. Inoltre, la creazione del dendrogramma consente una visione dettagliata delle relazioni tra i cluster. Tuttavia, uno degli svantaggi principali di AGNES e' la sua complessita' computazionale. Il calcolo delle distanze tra tutti i punti e l'aggiornamento delle distanze tra i cluster in ogni interazione richiede un tempo significativo, in particolare per grandi dataset. La complessita' di AGNES e' $O(n^3)$, che puo' risultare proibitiva per dataset molto grandi.

DIANA (DIvisive ANAlysis) e' un algoritmo di clustering gerarchico che segue l'approccio divisivo (top-down), in contrapposizione all'approccio agglomerativo di AGNES. A differenza di AGNES, che parte trattando ogni punto come un cluster separato e poi unisce i cluster, DIANA parte considerando tutti i dati come un singolo cluster e li suddivide progressivamente in sottogruppi.

Il funzionamento di DIANA inizia con l'intero dataset considerato come un unico cluster. Successivamente, l'algoritmo esamina il cluster e cerca di trovare il punto o la divisione che massimizza la distanza tra i dati, separando il cluster in due sotto-cluster distinti. La divisione avviene cercando il punto di divisione che massimizza la varianza o la dissimilarita' all'interno dei due nuovi cluster. una volta effettuata questa divisione iniziale, ogni sotto-cluster risultante viene trattato come un nuovo cluster, e il processo di divisione continua per ciascun cluster in modo ricorsivo, finche' ogni punto non forma un cluster separato.

La misura di distanza o dissimilarita' utilizzata da DIANA e' un fattore cruciale. L'algoritmo spesso utilizza una misura basata sulla varianza o sulla distanza tra i punti all'interno di un cluster per decidere come dividerlo. Il criterio di divisione tende a massimizzare la differenza tra i sotto-cluster generati, cercando di ridurre l'intra-cluster variance, ossia la variabilita' dei punti all'interno dello stesso cluster.

Un aspetto importante di DIANA e' che, essendo un algoritmo divisivo, come il clustering gerarchico agglomerativo, non richiede di specificare il numero di cluster in anticipo. Anche in questo caso, e' possibile determinare il numero finale di cluster tagliando il dendrogramma, che rappresenta la struttura gerarchica dei cluster, ad un livello desiderato.

I vantaggi principali di DIANA includono la sua capacita' di produrre una struttura gerarchica di clustering che puo' essere facilmente visualizzata attraverso il dendrogramma. Inoltre, essendo divisivo, DIANA e' generalmente piu' adatto a situazioni in cui e' necessario suddividere un cluster iniziale che contiene punti dissimili o che ha una struttura complessa. Rispetto agli approcci agglomerativi, DIANA puo' essere piu' efficace nel gestire cluster ben separati. Tuttavia, uno degli svantaggi principali di DIANA e' la complessita' computazionale. La divisione dei cluster puo' richiedere un numero significativo di calcoli, specialmente con misure di distanza complesse, e il processo di divisione ricorsiva puo' essere inefficiente su grandi dateset. La sua complessita' e' $O(n^2)$ per ogni divisione e la ricorsivita' aumenta la complessita' complessiva. Di conseguenza, DIANA e' meno scalabile rispetto ad altri algoritmi di clustering, come k-means o AGNES, quando si lavora con dataset molto grandi.

CURE (Clustering Using REpresentatives) e' un algoritmo di clustering gerarchico, progettato per affrontare alcune delle limitazioni degli algoritmi tradizionali come k-means e agglomerativi, in particolare la sensibilita' agli outliers e la difficolta' di gestire cluster di forma arbitraria. CURE e' stato sviluppato per essere piu' robusto rispetto ad altri metodi e per essere efficace in scenari con grandi dataset.

Il funzionamento di CURE si basa su un approccio che utilizza rappresentanti per ogni cluster. Inizialmente, ogni punto e' trattato come un cluster separato. Successivamente, l'algoritmo seleziona un insieme di rappresentanti per ogni cluster, che sono punti rappresentativi che sintetizzano meglio le caratteristiche di ciascun cluster. Per fare cio', l'algoritmo prende i k punti piu' distanti all'interno di ogni cluster e li usa per rappresentare la forma e l'orientamento di quel cluster. Questi rappresentanti vengono poi utilizzati per calcolare la distanza tra i cluster.

Una volta che i cluster sono rappresentati tramite questi punti rappresentativi, l'algoritmo unisce i cluster che sono piu' simili tra loro, come nel caso degli approcci gerarchici. La distanza tra due cluster e' misurata non solo considerando i singoli punti, ma valutando la distanza tra i punti rappresentativi. Questo approccio consente di catturare meglio la forma e l'orientamento dei cluster, superando la limitazione di metodi come k-means che tendono a favorire i cluster di forma sferica.

Una delle caratteristiche distintive di CURE e' la sua capacita' di gestire outliers e cluster di forma complessa. Poiche' l'algoritmo usa rappresentanti che possono essere scelti da qualsiasi punto all'interno del cluster, piuttosto che

semplicemente usare il centroide come in k-means, CURE e' meno sensibile agli outliers. In pratica, gli outliers non influenzeranno significativamente i punti rappresentativi, e quindi non avranno un impatto rilevante sul risultato finale del clustering.

Un altro aspetto importante di CURE e' che, come altri algoritmi gerarchici, non richiede di specificare in anticipo il numero di cluster, ma piuttosto costruisce una struttura gerarchica che puo' essere esplorata tramite un dendrogramma. E' possibile decidere il numero di cluster tagliando il dendrogramma ad un livello desiderato, a seconda della granularita' necessaria.

Per quanto riguarda i vantaggi, CURE e' particolarmente efficace quando si lavora con dati complessi, in cui i cluster non hanno una forma sferica e sono separati in modi non lineari. Grazie all'uso dei rappresentanti, e' piu' robusto rispetto a gli outliers rispetto ad altri algoritmi gerarchici o partizionali. Inoltre, la possibilita' di gestire grandi dataset e' una delle sue forze, grazie al fatto che i rappresentanti sintetizzano informazioni importanti in modo piu' efficiente rispetto al trattamento di ogni punto separatamente. Gli svantaggi principali di CURE riguardano la complessita' computazionale. L'algoritmo richiede il calcolo della distanza tra i punti rappresentativi, e il numero di rappresentanti da scegliere puo' influenzare sia la qualita' del clustering che l'efficienza computazionale. La sua complessita' e' $O(n^2)$, simile a quella degli altri algoritmi di clustering gerarchico, che puo' diventare problematica quando si lavora con data set molto grandi.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) e' un algoritmo di clustering ibrido, con caratteristiche sia agglomerati che Divisive. Da un lato, utilizza un approccio divisivo nella costruzione della CF Tree, suddividendo progressivamente i dati in sottostrutture gerarchiche compatte per gestire grandi volumi di informazioni in modo efficiente. Questa fase riduce la complessita' del clustering tradizionale, evitando il confronto diretto tra tutti i punti. Dall'altro lato, nella fase finale, puo' applicare un algoritmo di class agglomerati per affinare i cluster, attraverso un metodo come k-means o un altro algoritmo gerarchico.

L'idea principale di BIRCH e' quella di costruire una CF Tree (Clustering Feature Tree), una struttura dati che rappresenta una sintesi gerarchica dei dati, riducendo la necessita' di mantenere tutte le istanze in memoria. Ogni nodo della CF Tree contiene informazioni statistiche sui dati aggregati, come il numero di punti, la somma delle coordinate e la somma dei quadrati delle coordinate. Queste informazioni vengono utilizzate per calcolare proprieta' dei cluster, come il loro centroide e la varianza. Durante la costruzione dell'albero, i nuovi punti vengono aggiunti in modo iterativo, e se un cluster diventa troppo grande, il nodo corrispondente viene suddiviso.

Il funzionamento di BIRCH prevede inizialmente la costruzione della CF Tree, in cui i dati vengono inseriti nella struttura gerarchica in modo incrementare, mantenendo solo informazioni aggregate. Successivamente, l'albero viene ridotto eliminando informazioni ridondanti e mantenendo una rappresentazione piu' compatta dei cluster. Una volta ottenuta la struttura complessa, viene applicata un algoritmo di clustering, come k-means, per identificare i cluster finali. Infine, e' possibile affinare ulteriormente i risultati con metodi di clustering aggiuntivi.

Uno dei principali vantaggi di BIRCH e' la sua efficienza e scalabilita', poiche' permette di eseguire il clustering su dataset molto grandi senza dover mantenere in memoria tutti i punti. L'algoritmo lavora in modo incrementali, risultando particolarmente adatto a scenari di streaming di dati o in cui le informazioni arrivano in tempo reale. La capacita' di adattarsi dinamicamente alla struttura dei dati lo rende flessibile, permettendo di identificare cluster senza dover specificare a priori il numero esatto. Tuttavia, BIRCH presenta alcune limitazioni. Funziona meglio con cluster di forma sferica, simili a quelli identificati da k-means, ma puo' avere difficolta' con cluster di forme piu' complesse o con distribuzioni di dati non omogenee. Inoltre, la qualita' dei risultati dipende dalla scelta del parametro di soglia (threshold) nella CF Tree, che determina quando un nodo deve essere suddiviso o meno. Un valore errato puo' portare ad una rappresentazione distorta dei cluster, influenzando negativamente il risultato finale.

Dal punto di vista della complessita' computazionale, BIRCH e' molto piu' efficiente rispetto agli altri algoritmi gerarchici. La costruzione della CF Tree ha complessita' $O(n)$, rendendolo significativamente piu' veloce rispetto a metodi come AGNES, DIANA o CURE, che operano con complessita' $O(n^2)$ o superiore. Anche se il clustering finale dipende dall'algoritmo utilizzato, come k-means che ha complessita' $O(nk)$, la riduzione del numero di punti grazie alla CF Tree permette di gestire dataset molto grandi in modo efficiente.

CLUSTERING BASATO SULLA DENSITÀ

Il clustering basato sulla densita' e' un approccio al clustering che si distingue da altri metodi, come quelli tradizionali o gerarchici, perche' non si concentra su una partizione predefinita dei dati o su una struttura gerarchica fissa. Invece, il class ring basato sulla densita' cerca di identificare cluster come regioni ad alta densita' di punti, separati da regioni a bassa densita'. Questi algoritmi sono particolarmente utili per rilevare cluster di forme arbitrarie, che non sono necessariamente sferici o di dimensioni simili, come spesso accade con metodi come k-means.

Il principio alla base di questi algoritmi e' che i cluster si formano in aree densamente popolate da punti, mentre le regioni a bassa densita' rappresentano i confini tra i cluster e le zone di rumore. Un punto che si trova in una zona di bassa densita' non viene considerato come appartenente a nessun cluster, e spesso viene etichettato come outlier o rumore. U principali algoritmi di clustering basati sulla densita', come DBSCAN, OPTICS, DENCLUE e CLIQUE, identificano i cluster in base alla densita' di punti in una determinata regione. Questi algoritmi non richiedono un numero predefinito di cluster e sono in grado di separare automaticamente i cluster di diverse forme e dimensioni, anche quando questi sono disgiunti o non lineari.

Il funzionamento di un algoritmo di clustering basato sulla densita' inizia selezionando un punto di partenza. successivamente, vengono esaminati tutti i punti vicini a questo punto, entro una distanza definita (chiamata epsilon, o ε). se il numero di punti vicini e' sufficientemente elevato, il punto iniziale viene considerato come parte di un cluster, e l'algoritmo espande il cluster aggiungendo i punti vicini che soddisfano la stessa condizione di densita'. Questo processo continua fino a quando non vengono esaminati tutti i punti vicini e non e' piu' possibile aggiungere nuovi punti al cluster. se un punto non soddisfa il criterio di densita', viene etichettato come rumore. L'algoritmo quindi passa al punto successivo, non ancora visitato e ripete il processo.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) e' un algoritmo di clustering basato sulla densita' che e' in grado di identificare cluster di forma arbitraria, separati da aree a bassa densita'. A differenza di altri algoritmi come k-means, DBSCAN non richiede di specificare a priori il numero di cluster. Inoltre, ha la capacita' di identificare i cosiddetti outlier o rumore, ossia punti che non appartengono a nessun cluster e che vengono ignorati dall'algoritmo.

Il funzionamento di DBSCAN si basa su due parametri principali: la distanza massima tra due punti per essere considerati vicini (epsilon, o ε) e il numero minimo di punti richiesti per formare un cluster (MinPts). L'algoritmo inizia esaminando ogni punto del dataset. Se il punto ha abbastanza punti vicini, cioe' il numero di punti all'interno di una distanza ε e' maggiore o uguale a MinPts, viene considerato un punto centrale di un cluster e l'algoritmo espande il cluster includendo tutti i punti che soddisfano il criterio di densita'. Questo processo di espansione continua fino a quando non vengono aggiunti piu' punti nel cluster. I punti che non soddisfano il criterio di densita', ossia che non hanno abbastanza vicini, vengono etichettati come rumore e non appartengono a nessun cluster. L'algoritmo quindi passa ai punti successivi e ripete il processo fino a quando tutti i punti sono stati esaminati.

Uno dei principali vantaggi di DBSCAN e' la sua capacita' di identificare cluster di forme arbitrarie, anche se non sono sferici, il che lo rende molto piu' flessibile rispetto ad algoritmi come k-means. Inoltre, poiche' DBSCAN non richiede di conoscere il numero di cluster in anticipo, e' particolarmente utile in scenari in cui la struttura dei dati e' sconosciuta. Un altro vantaggio e' che DBSCAN e' in grado di rilevare e separare gli outliers, evitando che questi vengano erroneamente assegnati a cluster. Tuttavia, ci sono anche degli svantaggi. La scelta dei parametri ε e MinPts e' critica per il buon funzionamento dell'algoritmo e, se non vengono selezionati correttamente, DBSCAN puo' produrre risultati insoddisfacenti. Se ε e' troppo grande, potrebbero essere fusi insieme cluster distinti; se e' troppo piccolo, potrebbero esserci troppi punti etichettati come rumore. Inoltre, DBSCAN puo' avere difficolta' a gestire cluster con densita' molto variabili. Se i cluster nel dataset hanno densita' diverse, un singolo valore di epsilon potrebbe non essere adeguato per tutti, portando a errori nel rilevamento dei cluster.

Dal punto di vista della complessita' computazionale, DBSCAN ha una complessita' di O(n*log*n) quando si utilizza una struttura di dati come un albero di ricerca spaziale, come R-tree o KD-tree, per trovare i vicini di ogni punto. Tuttavia, se i dati sono distribuiti in modo denso e non si utilizzano strutture efficienti per il calcolo dei vicini, la complessita' puo' arrivare a $O(n^2)$, dove n e' il numero di punti nel dataset.

CLUSTERING BASATO SU GRIGLIE

Il clustering basato su griglie e' un tipo di algoritmo di clustering che suddivide il dataset in una griglia regolare di celle e utilizza questa struttura per raggruppare i dati in base alla densita' o ad altri criteri, come la vicinanza spaziale. A differenza dei metodi di clustering tradizionali, che operano direttamente sui dati, il clustering basato su griglie cerca di semplificare il processo suddividendo lo spazio in celle e poi raggruppando i dati in queste celle. Ogni cella della griglia rappresenta una regione dello spazio dei dati, e i dati vengono assegnati a queste celle in base alle loro coordinate.

Il funzionamento di un algoritmo di clustering basato su griglie inizia creando una struttura di griglia, che puo' essere una suddivisione regolare dello spazio in celle, ognuna delle quali contiene un certo numero di dati. Successivamente, l'algoritmo calcola la densita' di ciascuna cella, che e' definita come il numero di punti di dati contenuti in quella cella. Se la densita' di una cella e' abbastanza alta, l'algoritmo la considera come un cluster e continua a raggruppare le celle adiacenti con densita' simili. In alcuni casi, possono essere considerate anche le celle adiacenti nella griglia, e se la densita' delle celle adiacenti e' abbastanza alta, queste vengono unite nel cluster.

CLIQUE (CLustering in QUEst) e' un algoritmo che combina concetti di densita' e di griglie. In particolare, si puo' considerare come un algoritmo basato sulla densita', ma che utilizza un approccio a griglie per rappresentare e organizzare i dati. Questo algoritmo identifica i cluster in spazi multimediali senza richiedere una conoscenza preventiva del numero di cluster. Il suo principale obiettivo e' trovare regioni densamente popolate nei dati che possano essere considerate come cluster significativi. CLIQUE e' particolarmente utile per il clustering di dati con un alto numero di dimensioni, dove altri algoritmi di clustering come k-means potrebbero incontrare difficolta' a causa della "curse of dimensionality", ossia la difficolta' nell'individuare cluster significativi in spazi molto ampi.

Il funzionamento di CLIQUE si basa sulla suddivisione dello spazio dei dati in una griglia a piu' dimensioni. Inizialmente, l'algoritmo divide lo spazio in celle di dimensioni predefinite. Poi, per ogni cella, viene calcolata la densita', che e' il numero di punti contenuti in quella cella. Se una cella supera una certa soglia di densita', viene considerata parte di un cluster. CLIQUE non si limita a lavorare su singole dimensioni, ma esplora anche la combinazione di dimensioni per identificare cluster. Questo significa che l'algoritmo cerca combinazioni di celle densamente popolate che formano cluster ad alta densia'. Dopo aver identificato le celle dense, l'algoritmo le unisce per formare i cluster finali.

Uno degli aspetti distintivi di CLIQUE e' la sua capacita' di lavorare con dati ad alta dimensione. Poiche' ogni dimensione del dato viene trattata come una variabile indipendente, l'algoritmo e' in grado di gestire spazi multidimensionali senza la necessita' di ridurre le dimensioni o utilizzare tecniche di proiezione. Inoltre, CLIQUE puo' anche essere efficace nel

trattare cluster di forma arbitraria, a differenza di algoritmi come k-means, che tendono a identificare solo cluster di forma sferica.

I vantaggi di CLIQUE sono evidenti soprattutto nella gestione di dataset ad alta dimensione, dove gli altri algoritmi potrebbero fallire. Non necessita di una definizione preliminare del numero di cluster e puo' identificare cluster anche in spazi complessi e ad alta densita'. Inoltre, essendo basato su una griglia, l'algoritmo e' relativamente veloce rispetto ad altri approcci di clustering gerarchico o basato su densita'. Tuttavia, CLIQUE presenta anche alcuni svantaggi. Poiche' si basa su una griglia, la scelta delle dimensioni delle celle e della densita' minima per definire i cluster e' critica. Se la risoluzione della griglia non e' adeguata, l'algoritmo potrebbe riuscire a identificare correttamente i cluster o, al contrario, potrebbe produrre una grande quantita' di cluster troppo piccoli. Inoltre, sebbene CLIQUE sia in grado di trattare i dati ad alta dimensione, l'efficacia del clustering puo' diminuire se il numero di dimensioni e' troppo elevato, poiche' le celle della griglia potrebbero diventare troppo sparse.

In termini di complessita', CLIQUE ha una complessita' computazionale che dipende principalmente dal numero di dimensioni del dato e dalla dimensione della griglia e puo' essere espressa come $O(n*m^d)$, dove n e' il numero di punti nel dataset, m e' il numero di celle per ciascuna dimensione, che dipende dalla risoluzione della griglia, e d e' il numero di dimensioni del dataset. La complessita' e' relativamente alta per dataset con molte dimensioni, ma rimane gestibile rispetto ad altri algoritmi che cercano di ridurre la dimensionalita' prima di eseguire il clustering.

CLUSTERING BASATO SU MODELLI

Il clustering basato su modelli e' una tipologia di clustering che si concentra sull'assunzione di un modello statistico che descrive i dati. L'idea principale dietro questo approccio e' che i dati siano generati da un insieme di modelli, e l'obiettivo e' identificare questi modelli e utilizzarli per raggruppare i dati in modo piu' significativo. A differenza del clustering partizionale e gerarchico, dove i cluster sono formati in base alla distanza tra i punti, il cluster basato su modelli si concentra sulla distribuzione probabilistica dei dati.

Il funzionamento di un algoritmo di clustering basato su modelli inizia con l'assunzione di un modello probabilistico per ciascun cluster. Questi modelli possono essere diversi a seconda del tipo di dati e delle assunzioni fatte, ma l'idea comune e' che ogni cluster rappresenti una distribuzione di probabilita'. Durante l'addestramento, cerca di determinare quale modello si adatta meglio ai dati per ciascun cluster, aggiungendo i parametri del modello in modo iterativo, fino a trovare una soluzione ottimale che minimizza l'errore o la distanza tra i dati e il modello.

Un esempio molto conosciuto di clustering basato su modelli e' il Self-Organizing Map (SOM). SOM e' una rete neurale non supervisionata che proietta i dati ad alta dimensione in una griglia bidimensionale, cercando di rappresentare la topologia e le relazioni tra i dati in modo piu' comprensibile. Durante l'apprendimento, i dati vengono mappati sui nodi della griglia, con i nodi vicini che rappresentano dati simili. SOM e' utile per visualizzare dati complessi e per la riduzione della dimensionalita'.

Un altro esempio di clustering basato sui modelli e' il Gaussian Mixture Model (GMM). GMM e' un modello probabilistico che assume che i dati siano generati da una combinazione di distribuzioni gaussiane. Ogni cluster e' rappresentato da una distribuzione gaussiana con un proprio insieme di parametri, come la media e la varianza. L'algoritmo di clustering cerca di trovare la combinazione di gaussiane che meglio rappresenta i dati attraverso un processo iterativo, tipicamente utilizzando l'algoritmo Expectation-Maximization (EM). L'EM alterna due fasi: nella fase di Expectation (E) stima i valori mancanti, cioe' le variabili latenti, basandosi sui dati osservati e sul modello corrente; invece, nella fase di Maximization (M), ottimizza i parametri del modello massimizzando la verosimiglianza. Il GMM e' particolarmente utile per modellare cluster che non sono necessariamente sferici o che hanno forme complesse.

Un altro algoritmo di clustering basato su modelli e' COBWEB, che utilizza un approccio gerarchico per creare un albero di concetti. Si basa sull'assunzione che i dati possiedano una struttura gerarchica e tenta di dividere i dati in concetti, apprendendo via via le caratteristiche e le relazioni tra gli oggetti. COBWEB e' un tipo di algoritmo che si concentra sul clustering basato su attributi e valori, ed e' utile in contesti dove la categorizzazione dei dati e' di natura piu' concettuale.

Il vantaggio principale del clustering basato su modelli e' la sua capacita' di gestire distribuzioni di dati complesse, in particolare quando i cluster non sono ben separati o non seguono forme geometriche semplici. Inoltre, l'approccio probabilistico fornisce una misura di incertezza, utile in molte applicazioni. Tuttavia, uno degli svantaggi principali e' che il modello presupposto potrebbe non essere adeguato per tutti i tipi di dati, portando a risultati imprecisi o poco significativi. Inoltre, la scelta del modello e la sua stima richiedono calcoli complessi e sono sensibili ai parametri iniziali, il che puo' influenzare l'efficacia dell'algoritmo.

TIPI DI GRAFI

Nel contesto del clustering, i grafi vengono utilizzati come una struttura per rappresentare le relazioni tra gli oggetti di un dataset. Esistono diversi tipi di grafiche vengono utilizzati per applicare tecniche di clustering:

1. Grafo arbitrario: ha connessioni qualsiasi;
2. Grafo Clique: ha nodi tutti connessi tra loro;
3. Grafo delle distanze: connette nodi in base a distanze o somiglianze.

Ognuno di questi ha caratteristiche specifiche che li rendono adatti a determinati tipi di dati e approcci di clustering.

Un grafo arbitrario e' un tipo di grafo dove le connessioni tra i nodi sono generali e non seguono lo schema predefinito. O nodi possono essere connessi in base a qualsiasi criterio, senza vincoli particolari sulla distanza o sulla simmetria. In questo tipo di grafo le relazioni tra i nodi possono essere complesse, e il clustering avviene cercando sottoinsiemi di nodi che siano piu' simili tra loro in termini di connettivita', ma senza una struttura geometrica precisa.

Un grafo Clique e' un grafo in cui ogni nodo e' connesso a tutti gli altri nodi all'interno di un sottoinsieme. In altre parole, un clique e' una parte del grafo in cui ogni possibile connessione tra i nodi e' presente. Questo tipo di grafo e' utile quando si vogliono identificare gruppi di nodi strettamente interconnessi. Il problema di trovare un Clique in un grafo puo' essere complicato, soprattutto quando il grafo e' corrotto o presenta errori nelle connessioni. In questi casi, il Corrupted Clique Problem diventa rilevante, poiche' il grafo potrebbe non mostrare tutte le connessioni corrette, ma il gruppo di nodi che formano il clique potrebbe ancora essere identificabile con tecniche adeguate. Un esempio di algoritmo utilizzato per risolvere questo tipo di problema e' il CAST (Clique-Aided Subgraph Traversal), che esplora il grafo e individua il clique nonostante le corruzioni nelle connessioni.

Un grafo delle distanze e' un grafo basato su misure di distanza tra i nodi, come la distanza euclidea o altri metriche, per determinare la vicinanza tra gli oggetti. Nei grafi delle distanze, i nodi sono connessi in base alla loro similarita' o prossimita' in uno spazio multidimensionale. La costruzione di un grafo delle distanze e' particolarmente utile quando i dati sono rappresentati in uno spazio geometrico, e l'obiettivo e' raggruppare gli oggetti che sono piu' vicini tra loro.

Esistono anche tecniche che permettono di passare da un tipo di grafo ad un altro, come il passaggio da un grafo arbitrario ad un grafo clique o ad un grafo delle distanze. Questi passaggi sono utili quando si desidera esplorare un grafo da diverse angolazioni, a seconda degli obiettivi del clustering. Ad esempio, si puo' partire da un grafo arbitrario per identificare delle connessioni generali e successivamente costruire un grafo Clique per trovare i gruppi di nodi completamente interconnessi.

MICROARRAY

I microarray sono dispositivi utilizzati in biologia molecolare per misurare l'espressione genica di migliaia di geni simultaneamente. Si basano sulla tecnologia di ibridazione per rilevare la presenza di specifici trascritti di RNA in un campione biologico. In pratica, vengono utilizzati per osservare in che misura i geni sono attivi in diverse condizioni, consentendo di raccogliere una grande quantita' di dati che descrivono come i geni rispondono a vari stimoli o malattie.

Nel contesto del clustering, i microarray vengono utilizzati per raggruppare i geni in base ai loro modelli di espressione. Poiche' ogni gene ha un'espressione diversa in vari momenti o in risposta a diversi trattamenti, il clustering permette di identificare gruppi di geni che mostrano comportamenti simili, cosa che puo' portare a scoprire i geni che lavorano insieme in processi biologici simili. Ad esempio, in un esperimento su batteri, i micro aerei possono rivelare gruppi di geni che sono espressi in risposta a specifiche condizioni ambientali, come la presenza di un antibiotico, aiutando a capire come i batteri si adattino a questi cambiamenti.

Un'applicazione comune dei microarray nel clustering e' nello studio dell'espressione genica. I ricercatori possono identificare sottogruppi di geni che sono associati con specifici tratti fenotipici, come la crescita tumorale o la risposta del sistema immunitario, con l'obiettivo di comprendere meglio i meccanismi biologici sottostanti. Il clustering, in questo caso, aiuta a suddividere i dati in gruppi che mostrano tendenze simili di espressione, facilitando l'interpretazione dei dati biologici.

I vantaggi principali dei microarray nel clustering sono la capacita' di raccogliere ed analizzare una grande quantita' di dati in tempi relativamente brevi, permettendo di ottenere insight sui meccanismi biologici complessi. Tuttavia, presentano anche alcuni svantaggi. La qualita' dei risultati dipende fortemente dalla qualita' del campione e dalla precisione delle misurazioni. Inoltre, la gestione di un numero elevato di dati puo' diventare difficile senza algoritmi di clustering adeguati, che possano ridurre la dimensionalita' e trovare i gruppi significativi in modo efficace. La complessita' computazionale di questi algoritmi di clustering e' elevata soprattutto con grandi dataset, come quelli generati dai microarray. La gestione e l'analisi dei dati puo' richiedere un tempo significativo e un potere computazionale elevato. In termini numerici, la complessita' del clustering puo' variare a seconda dell'algoritmo scelto: ad esempio, l'algoritmo k-means ha una complessita' di O(nkd), dove n e' il numero di punti, k e' il numero di cluster e d e' la dimensionalita' dei dati, cioe' il numero di geni misurati. Se il numero di geni e' molto alto, la complessita' puo' diventare un fattore limitante.

VALUTAZIONE DEL CLUSTERING

La valutazione del clustering e' un aspetto cruciale per determinare la qualita' e l'affidabilita' dei gruppi individuati nei dati. In ambito biologico, il clustering viene spesso applicato all'analisi dell'espressione genica, e per valutarne i risultati si usano metriche sia interne che esterne. Le metriche interne si basano sulla coesione e separazione dei cluster, mentre quelle esterne confrontano il clustering ottenuto con una partizione di riferimento.

La Gene Ontology (GO) e' un sistema strutturato per la classificazione e l'annotazione funzionale dei geni, organizzato in una gerarchia ad albero che permette di descrivere le loro proprieta' biologiche in modo standardizzato. GO e' suddivisa in tre categorie principali:

1. Funzione molecolare: descrive le attivita' biochimiche di un gene o di una proteina;
2. Processo biologico: indica i meccanismi biologici in cui il gene e' coinvolto;
3. Componente cellulare: specifica la localizzazione del gene o della proteina all'interno della cellula.

Il funzionamento della Gene Ontology si basa su un vocabolario controllato e su relazioni gerarchiche tra termini, dove i concetti piu' generali si suddividono in termini piu' specifici. L'annotazione genica utilizza identificatori GO associati a dati sperimentali o predetti computazionalmente, fornendo un quadro funzionale che facilita l'interpretazione di dati omici come l'espressione genica.

Uno dei principali vantaggi della Gene Ontology e' la sua capacita' di standardizzare e integrare informazioni biologiche, consentendo il confronto tra dati provenienti da organismi diversi e facilitando l'analisi funzionale in studi di genomica, trascrittomica e proteomica. Inoltre, permette di effettuare analisi di arricchimento funzionale, individuando gruppi di geni con funzioni biologiche comuni, utile per interpretare i risultati di esperimenti di clustering o di espressione genica. Per determinare se un insieme di geni e' significativamente arricchito per una particolare categoria GO, si utilizza un p-value, ottenuto tipicamente tramite test di ipotesi come il test ipergeometrico o il test di Fisher. Tuttavia, poiche' vengono eseguiti numerosi test simultaneamente, e' necessaria una correzione per il multiple testing. La GO presenta anche alcuni svantaggi, tra cui la dipendenza dalla qualita' e completezza delle annotazioni, che possono essere incomplete o basate su predizioni bioinformatiche non sempre accurate. Inoltre, la GO e' una struttura in continua evoluzione, e i dati di annotazione possono cambiare nel tempo, rendendo necessario l'aggiornamento costante degli studi basati su di essa.

Dal punto di vista computazionale, l'analisi della Gene Ontology puo' essere complessa. La ricerca e il recupero di informazioni dai database GO hanno una complessita' che puo' variare da $O(n \log n)$ a $O(n^2)$ a seconda delle operazioni svolte, mentre l'analisi di arricchimento funzionale, se eseguita sudata set molto grandi, puo' richiedere un tempo computazionale significativo, soprattutto se si utilizzano approcci di permutazione o bootstrap per la significativita' statistica.

Nella Gene Ontology e nella valutazione del clustering, il Gibbs Sampling e' un algoritmo di campionamento stocastico utilizzato per stimare distribuzioni di probabilita' complesse quando il calcolo diretto e' impraticabile. Funziona generando campioni iterativi da distribuzioni condizionali parziali, aggiornando una variabile alla volta mentre le altre rimangono fisse, fino a convergere alla distribuzione desiderata.

Il Gibbs Sampling viene applicato per inferire l'associazione di geni con categorie GO quando le annotazioni disponibili sono incomplete o rumorose. Ad esempio, puo' essere usato per identificare gruppi di geni funzionalmente simili, stimando in modo probabilistico la loro appartenenza a cluster biologicamente significativi sulla base di dati di espressione genica. Inoltre, nel contesto dell'arricchimento funzionale, puo' migliorare la stima del p-value, riducendo la dipendenza da ipotesi parametriche rigide e gestendo meglio incertezze nei dati.

Nella Gene Ontology e nella valutazione del clustering, le Reti Bayesiane (Bayesian Networks, BN) sono modelli probabilistici che rappresentano le relazioni di dipendenza condizionale tra variabili attraverso un grafo aciclico diretto. Ogni nodo corrisponde ad una variabile, mentre gli archi indicano dipendenze probabilistiche, con ogni nodo caratterizzato da una distribuzione di probabilita' condizionale basata sui suoi genitori nel grafo.

Le Reti Bayesiane vengono utilizzate per modellare le relazioni tra geni e le loro funzioni, inferendo categorie funzionali mancanti o identificando pathway biologici regolati in modo congiunto. Possono essere impiegati per migliorare l'arricchimento funzionale, stimando in maniera piu' robusta la probabilita' che un cluster di geni sia associato ad una specifica funzione GO. Inoltre, offrendo un quadro probabilistico utile per integrare dati sperimentali e eterogenei e ridurre il rumore nei dati di espressione giudica.

Nella Gene Ontology e nella valutazione del clustering, le Reti di Markov (Markov Networks o Markov Random Fields, MRF), sono un'altra classe di modelli probabilistici grafici, ma a differenza delle BN, utilizzano grafi non diretti per rappresentare le dipendenze tra variabili. In questo caso, ogni nodo rappresenta una variabile e gli archi indicano relazioni di dipendenza senza una direzione specifica, modellate tramite funzioni di potenziale definite sui clique del grafo.

Le Reti di Markov vengono utilizzate per modellare dipendenze locali tra geni, specialmente in dati di espressione genica o interazioni proteina-proteina. Sono particolarmente utili per propagare informazioni tra geni correlati, migliorando l'accuratezza delle assegnazioni funzionali nei cluster. Inoltre, possono essere utilizzate in metodi di campionamento stocastico come il Gibbs Sampling, per stimare la probabilita' di appartenenza di un gene ad una categoria GO in base alle informazioni dei suoi vicini nel grafo.

CO-CLUSTERING

Il co-clustering, noto anche come bi-clustering, e' una tecnica avanzata di clustering che raggruppa simultaneamente le righe e le colonne di una matrice di dati, piuttosto che solo le righe (come nei metodi tradizionali). Questa tecnica e' particolarmente utile per dati ad alta dimensionalita', come quelli presenti in bioinformatica, analisi testuale e raccomandazioni.

Il funzionamento del co-clustering si basa sull'idea che sottogruppi di elementi nelle righe siano correlati solo con sottogruppi specifici nelle colonne, e viceversa.ad esempio, nell'analisi dell'espressione genica, il co-clustering puo' raggruppare i geni che mostrano comportamenti simili solo in determinate condizioni sperimentali, anziche' in tutto il dataset. I metodi per eseguire il co-clustering includono approcci basati su matrici, tecniche probabilistiche come i modelli generativo, e metodi basati su grafi, tra cui quelli che utilizzano la decomposizione a singoli valori (SVD) o algoritmi di ottimizzazione basati su mutua informazione.

Uno dei principali vantaggi del co-clustering la capacita' di identificare strutture locali nei dati, migliorando la qualita' dell'analisi rispetto al clustering tradizionale. Permette inoltre di lavorare in spazi ad alta dimensionalita' senza risentire della "maledizione della dimensionalita'", perche' considera simultaneamente le relazioni tra righe e colonne.questo lo rende particolarmente efficace in scenari come il text mining, dove puo' identificare gruppi di documenti correlati a gruppi specifici di parole chiave, o nella bioinformatica, dove aiuta a trovare gruppi di geni attivi in determinate condizioni sperimentali. Tuttavia, presenta anche svantaggi. La sua implementazione e' spesso piu' complessa rispetto ai metodi di clustering tradizionali, sia dal punto di vista computazionale che nell'interpretazione dei risultati. Inoltre, alcuni algoritmi di co-clustering richiedono l'impostazione di parametri specifici, come il numero di cluster in entrambe le dimensioni, il che puo' essere difficile da determinare in anticipo.

Dal punto di vista della complessita' computazionale, il costo dipende dall'algoritmo utilizzato. Metodi basati su decomposizione matriciale, come quelli che sfruttano la SVD, hanno una complessita' O(mn min(m,n)), mentre quelli basati su modelli probabilistici possono essere piu' costose, con tempi di esecuzione che dipendono dal numero di iterazioni necessarie per convergere. Tuttavia, tecniche piu' efficienti come l'information-theoretic co-clustering riducono la complessita' sfruttando principi di mutua informazione, risultando spesso piu' scalabili.

CO-CLUSTERING CON VINCOLI

Il co-clustering con vincoli e' una variante del co-clustering in cui vengono imposte delle restrizioni sui gruppi formati, al fine di guidare il processo di apprendimento in base a informazioni aggiuntive. Questo approccio e' utile in scenari in cui si dispone di conoscenze preliminari sui dati, come vincoli sulle relazioni tra righe o colonne della matrice, che possono migliorare la qualita' del clustering e rendere i risultati piu' interpretabili.

Il funzionamento del co-clustering con vincoli e' simile al co-clustering tradizionale, ma con l'aggiunta di restrizioni che possono essere di tipo must-link (due elementi devono essere nello stesso cluster) o cannot-link (due elementi non possono appartenere allo stesso cluster). Questi vincoli possono essere applicati alle righe, alle colonne o a entrambe. Ad esempio, nell'analisi dell'espressione genica, se si sa che due geni devono comportarsi in modo simile in determinate condizioni sperimentali, si puo' imporre un vincolo must-link affinche' siano raggruppati insieme.

I vantaggi del co-clustering con vincoli includono una maggiore accuratezza rispetto al co-clustering non vincolato, soprattutto quando si dispone di informazioni a priori affidabili. Inoltre, aiuta a ridurre l'ambiguita' nei dati e a migliorare la stabilita' di risultati, riducendo la sensibilita' agli outliers o a rumore nei dati. E' particolarmente utile in applicazioni dove il clustering non supervisionato potrebbe portare a risultati poco coerenti con le conoscenze esistenti, come nella bioinformatica e nell'analisi di testi. Tuttavia, presenta anche svantaggi, tra cui una maggiore complessita' computazionale, poiche' l'algoritmo deve soddisfare i vincoli imposti oltre a trovare la migliore partizione dei dati.inoltre, se i vincoli sono errati o inconsistenti, possono portare a risultati distorti o poco significativi.

Dal punto di vista della complessita' computazionale, il costo dipende dal metodo specifico utilizzato. Se i vincoli vengono gestiti attraverso un'ottimizzazione vincolata nei metodi basati su matrici o probabilistici, la complessita' puo' essere superiore a quella del co-clustering standard, aumentando da O(mn min(m,n)) ha un valore piu' elevato a seconda del numero di vincoli imposti e del metodo di risoluzione adottato.

ALBERI FILOGENETICI

RICOSTRUZIONE DELL'ALBERO EVOLUTIVO

La ricostruzione di un albero evolutivo e' il processo attraverso il quale si cerca di inferire le relazioni filogenetiche tra specie o sequenze genetiche a partire da dati biologici. Esistono due approcci principali: la filogenesi basata sulla distanza e la filogenesi basata su caratteri.

La ricostruzione dell'albero evolutivo e' il processo mediante il quale gli scienziati determinano le relazioni tra le specie, individuando i loro antenati comuni. Questo concetto si basa sulla teoria dell'evoluzione di Charles Darwin, secondo cui tutte le specie derivano da un antenato comune attraverso un processo di discendenza con modificazioni.

Uno degli esempi piu' emblematici di questa ricerca e' il caso del panda gigante, la cui classificazione e' stata a lungo incerta. Non era chiaro se appartenesse alla famiglia degli orsi o a quella dei procioni. Nel 1985, Stephen O'Brien e i suoi colleghi risolsero il dilemma grazie all'analisi del DNA, dimostrando che il panda gigante e' un orso.

L'idea di utilizzare il DNA per ricostruire gli alberi evolutivi risale a circa quarant'anni fa, quando Emile Zuckerkandl e Linus Pauling posero le basi per l'approccio basato sulle sequenze genetiche. Tuttavia, l'utilizzo del materiale genetico per tracciare le linee evolutive fu inizialmente molto dibattuto. Un famoso confronto si ebbe tra Zuckerkandl e George Gaylord Simpson: nel 1963, Zuckerkandl suggeri che la struttura dell'emoglobina potesse indicare una continuita' tra gorilla ed esseri umani, ma Simpson, nel 1964, contesto' questa ipotesi, sostenendo che l'emoglobina non fosse un indicatore affidabile dell'affinita' evolutiva. Questo dibattito mise in evidenza la necessita' di metodi piu' oggettivi.

L'introduzione della genetica molecolare e delle tecniche di sequenziamento del DNA ha permesso di superare i limiti delle osservazioni morfologiche, fornendo strumenti piu' precisi per la ricostruzione degli alberi evolutivi. Oggi, l'analisi del DNA e' una pratica consolidata che ha rivoluzionato la comprensione delle relazioni tra le specie.

Le due principali teorie sull'evoluzione umana, Out of Africa e l'ipotesi multiregionale, propongono visioni diverse sulla diffusione degli esseri umani moderni.

L'ipotesi Out of Africa sostiene che l'uomo moderno sia emerso in Africa circa 150.000 anni fa e da li si sia diffuso nel resto del mondo, sostituendo altre popolazioni umanoidi senza discendere direttamente dai Neanderthal. Questa teoria e' supportata dal fatto che le popolazioni africane mostrano la maggiore diversita' genetica, segno di una lunga evoluzione prima della migrazione globale. Le analisi del DNA mitocondriale (mtDNA) e i microsatelliti confermano questa separazione tra africani e non africani, rafforzando l'idea di un'origine africana dell'umanita'.

L'ipotesi multiregionale, invece, afferma che l'evoluzione umana sia avvenuta negli ultimi 2 milioni di anni in diverse aree geografiche, con tratti moderni sviluppatisi localmente. Secondo questa visione, le migrazioni fuori dall'Africa avrebbero portato a un continuo incrocio genetico con altre popolazioni, come i Neanderthal in Europa e Asia, senza una sostituzione completa. Questo spiegherebbe le variazioni regionali nelle popolazioni moderne.

Entrambe le ipotesi cercano di spiegare l'evoluzione dell'uomo moderno, ma le prove genetiche favoriscono l'ipotesi Out of Africa, suggerendo un'origine africana e una successiva diffusione globale. Tuttavia, l'ipotesi multiregionale evidenzia l'importanza delle interazioni tra le diverse popolazioni, dimostrando che l'evoluzione umana e' stata un processo complesso e interconnesso.

FILOGENESI BASATA SULLA DISTANZA

La filogenesi basata sulla distanza e' un metodo utilizzato per ricostruire le relazioni evolutive tra organismi o sequenze genomiche. Un esempio notevole di questa tecnica e' l'analisi filogenetica del virus HIV. Nel 1994, un caso giudiziario a Lafayette, Louisiana, coinvolse un medico accusato di aver iniettato alla sua ex-compagna sangue infetto da HIV. Poiche' il virus ha un alto tasso di mutazione, gli scienziati poterono confrontare le sequenze genetiche dell'HIV della donna, del medico e di altri pazienti. L'analisi mostro' che il virus nella donna derivava direttamente da un paziente HIV positivo, confermando la trasmissione. Questo fu il primo caso in cui la filogenesi fu utilizzata in tribunale. Un altro esempio riguarda l'evoluzione delle ali negli insetti stecco. Nel 2003, uno studio confronto' insetti stecco alati e privi di ali per capire se le ali si fossero evolute piu' volte. Analizzando il DNA e la morfologia, si ipotizzo' che gli antenati di questi insetti fossero privi di ali e che le ali si fossero sviluppate indipendentemente in piu' occasioni.

L'albero filogenetico rappresenta graficamente le relazioni evolutive tra specie. Le foglie dell'albero corrispondono alle specie attuali, mentre i nodi interni rappresentano gli antenati comuni. Un albero puo' essere radicato, cioe' con un antenato comune noto, o non radicato, senza una posizione definita per l'antenato comune. Ogni ramo puo' indicare il numero di mutazioni avvenute o il tempo trascorso nell'evoluzione. Per costruire un albero filogenetico, si utilizza una matrice delle distanze che riporta la distanza evolutiva tra le specie. Se la matrice e' additiva, esiste un albero che rispecchia le distanze evolutive, e il problema puo' essere risolto con algoritmi specifici. Se non e' additiva, significa che non esiste un albero che soddisfi esattamente tutte le distanze. Metodi come Neighbor-Joining e UPGMA appartengono alla filogenesi basata sulla distanza.

L'algoritmo Neighbor Joining (NJ) serve a costruire alberi filogenetici utilizzando le distanze tra le specie. Il suo obiettivo e' trovare il modo migliore per collegare le specie in un albero che rappresenti le loro relazioni evolutive. NJ e' adatto per l'analisi di dati additivi e non additivi, quindi e' un algoritmo molto versatile, anche se con dati non perfettamente additivi la sua accuratezza diminuisce.

Il processo inizia con una matrice delle distanze, che indica quanto sono diverse tra loro le specie analizzate. L'algoritmo cerca, a ogni passo, la coppia di specie che e' piu' simile, ossia con la distanza piu' bassa. Queste due specie vengono unite

in un nuovo nodo che rappresenta il loro antenato comune. A questo punto, l'algoritmo aggiorna la matrice delle distanze, calcolando una nuova distanza tra il nodo appena creato e tutte le altre specie. Il processo si ripete fino a quando tutte le specie sono collegate in un unico albero.

Uno dei vantaggi di NJ e' la sua velocita': riesce a costruire alberi filogenetici rapidamente, anche quando si analizzano molte specie. Ha una complessita' computazionale di $O(n^3)$, quindi e' molto piu' efficiente di altri metodi piu' dettagliati ma piu' lenti. Inoltre, tende a fornire alberi abbastanza accurati, specialmente se i dati rispettano certe condizioni. Tuttavia, ha anche dei limiti. NJ utilizza solo le distanze tra le specie e non direttamente i dati genetici, quindi eventuali errori nelle distanze possono portare a risultati meno precisi. Inoltre, essendo un metodo basato su regole fisse, non sempre trova l'albero migliore possibile.

NJ funziona particolarmente bene quando la matrice delle distanze e' additiva, cioe' quando esiste un albero che rappresenta perfettamente le distanze evolutive tra le specie. In questo caso, l'algoritmo e' in grado di ricostruire esattamente l'albero corretto. Se invece la matrice non e' additiva, a causa di rumore evolutivo o variazioni nei tassi di mutazione, NJ fornira' comunque un albero, ma questo potrebbe non riflettere con precisione la vera storia evolutiva.

UPGMA (Unweighted Pair Group Method with Arithmetic Mean) e' un metodo di filogenesi additiva, poiche' assume che le distanze tra le specie possano essere rappresentate come la somma delle lunghezze dei rami di un albero ultrametrico. Questo significa che l'evoluzione avviene a un tasso costante per tutte le specie, un'ipotesi nota come orologio molecolare.

L'algoritmo funziona partendo da una matrice di distanze tra le specie. A ogni passo, individua la coppia con la distanza minore e le unisce in un nuovo nodo, posizionandolo a meta' strada tra i due elementi. La matrice viene aggiornata calcolando la nuova distanza tra questo gruppo e le altre specie come la media aritmetica delle distanze originali. Il processo si ripete fino a quando tutte le specie vengono raggruppate in un unico albero radicato.

Uno dei principali vantaggi di UPGMA e' la sua semplicita' e rapidita' di esecuzione, con una complessita' di $O(n_2)$, che lo rende piu' efficiente rispetto ad altri metodi piu' sofisticati. Tuttavia, presenta un limite importante: l'accuratezza dell'albero dipende dall'ipotesi che l'evoluzione avvenga a velocita' costante. Se questo presupposto non e' valido e le specie si sono evolute con tassi diversi, l'albero risultante potrebbe essere distorto. Inoltre, e' sensibile agli errori nelle distanze, che possono portare a raggruppamenti errati.

Nonostante questi limiti, UPGMA e' utile in contesti in cui l'ipotesi di orologio molecolare e' ragionevole, come in alcuni studi su virus o batteri con tassi di mutazione relativamente costanti. In scenari piu' complessi, metodi come Neighbor-Joining, che non impongono un tasso di evoluzione uniforme, possono offrire risultati piu' accurati.

Dunque, UPGMA assume un tasso di evoluzione costante (orologio molecolare) ed e' un metodo di filogenesi additiva, poiche' costruisce un albero coerente con le distanze osservate. Neighbor-Joining, invece, e' piu' flessibile perche' non assume un tasso di evoluzione costante, ed e' adatto anche a situazioni in cui l'additivita' non e' perfetta.

La filogenesi additiva e' un metodo per costruire alberi filogenetici basati su matrici di distanze tra specie, assumendo che queste distanze siano coerenti con una struttura ad albero. In altre parole, la distanza tra due specie deve poter essere ricostruita sommando le lunghezze dei rami che le collegano, senza discrepanze. Questo approccio permette di ottenere un albero evolutivo accurato, in cui le relazioni tra le specie sono rappresentate senza contraddizioni.

Per garantire che una matrice di distanze sia adatta alla costruzione di un albero filogenetico, si utilizza la condizione dei quattro punti. Questa condizione stabilisce che, per ogni quartetto di specie (i, j, k, l), la matrice e' additiva se tra le tre possibili somme di coppie di distanze, due sono uguali e la terza e' inferiore alle altre due. Se questa condizione e' soddisfatta per tutti i quartetti della matrice, significa che le distanze possono essere rappresentate correttamente in una struttura ad albero, senza incongruenze nei dati.

Un concetto fondamentale nella costruzione dell'albero e' quello dei tripli degeneri. Un triplo degenerato si verifica quando tre specie, indicate come i, j e k, soddisfano una relazione specifica: la somma delle distanze tra i e j e tra j e k e' uguale alla distanza tra i e k. Questo significa che la specie j si trova esattamente lungo il percorso evolutivo tra i e k o e' collegata a esso con un bordo di lunghezza zero. La presenza di tripli degeneri nella matrice delle distanze e' particolarmente utile perche' permette di semplificare il problema della costruzione dell'albero, riducendo progressivamente la matrice e facilitando l'individuazione delle connessioni evolutive.

L'algoritmo di filogenesi additiva si basa su un processo iterativo per costruire l'albero filogenetico a partire dalla matrice delle distanze. Se la matrice ha dimensione 2x2, l'algoritmo restituisce un albero con un unico ramo, la cui lunghezza corrisponde alla distanza tra le due specie. Se invece la matrice e' piu' grande, il primo passo e' verificare se e' additiva, ovvero se soddisfa la condizione dei quattro punti. Una volta confermata l'additivita', si procede con l'identificazione di un triplo degenerato. Quando un triplo degenerato viene trovato, la specie j viene temporaneamente rimossa dalla matrice, riducendo la sua dimensione. Questo processo viene ripetuto ricorsivamente fino a ottenere una matrice 2x2. A quel punto, l'albero viene ricostruito progressivamente, reinserendo le specie rimosse come nodi con un bordo di lunghezza zero. Se inizialmente non si trovano tripli degeneri, si procede riducendo le lunghezze degli spigoli delle foglie dell'albero fino a farne emergere uno, consentendo di proseguire la costruzione.

L'algoritmo di filogenesi additiva si rivela particolarmente utile quando le distanze evolutive tra le specie sono ben definite e si vuole ottenere una rappresentazione chiara e coerente delle loro relazioni in una struttura ad albero. Grazie alla verifica della condizione dei quattro punti e all'uso dei tripli degeneri, questo metodo permette di costruire

alberi filogenetici in modo efficiente, riducendo progressivamente la complessita' del problema e garantendo un risultato privo di contraddizioni.

Quando le distanze tra le specie non seguono perfettamente un modello additivo, si parla di filogenesi non additiva. Un esempio di metodo usato in questo contesto e' la filogenesi della distanza dei minimi quadrati, che cerca di trovare l'albero che minimizza la discrepanza tra le distanze osservate e quelle previste dall'albero stesso. Questo problema e' computazionalmente difficile e spesso si usano metodi euristici per risolverlo.

La filogenesi non additiva e' un approccio per costruire alberi filogenetici che non fa riferimento alla condizione di additivita' delle distanze tra le specie, a differenza della filogenesi additiva. In altre parole, in questo tipo di filogenesi le distanze evolutive tra le specie possono essere rappresentate da matrici che non soddisfano la condizione di additivita', cioe' la somma delle distanze tra i vari nodi lungo l'albero non deve necessariamente corrispondere alla distanza diretta tra le specie. Questo approccio consente di trattare situazioni evolutive piu' complesse, in cui le distanze tra le specie non seguono una struttura ad albero semplice, come puo' avvenire in presenza di eventi evolutivi complicati, come il reticolato evolutivo o la convergenza evolutiva.

In filogenesi non additiva, l'albero filogenetico viene ricostruito utilizzando algoritmi che si basano sulle distanze tra le specie, ma senza la necessita' di farle soddisfare la condizione di additivita'. Questi algoritmi, infatti, sono in grado di gestire situazioni in cui le distanze evolutive non sono additive e dove le relazioni tra le specie potrebbero essere influenzate da fattori esterni, come eventi di ibridazione, trasferimento genico orizzontale o altre dinamiche evolutive non lineari.

Un esempio di approccio per la filogenesi non additiva e' l'uso di algoritmi basati sulla minimizzazione delle disuguaglianze di distanza, come quelli che impiegano metodi di alberi reticolati o matrici di dissimilarita' che non soddisfano la condizione dei quattro punti. In questi casi, gli algoritmi ricostruiscono l'albero cercando di minimizzare le discrepanze tra le distanze osservate e quelle previste dal modello evolutivo, ma senza la restrizione dell'additivita'.

L'algoritmo di ricostruzione in filogenesi non additiva si differenzia da quello della filogenesi additiva per il fatto che non assume che le distanze tra le specie siano compatibili con una struttura ad albero semplice. Invece, si concentra sull'analisi e sulla minimizzazione delle incongruenze tra le distanze evolutive, cercando di rappresentare le relazioni tra le specie nel miglior modo possibile, anche se questo richiede l'uso di modelli piu' complessi che ammettono la presenza di reticoli evolutivi o altre strutture non ad albero.

Poiche' la filogenesi non additiva non richiede la condizione dei quattro punti, essa puo' essere utilizzata in contesti evolutivi piu' complessi dove le relazioni tra le specie non possono essere descritte in modo lineare o additivo. Ad esempio, quando si studiano specie che sono state coinvolte in eventi evolutivi complessi, come la speciazione rapida o l'ibridazione tra popolazioni distinte, le distanze tra le specie potrebbero non seguire una struttura ad albero classica. In questi casi, l'approccio non additivo permette una rappresentazione piu' flessibile delle relazioni evolutive.

La filogenesi della distanza dei minimi quadrati e' un approccio per costruire alberi filogenetici a partire da una matrice di distanze che non segue il principio di additivita', ovvero le distanze evolutive tra le specie non possono essere rappresentate semplicemente come la somma delle lunghezze dei rami di un albero filogenetico. In questo contesto, l'obiettivo e' trovare un albero che migliori l'approssimazione delle distanze tra le specie, riducendo l'errore tra le distanze osservate e quelle previste dall'albero stesso.

Per fare cio', si utilizza la tecnica dei minimi quadrati, che cerca di minimizzare la somma degli errori quadrati tra le distanze osservate nella matrice e quelle calcolate nel modello ad albero. L'errore quadratico viene calcolato come la somma delle differenze al quadrato tra le distanze tra tutte le coppie di specie. L'obiettivo e' trovare un albero che minimizzi questa somma, ossia che migliori la corrispondenza tra le distanze nella matrice e quelle previste dalla struttura ad albero.

Tuttavia, il problema di costruire un albero filogenetico che minimizzi l'errore quadratico e' estremamente complesso e viene classificato come NP-difficile, il che significa che non esistono algoritmi in grado di risolverlo in tempo polinomiale per tutte le possibili dimensioni della matrice. La difficolta' deriva dal fatto che il numero di alberi possibili cresce esponenzialmente con l'aumentare del numero di specie, rendendo la ricerca della soluzione ottimale una sfida computazionale. Per affrontare questa difficolta', vengono utilizzati algoritmi euristici e approssimativi (come il simulated annealing o gli algoritmi genetici), che cercano di esplorare lo spazio delle soluzioni in modo intelligente per trovare una soluzione valida in tempi ragionevoli, anche se non sempre ottimale.

Il simulated annealing esplora soluzioni ampie all'inizio e poi si concentra progressivamente su soluzioni piu' specifiche, cercando di evitare soluzioni subottimali. Gli algoritmi genetici si ispirano invece al principio di evoluzione naturale, evolvendo un gruppo di soluzioni attraverso operazioni di crossover e mutazione, selezionando progressivamente quelle migliori.

Questo approccio si distingue dalla filogenesi additiva perche' non presuppone che le distanze tra le specie possano essere rappresentate come la somma delle distanze lungo i rami di un albero. In altre parole, in un albero filogenetico additivo, la struttura e' piu' semplice e le distanze sono coerenti con una struttura ad albero, ma quando le distanze evolutive non soddisfano questa condizione, come avviene nei casi di filogenesi non additiva, l'approccio dei minimi quadrati e' utile per cercare di ricostruire un albero che migliori la corrispondenza tra le distanze, pur riconoscendo che potrebbero esserci discrepanze.

FILOGENESI BASATA SUI CARATTERI

La filogenesi basata su caratteri si differenzia dalla filogenesi basata sulla distanza, in quanto non utilizza una matrice di distanze tra le specie, ma considera direttamente i caratteri genetici, morfologici o altri tratti distintivi delle specie. Ogni carattere puo' essere binario (presenza/assenza) o multi-stato (con diverse varianti). In questo approccio, il principale obiettivo e' costruire un albero filogenetico che minimizzi il numero di cambiamenti evolutivi necessari per spiegare le differenze nei caratteri tra le specie, attraverso un principio chiamato parsimonia.

La parsimonia nella filogenesi basata su caratteri e' un principio fondamentale che guida la ricostruzione degli alberi evolutivi. Questo concetto si ispira al rasoio di Occam, che afferma che, tra piu' spiegazioni plausibili, quella piu' semplice e' generalmente la migliore. In filogenesi, la parsimonia suggerisce che l'albero filogenetico ideale e' quello che richiede il minor numero di cambiamenti nei caratteri evolutivi osservati, ossia il minimo numero di mutazioni o trasformazioni necessarie per spiegare le differenze tra le specie. Esistono due principali problemi legati alla parsimonia: il problema della piccola parsimonia e il problema della grande parsimonia.

Il problema della piccola parsimonia e' un concetto fondamentale nella filogenesi basata su caratteri e si concentra sulla ricerca dell'albero filogenetico che richiede il minor numero di cambiamenti nei caratteri evolutivi tra le specie. In filogenesi, la parsimonia implica che l'albero evolutivo ideale e' quello che minimizza il numero di mutazioni nei caratteri, come mutazioni genetiche, che separano le specie.

Il problema della piccola parsimonia si risolve cercando di assegnare i caratteri in modo da ridurre al minimo il numero di cambiamenti evolutivi necessari per spiegare le differenze osservate tra le specie. Ogni cambiamento di stato in un carattere, ad esempio una mutazione genetica o un cambiamento fenotipico, viene conteggiato come una transizione. L'obiettivo e' quello di trovare l'albero filogenetico che minimizza queste transizioni.

Un algoritmo molto utilizzato per risolvere il problema della piccola parsimonia e' l'algoritmo di Fitch. Questo algoritmo si basa sul calcolo dell'insieme di stati piu' parsimoniosi per ogni nodo dell'albero, considerando il numero minimo di cambiamenti necessari per spiegare i dati. Per ogni nodo, Fitch esplora le possibili transizioni di stato dei caratteri e cerca quella che comporta il minor numero di mutazioni. Il vantaggio dell'algoritmo di Fitch e' che e' relativamente semplice e veloce, ed e' in grado di risolvere il problema della piccola parsimonia in modo abbastanza efficiente. Tuttavia, e' anche limitato dal fatto che, come molti algoritmi di parsimonia, non considera il contesto evolutivo piu' ampio e potrebbe non essere in grado di gestire in modo ottimale situazioni molto complesse, come alberi con molte specie e caratteri.

Un altro algoritmo usato per il problema della piccola parsimonia e' l'algoritmo di Sankoff, che estende l'approccio di Fitch per gestire caratteri che hanno piu' di due stati. L'algoritmo di Sankoff, infatti, e' adatto quando i caratteri non sono binari (ad esempio, quando un carattere ha piu' di due varianti). Il suo funzionamento e' simile a quello di Fitch, ma con l'aggiunta di una maggiore complessita' computazionale, poiche' esplora piu' opzioni per ogni carattere. Come vantaggio, permette una maggiore flessibilita' nel trattare dati complessi con caratteri multi-stato, ma la sua complessita' aumenta, rendendolo piu' lento rispetto a Fitch, specialmente con un numero elevato di caratteri e specie.

Il vantaggio principale del problema della piccola parsimonia e' la sua capacita' di generare alberi filogenetici semplici, che rappresentano la spiegazione piu' economica delle differenze evolutive tra le specie. Tuttavia, questa semplicita' puo' essere anche un svantaggio, poiche' la parsimonia non tiene conto di fattori come la velocita' di evoluzione dei caratteri o fenomeni come la convergenza evolutiva, che potrebbero rendere necessari piu' cambiamenti di quelli suggeriti dalla parsimonia.

In termini di complessita' computazionale, sia l'algoritmo di Fitch che quello di Sankoff sono considerati NP-difficili, il che significa che la loro difficolta' cresce esponenzialmente con l'aumento delle specie e dei caratteri. Cio' comporta che, per grandi set di dati, e' difficile ottenere la soluzione ottimale in tempi ragionevoli. Sebbene gli algoritmi siano efficienti per alberi di piccole o medie dimensioni, la loro applicazione a set di dati molto ampi richiede l'uso di approcci euristici o algoritmi di ottimizzazione che permettano di trovare soluzioni approssimative in tempi piu' brevi.

Il problema della grande parsimonia e' una generalizzazione del problema della piccola parsimonia, ma con un obiettivo piu' complesso: non si tratta solo di minimizzare il numero di cambiamenti evolutivi su un albero dato, ma di trovare direttamente la topologia dell'albero filogenetico che soddisfi questo criterio. In altre parole, mentre la piccola parsimonia assume che l'albero sia gia' noto e cerca di ottimizzare l'assegnazione dei caratteri, la grande parsimonia cerca contemporaneamente sia la struttura dell'albero sia la disposizione dei caratteri piu' parsimoniosa.

Questo problema e' molto piu' difficile da risolvere perche' il numero di possibili alberi cresce in modo esponenziale con l'aumentare del numero di specie. Ogni configurazione dell'albero deve essere valutata in termini di numero di cambiamenti nei caratteri, e il compito diventa rapidamente intrattabile per insiemi di dati di grandi dimensioni. Per questo motivo, il problema della grande parsimonia e' NP-difficile, rendendo impossibile una ricerca esaustiva della soluzione ottimale in tempi ragionevoli per dataset complessi.

Per affrontare questa difficolta', vengono utilizzati algoritmi euristici, tra cui uno dei piu' noti e' l'algoritmo di scambio del vicino piu' prossimo (Nearest Neighbor Interchange, NNI). Questo algoritmo cerca di migliorare la topologia dell'albero apportando piccoli cambiamenti locali, scambiando la posizione di sottoalberi adiacenti per verificare se la nuova configurazione riduce il numero totale di cambiamenti evolutivi richiesti. Il procedimento e' iterativo: partendo da un albero iniziale, si esplorano piccole variazioni e si accettano quelle che migliorano la parsimonia fino a raggiungere un punto in cui non e' possibile ottenere ulteriori miglioramenti con scambi locali.

Il vantaggio principale dell'algoritmo NNI e' che permette di esplorare efficientemente lo spazio delle possibili topologie senza dover testare tutte le configurazioni possibili, riducendo il costo computazionale rispetto a una ricerca esaustiva. Tuttavia, ha anche limitazioni: essendo un metodo locale, puo' rimanere bloccato in ottimi locali, cioe' soluzioni che sembrano ottimali in un contesto ristretto ma che non sono la soluzione globale migliore. Per ovviare a questo problema, spesso si combinano piu' tecniche di ottimizzazione, come il Simulated Annealing o gli algoritmi genetici, che permettono di esplorare in modo piu' ampio lo spazio delle possibili soluzioni. Un altro svantaggio dell'approccio basato sulla grande parsimonia e' che, sebbene minimizzare il numero di cambiamenti nei caratteri sia una strategia efficace, non sempre riflette accuratamente i processi evolutivi reali. L'evoluzione puo' essere influenzata da tassi di mutazione diversi tra caratteri e cladi, eventi di convergenza evolutiva e selezione naturale, che non sono catturati dal semplice criterio della parsimonia.

ALBERO DI COPERTURA MINIMO

Gli alberi di copertura minimi (Minimum Spanning Trees, MST) sono strutture fondamentali nella teoria dei grafi e trovano applicazione in diversi ambiti, tra cui la genetica e la filogenesi. Un MST e' un sottoinsieme di un grafo connesso e pesato che collega tutti i nodi minimizzando la somma dei pesi dei collegamenti, evitando cicli. Questa proprieta' lo rende utile per rappresentare le relazioni evolutive tra specie o popolazioni, riducendo la complessita' senza perdere informazioni essenziali sulle distanze genetiche.

Nel contesto delle sequenze Alu, che forniscono dati genetici sulle relazioni tra popolazioni umane, gli alberi di copertura minimi vengono utilizzati per costruire una struttura che rappresenti la storia evolutiva con il minor numero di passaggi necessari. Dato che le distanze tra le popolazioni vengono calcolate in base alla presenza o assenza di specifiche sequenze Alu, un MST aiuta a identificare il percorso piu' parsimonioso che collega i diversi gruppi umani, rivelando cosi informazioni sulle loro migrazioni e origini comuni.

La costruzione di un MST si basa su algoritmi greedy, che prendono decisioni locali ottimali a ogni passo per ottenere una soluzione globale efficiente. Tra gli algoritmi piu' noti, l'algoritmo di Prim e' particolarmente utilizzato per la sua semplicita' ed efficienza. Esso inizia scegliendo un nodo qualsiasi e, a ogni iterazione, aggiunge il collegamento piu' corto che collega un nuovo nodo all'albero in costruzione. Questo processo continua fino a includere tutti i nodi, garantendo sempre la soluzione ottimale.

L'efficienza computazionale di Prim dipende dalla struttura utilizzata per gestire le distanze: con una matrice di adiacenza ha complessita' $O(n^2)$, mentre con una coda di priorita' basata su heap binario puo' essere migliorato a $O(m \log n)$, rendendolo adatto per grafi di grandi dimensioni. Questo e' un aspetto fondamentale nelle applicazioni biologiche, dove il numero di specie o popolazioni puo' essere molto elevato.

I vantaggi degli alberi di copertura minimi nel campo della genetica includono la capacita' di sintetizzare grandi quantità di dati in una rappresentazione chiara e intuitiva delle relazioni evolutive, evidenziando connessioni tra gruppi con il minor costo possibile. Tuttavia, presentano anche alcuni limiti: un MST fornisce solo una delle possibili rappresentazioni delle relazioni tra i dati e puo' non catturare la complessita' di processi evolutivi piu' intricati, come eventi di ibridazione o selezione naturale che coinvolgono piu' percorsi paralleli.

Nonostante questi limiti, gli MST rimangono uno strumento potente nella genetica delle popolazioni e nella filogenesi, specialmente quando si lavora con dati che non si adattano perfettamente a modelli di alberi filogenetici piu' rigidi. Nel caso delle sequenze Alu, l'uso degli MST aiuta a ricostruire in modo efficiente la storia delle migrazioni umane, fornendo una rappresentazione semplificata ma efficace della loro evoluzione.

MODELLI DI RETI E REGOLAZIONE GENICA

La teoria dei grafi, sviluppata da Leonhard Euler nel 1735 con il problema dei sette ponti di Königsberg, fornisce un potente strumento per modellare sistemi complessi. Euler dimostro' che un grafo possiede un cammino euleriano (un percorso che attraversa ogni arco una sola volta) se e solo se al massimo due vertici hanno un numero dispari di connessioni. Questo concetto ha avuto un impatto significativo in diversi campi, inclusa la biologia.

Nella regolazione genica, i geni e le proteine interagiscono in reti complesse, che possono essere rappresentate come grafi, dove i nodi corrispondono ai geni o alle proteine e gli archi alle loro interazioni. Questi modelli aiutano a comprendere come l'attivazione o la repressione di un gene influenzi l'intero sistema biologico. Perturbazioni in queste reti possono portare a malattie come il cancro o i disturbi neurologici, e la loro analisi permette di identificare geni chiave per lo sviluppo di terapie mirate.

I grafi possono essere orientati (con direzione nei collegamenti) o pesati (dove gli archi hanno un valore numerico che rappresenta la forza o il costo della connessione). Inoltre, la struttura di una rete influisce sul suo funzionamento: ad esempio, i geni altamente connessi (hub) giocano un ruolo centrale nel controllo della regolazione genica.

Negli anni '50, Erdos e Renyi introdussero la teoria dei grafi casuali, dove le connessioni tra nodi seguono una probabilita' casuale, in contrasto con le strutture regolari e prevedibili. Questo approccio ha portato alla scoperta di fenomeni come il "small-world", che descrive come la maggior parte delle reti reali (biologiche, sociali, computazionali) presenti sia connessioni casuali sia schemi strutturati.

Le reti possono essere regolari o casuali: le prime hanno connessioni prevedibili, mentre le seconde sono basate su probabilita'. Le reti reali mescolano queste caratteristiche. Il concetto di "small-world", che afferma che chiunque sulla Terra puo' essere connesso a qualsiasi altra persona tramite pochi intermediari, e' stato dimostrato da Karinthy e

studiato da Milgram. Nel 1998, Watts e Strogatz hanno scoperto che le reti reali sono altamente "clusterizzate" e presentano brevi percorsi, combinando cosi le proprieta' dei grafi regolari e casuali.

Le reti reali seguono distribuzioni di grado asimmetriche, descritte dalla legge di potenza, con molte connessioni locali e pochi nodi con numerose connessioni. Le reti sociali, biologiche, di informazione e tecnologiche mostrano questa struttura. Queste reti sono robuste rispetto alla rimozione casuale di nodi, ma vulnerabili a rimozioni mirate di nodi cruciali. Le reti sociali, ad esempio, tendono a formare comunita' dense, che sono vulnerabili a eventi mirati.

L'approccio della vita artificiale (Alife) si concentra su simulazioni sintetiche per comprendere i fenomeni biologici e applicarli in contesti tecnologici. Le reti dinamiche discrete vengono utilizzate per modellare sistemi biologici, sociali e tecnologici. Ogni nodo puo' assumere un valore che rappresenta uno stato, e la rete evolve in modo sincrono o asincrono in base agli stati dei nodi vicini.

Le automobili cellulari (CA), come nel Gioco della Vita di Conway, sono un esempio di sistema dinamico in reti discrete. In questo modello, la dinamica e' determinata da regole locali basate sugli stati delle celle vicine, creando comportamenti complessi da regole semplici. Questo approccio e' utile per modellare fenomeni biologici, come la regolazione genica, in cui i geni interagiscono per attivarsi o reprimersi.

I Random Boolean Networks (RBNs) sono un modello semplificato per lo studio della dinamica della regolazione genica. Ogni gene e' rappresentato come un nodo in una rete, e la sua espressione dipende dalle interazioni con altri geni. Le RBNs evolvono in configurazioni determinate da regole di transizione locali. Queste reti possono entrare in regimi dinamici ordinati, caotici o critici, con il regime critico considerato ideale per la flessibilita' biologica. L'impatto di perturbazioni come errori nei geni puo' essere studiato attraverso l'analisi dei Derrida plots, che esaminano la distanza tra configurazioni della rete.

Nonostante i modelli come gli RBNs siano semplificati, sono comunque utili per comprendere le dinamiche delle reti biologiche, come la regolazione genica e la transizione tra gli stati cellulari. I modelli piu' complessi, come quelli che considerano funzioni di aggiornamento additive e soglie, cercano di avvicinarsi di piu' alla realta' biologica.

PREDIZIONE STRUTTURALE PROTEICA

COS'È LA PREDIZIONE STRUTTURALE PROTEICA?

La predizione strutturale proteica e' un campo dell'avvio informatica che si occupa di determinare la struttura tridimensionale di una proteina a partire dalla sua sequenza amminoacidica. L'obiettivo e' comprendere come una proteina si ripiega e assume la sua forma definitiva, poiche' la struttura e' strettamente legata alla funzione.conoscere la forma di una proteina permette di capire come interagisce con altre molecole, quali reazioni catalizza e in che modo puo' essere coinvolta in processi biologici o patologie.questa conoscenza e'essenziale per progettare farmaci mirati, comprendere malattie genetiche e sviluppare nuove biotecnologie.

Le proteine si organizzano in quattro livelli di struttura:

1. Struttura primaria: la sequenza lineare di amminoacidi uniti da legami peptidici.questa sequenza contiene gia' tutte le informazioni necessarie per il ripiegamento della proteina;
2. Struttura secondaria: rappresenta le prime interazioni locali tra amminoacidi vicini nella sequenza, dando origine a elementi strutturali come α-eliche e foglietti ß, stabilizzati da legami a idrogeno;
3. Struttura terziaria: la disposizione tridimensionale completa della proteina, risultante dall'interazione tra le strutture secondarie e le catene laterali degli aminoacidi. E' stabilizzata da legami deboli (idrogeno, idrofobici, ionici) e forti (ponti disolfuro). Questa struttura determina la funzione biologica della proteina: ad esempio, un enzima deve avere una forma specifica per legarsi al suo substrato e catalizzare una reazione;
4. Struttura quaternaria (se e' presente): l'assemblaggio di piu' subunita' proteiche, come avviene per l'emoglobina.

La predizione strutturale cerca di passare dalla struttura primaria alla terziaria, prevedendo il modo in cui una proteina si ripiega. Questo processo e' complesso perche' dipende da numerose forze chimiche e fisiche. Tuttavia, grazie a metodi computazionale avanzati, come modelli fisici, intelligenza artificiale (es. AlphaFold) i confronto con proteine note, e' possibile determinare la struttura con crescente accuratezza. Questo ha un impatto enorme nella ricerca biomedica e farmacologica, permettendo di sviluppare terapie piu' efficaci basati sulla comprensione della funzione delle proteine.

PROPENSIONI STRUTTURALI

Le propensioni strutturali descrivono la tendenza degli aminoacidi a favorire determinate strutture secondarie in base alle loro proprieta' chimico-fisiche. Poiche' ogni amminoacido ha una catena laterale con caratteristiche uniche, come dimensione, forma, idrofobicita' e carica, esso influenzera' il modo in cui la proteina si ripiega nello spazio.

Ad esempio, alcuni amminoacidi hanno una maggiore predisposizione per le α-eliche, come l'alanina, perche' la sua catena laterale e' piccola e non interferisce con la formazione dei legami a idrogeno tipici di questa struttura. Altri, come valina, isoleucina e leucina, sono piu' frequentemente trovati nei foglietti ß, perche' le loro catene laterali ramificate si adattano bene alla disposizione planare di questa struttura. La prolina, invece, e' nota per la sua rigidita' dovuta alla sua struttura ciclica, che interrompe le α-eliche e spesso introduce curve o svolte nelle proteine.

I due principi fondamentali delle propensioni strutturali sono:

1. Influenza diretta sulla conformazione locale: un amminoacido, con le sue caratteristiche chimiche e steriche, puo' determinare direttamente la struttura secondaria in cui si trova.ad esempio, la presenza di prolina in una catena polipeptidica spesso forza una piega nella proteina, mentre una sequenza ricca di alanina e glutammato favorisce la formazione di un' α-elica;
2. Pressione selettiva e funzione della proteina: in alcuni casi, la struttura globale della proteina impone i vincoli sulla selezione degli aminoacidi. Questo si osserva particolarmente nelle proteine di membrana, che spesso contengono regioni strutturati in foglietti ß per adattarsi ad un ambiente lipidico e formare pori e canali. Qui, la necessita' di una struttura specifica puo' guidare la selezione evolutiva degli aminoacidi che meglio si adattano a quella conformazione.

Due dei metodi piu' noti per prevedere la struttura secondaria delle proteine sono il metodo di Chou-Fasman e il metodo di Garnier-Osguthorpe-Robson (GOR).

Il metodo di Chou-Fasman utilizza dati statistici basati su proteine con strutture note per assegnare a ciascun amminoacido una propensione a formare α-eliche, foglietti ß o regioni disordinate. Se una sequenza contiene una concentrazione sufficiente di aminoacidi con alta propensione per una determinata struttura, il metodo predice la formazione di quella struttura. Si tratta di un approccio semplice ma efficace per ottenere una prima stima delle strutture secondarie.

Il metodo di Garnier-Osguthorpe-Robson (GOR), utilizza un modello basato su finestra di contesto, analizzando non solo la propensione di un singolo amminoacido, ma anche quella dei suoi vicini nella sequenza. Si basa su un approccio probabilistico che considera l'influenza degli aminoacidi circostanti per migliorare l'accuratezza della predizione.

Entrambi i metodi hanno una precisione limitata, circa 55-65%, principalmente perche', pur considerando il contesto locale di ogni amminoacido, non tengono conto del contesto globale della sequenza proteica, che puo' influenzare la struttura finale. Ad esempio, un amminoacido potrebbe avere una preferenza strutturale diversa se si trova in una proteina citoplasmaticarispetto ad una proteina di membrana.questa limitazione ha portato allo sviluppo di metodi piu' avanzati.

Per superare le limitazioni dei modelli tradizionali, sono stati sviluppati metodi adattivi che includono modelli basati su reti neurali e modelli basati sull'omologia. Grazie a questi metodi avanzati, l'accuratezza della produzione della struttura secondaria e terziaria e' migliorata notevolmente, arrivando anche oltre il 90% di affidabilita' nei casi migliori,

rivoluzionando cosi la biologia strutturale e le sue applicazioni in campo biomedico e farmaceutico.

I metodi basati su reti neurali utilizzano algoritmi di apprendimento automatico per analizzare enormi dataset di strutture proteiche note e apprendere schemi complessi. Le reti neurali possono tenere conto di interazioni a lungo raggio nella sequenza e migliorare significativamente l'accuratezza della predizione.un esempio moderno di questo approccio e' AlphaFold, che utilizza il deep learning per prevedere strutture proteiche con una precisione senza precedenti.

I metodi basati sull'omologia, invece, si basano sulla comparazione con proteine gia' conosciute e caratterizzate sperimentalmente. Se una nuova proteina ha una sequenza simile a una proteina con struttura nota, e' probabile che abbia una conformazione simile.questo approccio e' particolarmente efficace perche' sfrutta dati sperimentali reali per migliorare le tradizioni.

EVOLUZIONE DELLA PREDIZIONE

Fino al 2015, la predizione della struttura delle proteine aveva una bassa percentuale di successo e si basava principalmente su due approcci: la modellazione basata su template e quella senza template. La prima sfruttava proteine con strutture note come riferimento, mentre la seconda cercava di prevedere la conformazione senza appoggiarsi a modelli esistenti, utilizzando tecniche come l'allineamento di sequenze multiple e l'analisi della coevoluzione.Tuttavia, questi metodi avevano forti limitazioni: la modellazione basata su template era inefficace per proteine senza omologhi noti, mentre quella senza template faticava a fornire previsioni accurate a causa della complessita' del ripiegamento proteico.

A partire dal 2008, l'introduzione delle deep neural networks ha iniziato a migliorare la qualita' delle previsioni. Architetture come ResNet si sono rivelate fondamentali non solo nella predizione strutturale delle proteine, ma anche in altri ambiti, come l'identificazione di masse nelle mammografie. Nel 2016, ResNet ha mostrato un miglioramento significativo nella capacita' di estrarre caratteristiche strutturali dalle sequenze proteiche, segnando un punto di svolta.

Parallelamente, l'analisi della coevoluzione ha continuato ad essere utilizzata per individuare interazioni tra aminoacidi sulla base di correlazioni statistiche piuttosto che di contatti diretti. Nel 2018, il problema e' stato riformulato come una segmentazione semantica delle immagini, permettendo di prevedere simultaneamente tutti i contatti di una proteina. L'adozione di ResNet l'uso di dati e evolutivi hanno spinto enormemente i progressi nella previsione della struttura tridimensionale delle proteine.

Negli ultimi anni, l'introduzione dei Transformer ha portato ad un'ulteriore miglioramento, integrando meglio le informazioni dei template nei modelli di deep learning e affinando le previsioni.questi sviluppi hanno reso possibile una predizione sempre piu' accurata della struttura proteica, con impatti significativi nella biologia strutturale, nella bio informatica e nella progettazione di nuovi farmaci.

ALLINEAMENTO DI SEQUENZE

COS'È' L'ALLINEAMENTO DI SEQUENZE?

L'allineamento di sequenze e' una tecnica utilizzata per confrontare due o piu' sequenze di caratteri, come DNA, RNA o proteine, al fine di individuare somiglianze, differenze ed eventuali relazioni evolutive. Il processo consiste nel disporre le sequenze in modo che le corrispondenze tra le loro posizioni siano evidenziate, minimizzando le differenze attraverso operazioni di inserzione, delezione o sostituzione.

Per valutare la qualita' di un allineamento, viene assegnato uno score basato su determinati criteri: un match tra due caratteri identici contribuisce positivamente, un mismatch (ovvero una sostituzione tra due caratteri diversi) ha un costo negativo, mentre un gap (inserzione o delezione) introduce penalita' che dipendono dalla strategia adottata. In genere, la penalita' per i gap e' maggiore di quella per i mismatch, poiche' le inserzioni e delezioni possono indicare eventi di mutazione piu' significativi.

Uno degli strumenti piu' semplici per misurare la differenza tra due sequenze e' la distanza di Hamming, che conta il numero di posizioni in cui i caratteri corrispondenti differiscono. Tuttavia, questo metodo e' applicabile solo a sequenze di lunghezza uguale e non tiene conto delle inserzioni o delezioni, limitandone l'uso in contesti piu' complessi.

L'allineamento puo' essere calcolato con metodi come la programmazione dinamica, che permette di trovare la soluzione ottimale memorizzando i risultati intermedi e riducendo il numero di calcoli necessari. Due approcci principali sono l'allineamento globale, che confronta le sequenze nella loro interezza, e l'allineamento locale, che cerca solo le regioni di massima similarita'. La scelta del metodo dipende dall'obiettivo: il primo e' utile quando le sequenze sono affini lungo tutta la loro lunghezza, mentre il secondo e' piu' adatto a individuare regioni conservate in sequenze molto diverse.

LONGEST COMMON SUBSEQUENCE

La Longest Common Subsequence (LCS) e' un approccio flessibile all'allineamento di sequenze, poiche' permette di trovare la sottosequenza piu' lunga comune a due stringhe senza imporre che i caratteri siano consecutivi, ma mantenendo il loro ordine originale. Questo metodo e' particolarmente utile per l'allineamento globale, poiche' identifica la struttura comune tra due sequenze minimizzando il numero di operazioni necessarie per trasformarne una nell'altra.

L'Alignment Game e' una rappresentazione formale del problema di allineamento delle sequenze sotto forma di un gioco combinatorio. In questo modello, due giocatori collaborano per costruire un allineamento ottimale tra due stringhe, massimizzando il numero di caratteri corrispondenti (match) e minimizzando il numero di operazioni di inserzione e delezione (gap).

Il gioco si svolge su una griglia bidimensionale, dove ogni cella rappresenta un possibile stato dell'allineamento. I giocatori possono scegliere tra tre mosse: avanzare diagonalmente (match), muoversi orizzontalmente (inserzione in una sequenza) o muoversi verticalmente (delezione in una sequenza). L'obiettivo e' raggiungere l'ultima cella della griglia seguendo un percorso che massimizzi i match e minimizzi i costi associati ai gap.

Uno dei principali vantaggi dell'Alignment Game e' che fornisce una chiara visualizzazione del processo di allineamento e permette di modellarlo in modo intuitivo. Inoltre, evidenzia le somiglianze con altri problemi di ottimizzazione, facilitando l'applicazione di tecniche come la programmazione dinamica per trovare una soluzione ottimale. Tuttavia, un limite e' che la rappresentazione sotto forma di gioco non riduce la complessita' computazionale del problema, che rimane polinomiale nel caso della programmazione dinamica standard (O(mn), con m e n lunghezze delle due sequenze). Inoltre, se si considerano costi complessi per i gap, la soluzione puo' diventare piu' difficile da calcolare.

Dal punto di vista pratico, l'Alignment Game e' utile per comprendere il comportamento degli algoritmi di allineamento e le scelte ottimali nel confronto tra due sequenze, ma non rappresenta un metodo computazionalmente piu' efficiente rispetto alle tecniche gia' note come la programmazione dinamica.

Il problema del turista di Manhattan e' un modello utilizzato in ottimizzazione combinatoria per rappresentare percorsi ottimali su una griglia. L'idea e' quella di un turista che si muove su una citta' con strade disposte a griglia (come Manhattan) e deve raggiungere una destinazione nel minor tempo possibile, potendo spostarsi solo verso destra o verso il basso. Ogni strada puo' avere un costo associato, e l'obiettivo e' trovare il percorso di costo minimo dalla partenza all'arrivo.

Questo problema si risolve in modo efficiente con la programmazione dinamica. Si costruisce una matrice in cui ogni cella rappresenta il costo minimo per arrivare a quella posizione. Il valore di una cella dipende dai valori delle celle adiacenti (a sinistra e sopra), perche' il turista puo' arrivare solo da quelle direzioni. In questo modo, si evita di dover esplorare tutte le possibili combinazioni di percorsi, riducendo la complessita' computazionale rispetto a una ricerca esaustiva.

Uno dei vantaggi principali di questo approccio e' la sua efficienza computazionale: la soluzione ottima viene calcolata in tempo polinomiale, con complessita' O(nm) per una griglia di dimensioni nxm. Inoltre, il metodo permette di incorporare facilmente variazioni, come penalita' per alcuni percorsi o premi per altri. Tuttavia, il problema presenta anche degli svantaggi. Se la griglia e' molto grande, lo spazio necessario per memorizzare la matrice puo' diventare proibitivo. Inoltre, il metodo funziona bene solo se le mosse sono limitate a destra e in basso; in scenari piu' complessi, come percorsi con movimenti liberi, sono necessarie tecniche piu' avanzate.

Il collegamento con la Longest Common Subsequence (LCS) nasce dal fatto che il calcolo della LCS puo' essere visto come un problema su una griglia, dove ogni cella rappresenta un sottoproblema parziale dell'allineamento. Muoversi diagonalmente nella griglia corrisponde a un match tra due caratteri, mentre spostarsi orizzontalmente o verticalmente

rappresenta un'operazione di inserzione o delezione. Cosi come il turista cerca il percorso di costo minimo, l'algoritmo della LCS cerca la sequenza piu' lunga comune, minimizzando le operazioni di modifica necessarie.

Il problema del cambio e' un classico problema di ottimizzazione combinatoria in cui, dato un certo importo di denaro e un insieme di tagli di monete disponibili, si cerca di determinare il numero minimo di monete necessarie per ottenere esattamente quell'importo. L'obiettivo e' trovare la combinazione ottimale di monete che minimizzi il numero totale utilizzato.

Il problema si affronta in modo efficiente con la programmazione dinamica. Si costruisce un array in cui ogni posizione rappresenta l'importo da raggiungere e il valore in quella posizione indica il numero minimo di monete necessario per ottenerlo. La soluzione di ogni sottoproblema dipende dalle soluzioni precedenti: per ogni importo, si valuta qual e' la scelta migliore tra le monete disponibili, aggiornando il valore dell'array in base alla soluzione piu' conveniente tra quelle gia' calcolate.

Uno dei vantaggi di questo approccio e' la sua efficienza: invece di esplorare tutte le possibili combinazioni di monete con un algoritmo esaustivo, la programmazione dinamica permette di risolvere il problema in tempo O(nC), dove C e' l'importo da raggiungere e n e' il numero di tagli di monete disponibili. Inoltre, l'algoritmo garantisce sempre una soluzione ottimale, purche' il problema sia formulato correttamente.

Tuttavia, presenta anche svantaggi. Se il numero di monete disponibili e' grande o se l'importo da raggiungere è molto alto, la memoria necessaria per memorizzare le soluzioni intermedie puo' diventare un limite. Inoltre, non tutte le varianti del problema possono essere risolte con programmazione dinamica: se le monete non seguono un sistema "canonico" (come nei paesi con tagli non convenzionali), l'algoritmo puo' fallire nel trovare la soluzione ottimale.

Il collegamento con la Longest Common Subsequence (LCS) con penalita' per gap nasce dal fatto che entrambi i problemi condividono una struttura simile. Nel caso della LCS con gap, si cerca la sequenza piu' lunga possibile minimizzando le penalita' per le inserzioni e delezioni. Analogamente, nel problema del cambio, si cerca la combinazione ottimale di monete minimizzando il numero di elementi utilizzati. In entrambi i casi, la soluzione ottimale si costruisce passo dopo passo, scegliendo sempre l'opzione migliore tra quelle disponibili, il che giustifica l'uso della programmazione dinamica per risolverli in modo efficiente.

PROGRAMMAZIONE DINAMICA E BACKTRACKING

La programmazione dinamica e' una tecnica fondamentale per risolvere problemi di ottimizzazione in cui una soluzione globale puo' essere costruita a partire da soluzioni ottimali di sottoproblemi piu' semplici. Questo approccio e' particolarmente efficace quando il problema presenta una struttura ricorsiva e quando le soluzioni parziali possono essere riutilizzate per evitare calcoli ridondanti.

Nel caso dell'alignment game, del problema del turista di Manhattan e del problema del cambio, la programmazione dinamica permette di trovare la soluzione ottimale in modo efficiente, memorizzando i risultati intermedi in una tabella e costruendo la soluzione finale in modo incrementale. Ad esempio, nel problema della Longest Common Subsequence (LCS), che e' alla base dell'allineamento di sequenze, si costruisce una matrice in cui ogni cella rappresenta la migliore soluzione trovata fino a quel punto, considerando le possibili operazioni di match, mismatch o gap. Questo stesso principio si applica al problema del turista di Manhattan, in cui il percorso ottimale viene determinato considerando il miglior tragitto possibile per ogni punto della griglia, e al problema del cambio, in cui la combinazione ottimale di monete si ottiene valutando tutte le possibilita' precedenti.

Una volta completata la tabella con le soluzioni parziali, e' necessario risalire attraverso le scelte effettuate per ricostruire il percorso ottimale. Questo passaggio viene gestito con il backtracking, una tecnica che permette di ripercorrere le decisioni prese, partendo dal risultato finale e risalendo fino all'inizio del problema. Ad esempio, nell'allineamento di sequenze, il backtracking consente di determinare esattamente quali caratteri sono stati allineati e dove sono stati introdotti gap. Nel problema del turista di Manhattan, permette di identificare il cammino seguito sulla griglia, mentre nel problema del cambio consente di ricostruire quali monete sono state utilizzate per ottenere l'importo desiderato.

L'efficacia della programmazione dinamica deriva dalla sua capacita' di ridurre il numero di calcoli ripetuti, migliorando significativamente le prestazioni rispetto ad approcci ingenuamente ricorsivi o esaustivi. Tuttavia, il costo in termini di memoria puo' diventare un limite quando le tabelle di memorizzazione crescono troppo, rendendo necessarie ottimizzazioni per ridurre il consumo di spazio.

GRAFO DI ALLINEAMENTO

Il grafo di allineamento e' una rappresentazione strutturata di tutte le possibili corrispondenze tra due sequenze. Si costruisce utilizzando una matrice in cui ogni cella rappresenta un possibile stato dell'allineamento tra i caratteri delle due sequenze. Gli archi del grafo collegano queste celle, indicando le operazioni che si possono eseguire per passare da un allineamento parziale a quello successivo.

Ogni arco e' associato a un peso, che corrisponde al costo di una determinata operazione. Quando due caratteri corrispondono, l'arco rappresenta un match con un punteggio positivo. Se invece i caratteri sono diversi, l'arco rappresenta un mismatch, a cui viene assegnata una penalita'. Infine, se l'allineamento richiede l'inserimento o la cancellazione di un carattere, l'arco rappresenta un gap, e il suo peso e' determinato dalla penalita' per inserzioni e delezioni.

L'obiettivo dell'allineamento e' trovare il percorso con il costo ottimale all'interno di questo grafo. Seguendo un approccio di programmazione dinamica, si calcola per ogni cella della matrice il miglior punteggio possibile combinando le soluzioni precedenti, in modo da costruire progressivamente il miglior allineamento globale o locale tra le sequenze. Le

penalita' assegnate ai gap influenzano notevolmente il risultato: penalita' elevate rendono l'allineamento piu' rigido, evitando la presenza di troppe inserzioni e delezioni, mentre penalita' piu' basse consentono una maggiore flessibilita' nella corrispondenza tra le due sequenze.

Questa struttura consente di modellare il problema dell'allineamento in modo chiaro e sistematico, fornendo una base solida per implementare algoritmi efficienti in grado di determinare la soluzione ottimale in termini di costo e qualita' dell'allineamento.

L'allineamento globale, l'allineamento locale e l'allineamento multiplo sono tre approcci distinti per confrontare sequenze, ciascuno con caratteristiche specifiche e con diverse implicazioni in termini di complessita' computazionale e qualita' dei risultati.

Nell'allineamento globale, l'intera lunghezza delle due sequenze viene considerata nel confronto, cercando di trovare il miglior allineamento possibile dall'inizio alla fine. Questo significa che, anche se alcune regioni delle sequenze sono molto diverse, il metodo cerchera' comunque di allinearle, spesso introducendo gap per mantenere la struttura complessiva. L'uso di penalita' per inserzioni e delezioni e' cruciale: se i costi dei gap sono elevati, si preferisce mantenere una corrispondenza piu' stretta tra i caratteri, mentre penalita' piu' basse rendono l'allineamento piu' flessibile, permettendo interruzioni. Un vantaggio di questo approccio e' che fornisce un confronto completo tra due sequenze, utile per analisi filogenetiche o per confrontare geni interi. Tuttavia, se le sequenze hanno solo regioni parzialmente simili, il metodo puo' produrre allineamenti poco significativi, con un costo computazionale relativamente alto, pari a O(mn) per due sequenze di lunghezza m e n.

L'allineamento locale, invece, si concentra sull'identificazione della regione di maggiore similarita' tra le sequenze, ignorando le parti non allineabili. Questo approccio e' piu' adatto per confrontare sottosequenze significative all'interno di sequenze piu' lunghe, come il riconoscimento di domini proteici o il confronto tra frammenti di DNA. L'uso delle penalita' nei gap e' diverso rispetto all'allineamento globale: poiche' l'obiettivo e' trovare segmenti altamente conservati, penalita' elevate tendono a troncare regioni con scarsa similarita', migliorando la qualita' del risultato. Il principale vantaggio di questo metodo e' la capacita' di individuare somiglianze biologicamente rilevanti anche tra sequenze che, nel loro insieme, potrebbero non essere comparabili. Lo svantaggio e' che potrebbe trascurare informazioni globali utili in alcuni contesti, e il calcolo rimane comunque costoso, con una complessita' O(mn), sebbene spesso piu' veloce nella pratica grazie alla terminazione anticipata.

L'allineamento multiplo di sequenze (MSA) e' un'estensione dell'allineamento globale a piu' di due sequenze. L'obiettivo e' trovare la migliore disposizione relativa di tutte le sequenze, massimizzando le corrispondenze e minimizzando le penalita' per gap. Questo tipo di allineamento e' particolarmente utile in bioinformatica per identificare regioni conservate tra piu' geni o proteine. Tuttavia, la complessita' computazionale cresce esponenzialmente con il numero di sequenze, rendendo impraticabile l'uso diretto della programmazione dinamica. Per questo motivo, si utilizzano algoritmi euristici o metodi basati su guide di allineamento progressivo, che riducono il costo computazionale ma possono produrre soluzioni subottimali. L'inclusione di penalita' per gap e' cruciale per evitare allineamenti distorti, ma la scelta dei parametri puo' influenzare notevolmente il risultato finale.

In generale, l'allineamento globale e' ideale per sequenze simili e di lunghezza comparabile, mentre l'allineamento locale e' piu' adatto per identificare regioni di omologia in sequenze piu' lunghe e meno correlate. L'allineamento multiplo offre un quadro piu' ampio delle relazioni evolutive tra sequenze, ma a costo di una maggiore complessita' computazionale e di soluzioni spesso approssimate.

RICERCA COMBINATORIA DI PATTERN

PATTERN MATCHING MULTIPLO

Il pattern matching multiplo e' un problema computazionale che consiste nel cercare simultaneamente piu' pattern (sequenze) all'interno di un testo. Questo e' utile in molte applicazioni, tra cui la mappatura delle read in bioinformatica, dove si devono allineare migliaia o milioni di frammenti di DNA a un genoma di riferimento.

La mappatura delle read puo' essere vista come un problema di pattern matching multiplo: ogni read e' un pattern da cercare nel genoma, che funge da testo. Poiche' il genoma e' molto lungo e il numero di read da mappare e' enorme, il processo deve essere ottimizzato per ridurre il tempo di calcolo e l'uso di memoria. Qui entra in gioco la complessità computazionale, poiche' le soluzioni piu' semplici, come il confronto diretto di ogni read con il genoma, sarebbero troppo lente. Per migliorare l'efficienza, si utilizzano strutture dati avanzate (come trie, suffix tree o BWT) e algoritmi ottimizzati per gestire grandi quantita' di dati riducendo il numero di operazioni necessarie per trovare una corrispondenza.

PREFIX TRIE

I trie (o prefix trie) sono una struttura dati ad albero utilizzata per memorizzare e cercare efficientemente insiemi di stringhe, specialmente quando condividono prefissi comuni. Ogni nodo dell'albero rappresenta un carattere, e un cammino dalla radice a un nodo terminale forma una parola dell'insieme.

Il funzionamento di un trie si basa sulla suddivisione progressiva delle stringhe: partendo dalla radice, ogni lettera di una parola determina quale ramo dell'albero seguire. Questo permette di verificare in tempo ottimale se una stringa appartiene all'insieme e di effettuare ricerche di prefissi in maniera estremamente veloce.

Il principale vantaggio dei trie e' la velocita' di ricerca, inserimento e cancellazione, che avvengono in tempo O(m), dove m e' la lunghezza della stringa. A differenza delle strutture basate su confronti, come gli alberi binari di ricerca, il tempo di ricerca nei trie non dipende dal numero totale di stringhe memorizzate. Inoltre, supportano efficientemente operazioni come il pattern matching multiplo e la ricerca di parole con un determinato prefisso.

Tuttavia, i trie hanno anche degli svantaggi. Il principale e' il consumo di memoria, che puo' essere elevato se il numero di stringhe e' grande o se i caratteri possibili sono molti, poiche' ogni nodo puo' avere un numero elevato di figli. Per mitigare questo problema si utilizzano varianti piu' compatte, come il suffix tree, che riducono lo spazio necessario mantenendo buone prestazioni.

Nonostante il costo in memoria, i trie rimangono fondamentali in applicazioni come la mappatura delle read, il filtraggio dei dizionari e la ricerca veloce di parole, grazie alla loro capacita' di gestire grandi insiemi di stringhe con efficienza.

SUFFIX TRIE

I suffix trie sono una variante dei trie in cui vengono memorizzati tutti i suffissi di una stringa. In questa struttura, ogni suffisso della stringa viene inserito nel trie come una sequenza distinta di caratteri, partendo da ogni possibile posizione della stringa stessa.

Il funzionamento del suffix trie e' simile a quello di un trie tradizionale, ma invece di costruire l'albero a partire da parole indipendenti, lo si costruisce considerando ogni suffisso della stringa. Questo permette di rispondere in modo molto efficiente a domande sulla presenza di sottostringhe e di supportare operazioni di pattern matching rapide.

Uno dei principali vantaggi del suffix trie e' che consente di verificare se una sottostringa e' presente in una stringa di riferimento in tempo O(m), dove m e' la lunghezza della sottostringa cercata. E' anche utile per risolvere problemi come la ricerca del longest repeated substring o la compressione di dati.

Tuttavia, il costo in memoria e' un grande svantaggio. Un suffix trie non compresso puo' occupare $O(n^2)$ spazio per una stringa di lunghezza n, perche' ogni suffisso viene memorizzato esplicitamente e con esso molte ripetizioni di caratteri. Questo lo rende poco pratico per stringhe lunghe, come sequenze genomiche.

Per risolvere questo problema si utilizza spesso il suffix tree, che e' una versione compressa del suffix trie. Il suffix tree riduce drasticamente l'uso di memoria mantenendo tempi di ricerca efficienti, ed e' quindi preferito nelle applicazioni pratiche che richiedono la gestione di stringhe lunghe.

TRASFORMATA DI BURROWS-WHEELER

La trasformata di Burrows-Wheeler (BWT) e' una tecnica di riordinamento dei caratteri di una stringa che migliora la compressione e la ricerca di pattern. Non modifica il contenuto della stringa, ma ne altera l'ordine in modo da raggruppare caratteri simili, facilitando la compressione tramite tecniche come il run-length encoding.

Il funzionamento della BWT si basa sulla costruzione della matrice delle rotazioni cicliche della stringa, che vengono ordinate lessicograficamente. L'ultima colonna di questa matrice costituisce la trasformata BWT. Un aspetto fondamentale e' che questa trasformazione e' invertibile, cioe' permette di ricostruire la stringa originale senza perdita di informazione.

L'inversione della BWT sfrutta la proprieta' della trasformata per risalire alla stringa iniziale. Poiche' la prima colonna della matrice ordinata corrisponde ai caratteri della stringa originale in ordine alfabetico, e' possibile ricostruire la sequenza iniziale sfruttando le posizioni dei caratteri nella BWT e nelle colonne ordinate. Questo processo e' noto come LF-mapping (Last-to-First mapping) e permette di ricostruire la stringa in tempo lineare.

Uno dei principali vantaggi della BWT e' la capacita' di migliorare l'efficienza della compressione e della ricerca di pattern. Poiche' i caratteri simili tendono a raggrupparsi, la compressione risulta piu' efficace rispetto alla stringa originale. Inoltre, la ricerca di pattern diventa piu' efficiente grazie a tecniche come il backward search, che consente di individuare rapidamente tutte le occorrenze di un pattern senza dover esaminare l'intera stringa.

Tuttavia, la BWT ha anche degli svantaggi. Il principale e' la necessita' di calcolare e memorizzare la matrice delle rotazioni, che puo' richiedere tempo e spazio significativi per stringhe molto lunghe. Inoltre, l'inversione della trasformata, pur essendo lineare, richiede comunque l'accesso a strutture dati ausiliarie come la prima colonna della matrice ordinata, aumentando il consumo di memoria.

Nonostante queste limitazioni, la BWT e' ampiamente utilizzata in applicazioni come il sequenziamento del DNA e la compressione dei dati, dove la combinazione tra efficienza di ricerca e riduzione dello spazio occupato e' fondamentale.