Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea Magistrale in Fisica

Approccio network-based alla Discriminant analysis mediante HPC per la ricerca di signature ottimali in dati ad alta dimensionalità

Relatore:
Prof. Daniel Remondini

Presentata da:
Stefano Malagoli

Correlatore:
Dott. Giuseppe Levi

Sessione II

Anno Accademico 2013/2014

Indice

Sommario

Il presente lavoro di tesi si inserisce nell'ambito della classificazione di dati ad alta dimensionalità, sviluppando un algoritmo basato sul metodo della Discriminant Analysis. Esso classifica i campioni attraverso le variabili prese a coppie formando un network a partire da quelle che hanno una performance sufficientemente elevata. Successivamente, l'algoritmo si avvale di proprietà topologiche dei network (in particolare la ricerca di subnetwork e misure di centralità di singoli nodi) per ottenere varie signature (sottoinsiemi delle variabili iniziali) con performance ottimali di classificazione e caratterizzate da una bassa dimensionalità (dell'ordine di 10^1, inferiore di almeno un fattore 10^3 rispetto alle variabili di partenza nei problemi trattati). Per fare ciò, l'algoritmo comprende una parte di definizione del network e un'altra di selezione e riduzione della signature, calcolando ad ogni passaggio la nuova capacità di classificazione operando test di cross-validazione (k-fold o leave-one-out).

Considerato l'alto numero di variabili coinvolte nei problemi trattati – dell'ordine di 10^4 – l'algoritmo è stato necessariamente implementato su *High-Performance Computer*, con lo sviluppo in parallelo delle parti più onerose del codice C++, nella fattispecie il calcolo vero e proprio del discriminante e il sorting finale dei risultati.

L'applicazione qui studiata è a dati high-throughput in ambito genetico, riguardanti l'espressione genica a livello cellulare, settore in cui i database frequentemente sono costituiti da un numero elevato di variabili ($10^4 - 10^5$) a fronte di un basso numero di campioni ($10^1 - 10^2$). In campo medico-clinico, la determinazione di signature a bassa dimensionalità per la discriminazione e classificazione di campioni (e.g. sano/malato, responder/not-responder, ecc.) è un problema di fondamentale importanza, ad esempio per la messa a punto di strategie terapeutiche personalizzate per specifici sottogruppi di

pazienti attraverso la realizzazione di kit diagnostici per l'analisi di profili di espressione applicabili su larga scala.

L'analisi effettuata in questa tesi su vari tipi di dati reali mostra che il metodo proposto, anche in confronto ad altri metodi esistenti basati o meno sull'approccio a network, fornisce performance ottime, tenendo conto del fatto che il metodo produce signature con elevate performance di classificazione e contemporaneamente mantenendo molto ridotto il numero di variabili utilizzate per questo scopo.

Capitolo 1

Panoramica su dati "omici": tipologia e metodi di misurazione

In questo capitolo si descrivono i tipi di dati che vengono utilizzati in ambito biomedico per la caratterizzazione di processi biologici a livello cellulare e per studi di campioni sottoposti a diverse perturbazioni (e.g. case - control studies di tipo sano/malato, giovane/vecchio, trattamento con diversi farmaci). Questi dati sono caratterizzati da un elevata dimensionalità ($10^4 - 10^6$ variabili) e da notevoli fonti di rumore, per cui occorrono analisi con grande *robustness* per ottenere risultati accurati e consistenti con i reali meccanismi biologici soggiacenti. Nonostante questi problemi si evidenzia l'importanza di questi studi per meglio comprendere tali meccanismi e produrre applicazioni anche in ambito medico.

Con l'avvento del sequenziamento del DNA, a partire dagli anni '70 è stata aperta la strada allo studio dell'espressione genica, ovvero alla possibilità di misurare l'attività di migliaia di geni contemporaneamente, per osservare il comportamento cellulare a seconda del tessuto e dello stadio del ciclo cellulare. Nel DNA sono contenute tutte le informazioni necessarie per la trasmissione dell'informazione genetica e la codifica in proteine, ma questo codice genetico dà anche le direttive temporali perché avvenga la biosintesi dell'acido ribonucleico (RNA). L'analisi dell'espressione genica (Gene Expression Profile) ha la funzione di "fotografare" l'attività momentanea di una cellula o di un tessuto, in cui si troveranno alcuni geni espressi (on,

up) ed altri inespressi (off, down). In oncologia, l'uso di una tecnica siffatta rappresenta uno strumento diagnostico di grande importanza, in quanto permette di risalire al tipo di cellula che esprime uno specifico gene, qual è il suo stato e se si trova in fase proliferativa. La sua applicazione clinica, inoltre, è notevolmente più facile rispetto a tecniche antecedenti: come viene spiegato successivamente nel capitolo, soltanto in tempi recenti si sono affacciati sul panorama medico chip ad alta densità in grado di fornire simultaneamente la misura di moltissimi geni e addirittura del genoma intero, con tempi e costi destinati a diminuire sempre più. Il genoma umano si pensa essere composto da circa $20000 - 30000$ geni. Questo numero non rende immediata ragione della complessità con cui l'impronta genetica si esprime, dal momento che un singolo gene può generare diversi RNA messaggero (mRNA).

1.1 Proteomica e metabolomica

Affianco della genomica, ancora più recentemente sono comparse altre branche di scienze -omiche, come proteomica e metabolomica, accomunate dal tentativo di studiare un sistema biologico rispetto alla totalità di qualche suo tipo di struttura e funzione.

Il termine proteoma è la fusione tra proteins e genoma, coniato per la prima volta nel 1994 da Wilkins per descrivere l'intero pattern proteico espresso dal genoma a livello cellulare e tissutale. Mentre il genoma in condizioni fisiologiche in uno stesso organismo è un'entità pressoché costante, il proteoma è estremamente dinamico, in quanto l'espressione di una proteina differisce da cellula a cellula e da tessuto a tessuto in funzione del ciclo cellulare, dell'attività metabolica, del sesso, dell'età ed anche in funzione dell'ambiente. Dopo il completamento del progetto "Genoma Umano" è emersa la staticità delle informazioni generate dal solo studio dei geni che non fornisce informazioni sulla loro funzione e non è in grado di predire l'insorgenza della malattia. La ricerca basata sui profili di espressione genica è stata di grande aiuto nel fornire le informazioni relative al cambiamento di espressione globale indotto, ad esempio, da un evento canceroso; essa però non è in grado di fornire informazioni riguardanti le modificazioni post-traduzionali che avvengono a livello proteico e che sono spesso responsabili dei segnali iniziali di insorgenza della malattia. Tali tipi di informazioni sono ottenibili solo tramite uno studio dell'espressione proteica globale, quale la proteomica. Lo studio del-

la proteomica si basa sull'indagine molecolare dei trascritti e delle proteine espresse in un comparto cellulare, partendo dalla separazione (ad esempio attraverso la tecnica dell'elettroforesi) e quantizzazione di prodotti specifici e sulla ricerca dei meccanismi alla base di processi biologici includendo sia ricerche di carattere metodologico che tecnologico. Tale studio mira a identificare le proteine e ad associarle con uno stato fisiologico in base all'alterazione del livello di espressione fra controllo e trattato, permettendo di correlare il livello di proteine prodotte da una cellula o tessuto e l'inizio o la progressione di uno stato patologico. Le alterazioni a livello proteico che portano, o favoriscono, lo stato di malattia possono essere di natura quantitativa, funzionale o strutturale. A tal proposito la proteomica ha identificato proteine che possono essere utilizzate come targets diagnostici, prognostici e terapeutici, in quanto oggi la ricerca è sempre più orientata verso lo sviluppo di terapie personalizzate (*personalized medicine*), specialmente nel campo di malattie che richiedono un trattamento cronico, come nel caso del cancro. L'identificazione di questi targets, o biomarkers, è attualmente basata su singole proteine, non sempre rilevabili: lo studio proteomico richiede, infatti, il continuo sviluppo di metodi per il miglioramento delle capacità separative, della sensibilità e delle possibilità di interpretazione dei dati correlati ai segnali biologici. Le alterazioni fisiologiche/patologiche hanno effetti molteplici non solo sul proteoma, ma anche sul metaboloma e sottolineano come questi processi coinvolgano una rete modulare piuttosto che evoluzioni lineari.

Mentre la genomica e la proteomica descrivono le potenzialità dei sistemi biologici, la metabolomica ne rappresenta il funzionamento dinamico reale essendo la risultante delle alterazioni tra gene, organismo e ambiente. La metabolomica mostra come i processi metabolici regolino l'espressione dei geni e l'attività enzimatica. Inizialmente, in genetica clinica si tendeva ad associare la patologia all'alterazione di un singolo gene, oggi invece è noto che molte malattie tra cui il cancro sono multifattoriali, cioè per potersi esprimere devono coinvolgere diversi geni e diversi fattori individuali e ambientali diversi. La metabolomica consente di vedere l'evoluzione biologica delle proteine (codificate) e i processi attraverso l'analisi di molecole in stadi intermedi che influiscono nell'equilibrio delle attività cellulari dei singoli tessuti e nell'espressione fenotipica. Tale analisi è favorita non solo dalla possibilità di procedere su dati selezionati in funzione del sospetto diagnostico, ma anche nel poter raccogliere, classificare ed individuare elementi appa-

rentemente diffusi e disomogenei che possono essere ottenuti con metodiche differenziate riuscendo ad aggregarne le informazioni. Lo studio della metabolimica richiede l'applicazione di tecniche analitiche come ad esempio Gas chromatography-mass spectrometry (GC-MS) o Metabolite array, e metodi di analisi statistica come l'Analisi del discriminante. In campo clinico sia la proteomica che la metabolomica risultano essere degli approcci innovativi e rispetto alla genomica hanno il vantaggio di tener conto di più determinanti patologiche, e per questo rappresentano un buon modello per lo studio di un organismo. Purtroppo tali studi sono per lo più sperimentali, e ancora i costi elevati così come la scarsa disponibilità dei dati raccolti risultano essere uno dei principali limiti.

1.2 PCR

L'indagine genetica ha avuto un'importante accelerazione a partire dalla metà degli anni '80 del secolo scorso, quando il biochimico americano Kary Mullis inventò la tecnica della Reazione a Catena della Polimerasi (PCR). Si tratta di una metodologia di laboratorio per amplificare sequenze del DNA, con crescita esponenziale nel corso di poche ore. Il suo enorme successo è legato all'alta sensibilità (10^7 volte più preciso dei metodi precedenti), alla versatilità, alla rapidità e alla capacità di amplificare anche quantità minime di DNA. Esistono attualmente delle varianti alla classica PCR, tra cui la Real Time PCR, detta anche PCR quantitativa, sviluppata per la quantificazione in tempo reale del materiale genetico. Sinteticamente, i passaggi iniziali della Real Time PCR sono due: l'estrazione dell'RNA totale dal tessuto in analisi e la retro-trascrizione dell'mRNA totale in cDNA. Successivamente, seguono le tre fasi della PCR a temperature stabilite: la denaturazione del DNA (in cui si separano i due filamenti), l'appaiamento (annealing, quando due primers si legano alle sequenze complementari del gene target) e la copia del DNA da parte dell'enzima Taq polimerasi. Da una singola copia di una specifica sequenza, lunga poche kilobasi (kb), la replicazione permette di averne infine miliardi. Teoricamente, ad ogni ciclo si ottiene il raddoppio del numero di copie, perciò la PCR ha un fattore di amplificazione pari a 2^n dove n è il numero dei cicli; la resa effettiva, invece, è inferiore poiché dipende da fattori come la quantità dei primers e l'attività della Taq polimerasi. La peculiarità della tecnica denominata Real Time è la misurazione della resa

effettiva dell'amplificazione ad ogni ciclo, quando la reazione cresce ancora in modo esponenziale, prima che la quantità di DNA raggiunga la fase di plateau. Per fare ciò, la tecnica si serve di marcatori fluorescenti, cioè di molecole che variano la loro fluorescenza legandosi ai filamenti: la fluorescenza varia in proporzione alla quantità di amplificato. La PCR risulta, quindi, una metodologia tuttora molto valida, sebbene richieda un'alta conoscenza della tecnica e i suoi costi siano elevati. Il punto di debolezza di tale tecnica consiste nello scarso numero di geni che si possono analizzare in contemporanea, perciò risulta inadeguata per uno studio approfondito del profilo di espressione genica di un individuo, non riuscendo ad individuare in tempi rapidi quali di essi siano coinvolti in una malattia.

1.3 Microarray

Nel 1995 Schena e colleghi pubblicarono su Science l'articolo "Quantitative monitoring of gene expression patterns with a complementary DNA microarray" [Schena et al., 1995] nel quale per la prima volta si utilizzarono i cosiddetti microarray con sonde di DNA complementare. Questa tecnologia è costituita da un supporto solido, come vetro, plastica o un chip di silicio, sul quale vengono immobilizzati migliaia di filamenti di DNA a singola catena. Tali frammenti di materiale genetico, detti sonde o probes, vengono scelti da librerie genomiche, da tessuti o cellule o da raccolte di cDNA, e prima di essere disposti sull'array secondo sequenze note, sono amplificati in PCR e purificati. Quando il filamento si trova in presenza del complementare (di test) che viene marcato con un marker fluorescente come il fluorocromo, esso tenderà ad appaiarvisi grazie alla possibilità di ibridare con il cDNA. Infine, la scansione dell'array rivela la quantità di segnale di fluorescenza derivante da una specifica posizione, in modo direttamente proporzionale all'abbondanza del trascritto legato. Le sonde contenute in una singola feature hanno tutte sequenze identiche e sono mediamente $10^5 - 10^6$, mentre ogni feature ha una sequenza differente dalle altre. In un array sono presenti $10^3 - 10^6$ feature: si comprende, quindi, che con i microarray si può determinare il profilo di espressione di tutto il trascrittoma contemporaneamente e non solo di un gene alla volta come nel caso della PCR. Attualmente le tecnologie costruttive dei microarray si distinguono in due classi in base al tipo di acido nucleico usato per la composizione delle sonde: i microarray con sonde di

cDNA e quelli con sonde di oligonucleotidi.

1.3.1 Microarray di cDNA

Figura 1.1: Microarray di cDNA

Chiamati anche microarray *spotted*, questi strumenti sono costituiti da migliaia di sonde depositate su uno spot di un vetrino da laboratorio rivestito di polilisina (omo-polipeptide dell'amminoacido lisina che favorisce l'adesione cellulare al vetro) per mezzo di una testa robotizzata dotata di pennini molto sottili, in numero da 4 a 32. Essi lasciano qualche *ng* di DNA in punti distanti fra loro $120 - 250\mu m$ e di diametro $\sim 100\mu m$, disposti a formare una matrice tipicamente di 2.5x2.5cm^2; in un vetrino si trovano complessivamente $5\,000 - 10\,000$ elementi. Fissate alla base con radiazione UV, le sequenze che vengono trascritte sul supporto derivano da genoteche di *Expressed Sequence Tags*, ESTs, ovvero corte sotto-sequenze di cDNA composte da $500 - 800$ nucleotidi; esse contengono abbastanza informazione per risalire all'intero cDNA relativo. Con questa tecnica è possibile ibridare contemporaneamente due campioni fluorescenti di DNA complementare, quello di riferimento marcato Cy5 (rosso) e quello di test marcato Cy3 (verde). La scannerizzazione al computer del microarray produce una misurazione dell'espressione genica attraverso una mappa visiva -*heat map*- degli spot a cui i campioni si legano: in un elemento della matrice, la fluorescenza verde (o rossa) indica che quel gene è espresso solo nel campione noto (o in quello di prova), mentre se la fluorescenza ha una tonalità verso il giallo significa che il gene è espresso in entrambi i campioni. Se, invece, il punto resta scuro il gene non è espresso. In sintesi, lo scanner rivela il rapporto tra le intensità luminose nei due canali rosso/verde.

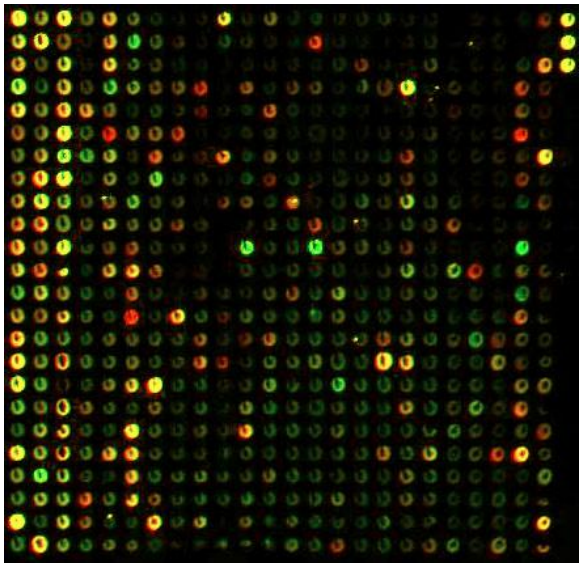

Figura 1.2: *Heat map* dell'espressione genica in un microarray di cDNA

La marcatura dei target con diverse molecole fluorescenti, d'altra parte, può produrre una differente efficienza di incorporazione. Per ottenere risultati migliori è opportuno, perciò, adottare accorgimenti quali la marcatura reciproca dello stesso campione con entrambi i fluorescenti, le repliche dei campioni di esperimenti diversi da utilizzare come controllo per la variabilità sperimentale e le repliche dell'ibridazione su vetrini diversi.
A fronte di alcuni vantaggi, rimangono difetti nella tecnica *spotted*, come i problemi legati al colore: dato che l'intensità dipende dalle quantità di sonde depositate, risulta difficoltosa la comparazione dei risultati provenienti da array differenti.

1.3.2 Microarray di oligonucleotidi

Con la seconda tipologia di microarray è consentito creare chip con una notevole maggiore densità di probe. Le sonde, in questo caso, non sono più EST bensì sono catene di oligonucleotidi a singola elica sintetizzate artificialmente e provenienti da raccolte disponibili in commercio. Appositi software estraggono sequenze sonda da basi di dati genomiche, univoche per i singoli geni, e ne ottimizzano la capacità di ibridazione: una soluzione come questa permette di non dover conservare le grandi banche di cDNA, con i relativi problemi di manutenzione.

L'azienda californiana Affymetrix è leader nella produzione dei microarray GeneChip® mediante fotolitografia e fornisce chip con catene di 25 basi. Un primo livello di oligonucleotidi monomeri è sintetizzato *in situ* sul wafer di silicio o di quarzo e ricoperto da uno strato protettore fotosensibile; utilizzando un lampo luminoso e una maschera, che permette alla luce di colpire solo i punti desiderati dell'array, tale protezione viene meno. L'array è esposto ad una soluzione di un nuovo nucleotide e le zone deprotette sono libere di legarsi alla base. Ripetendo ciclicamente queste operazioni e cambiando pattern della maschera ad ogni passaggio per lasciare scoperte le aree necessarie, si ottengono filamenti della sequenza richiesta. Può essere effettuata la sintesi anche senza utilizzare maschere ma servendosi di un Digital Micromirror Device, ovvero una matrice di micro-specchi che riflettono o meno la luce sui punti dell'array a seconda che i suoi pixel siano correttamente orientati (stato on) oppure siano ruotati con un angolo diverso (stato off): con la presente tecnica si evita la delicata preparazione delle maschere fotolitografiche. Infine il wafer viene tagliato nei chip di attuale dimensione standard di $12.8mm^2$. Ogni microarray contiene fino a 400 000 diversi oligonucleotidi e ognuno di questi è presente in milioni di copie.

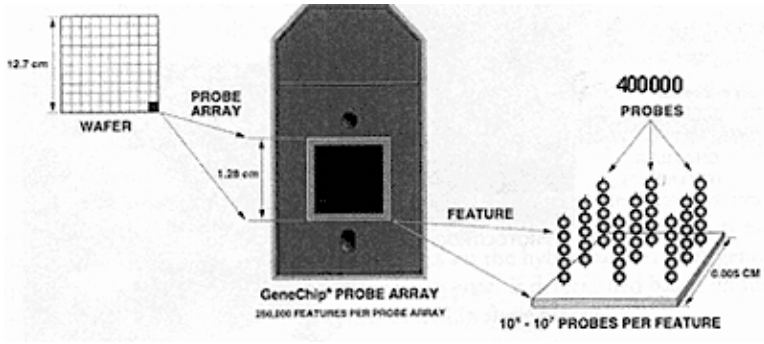

Figura 1.3: Struttura di un GeneChip® Microarray

Ciascun gene in analisi viene rappresentato sul chip da una decina di coppie di feature. A loro volta, le due feature di ogni coppia sono formate l'una da oligonucleotidi esattamente complementari alla sequenza di riferimento (sonde PM, perfect match), l'altra invece presenta una sola base differente dalla sequenza target (sonde MM, mismatch), solitamente la base intermedia, la 13ª. Il DNA marcato con un composto fluorescente riuscirà, perciò, ad ibridare del tutto con le sonde PM ma troverà difficoltà con le sonde MM. Una soluzione come questa aiuta a discriminare tra rumore di fondo e segnale nella

fase di lettura: il sistema di acquisizione dell'immagine sottrae l'intensità luminosa dell'ibridazione della sonda MM da quella della contigua sonda PM per determinare il valore dell'intensità assoluta.
Una seconda azienda californiana, la Agilent Technologies, ha introdotto un modo alternativo per produrre microarray, portando il numero di basi per oligonucleotide da 25 a 60 e realizzando la sintesi *in situ* con stampa a getto di inchiostro senza contatto con il supporto. La progettazione è completamente controllata da software, la riproducibilità delle feature è alta e la maggior lunghezza dei filamenti aumenta la specificità e la sensibilità poiché il computer seleziona le sonde con una composizione di basi ottimale e in contemporanea con ridotto potenziale di cross-ibridazione: questi sono i principali vantaggi dei microarray Agilent.

1.3.3 Acquisizione dei dati

I risultati provenienti da microarray multipli non sono immediatamente paragonabili, dato che le misure di espressione eseguite con diverse tecniche di microarray e su diverse piattaforme non sono direttamente comparabili. Ciò rappresenta un fattore penalizzante per queste tecnologie, ma da tempo gruppi di lavoro e società, come la Functional Genomics Data Society, stanno lavorando per assicurare la standardizzazione mondiale delle procedure di archiviazione e scambio dei dati. Tra i tanti, sono stati messi a punto i progetti MAGE (MicroArray Gene Expression), in particolare il recente MAGE-TAB attualmente in vigore, che mirano a fornire uno standard per la rappresentazione dei dati di espressione di microarray in grado di facilitare lo scambio di informazioni tra sistemi di dati diversi. La procedura di normalizzazione rientra nelle suddette misure da applicare ed è un'importante operazione da effettuare per correggere i bias sistematici e casuali dei dati e per rimuovere la parte di segnale che può oscurare l'informazione biologica rilevante. Tuttavia, le tecniche esistenti facilmente intaccano anche la parte di segnale utile (fenomeno chiamato *tail flattening*), perciò conviene usare metodi non invasivi.

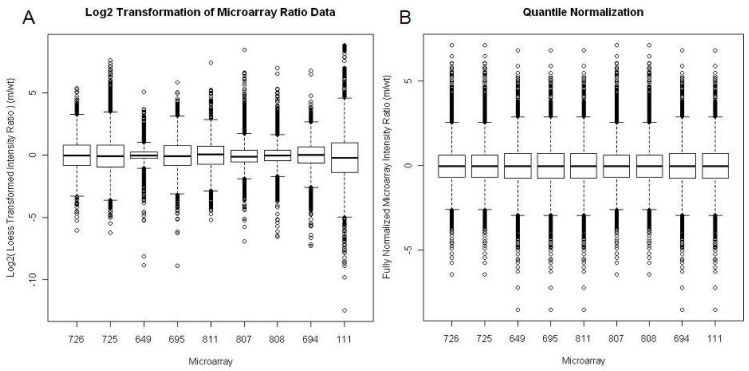

Figura 1.4: Normalizzazione e *tail flattening* dei dati

Capitolo 2

Metodi di classificazione e teorie dei network

In letteratura sono presenti oramai molti studi sulla ricerca di signature significative di ridotta dimensionalità riguardanti i dati di gene expression e, proprio per questo, anche molti strumenti mutuati dalla statistica sono stati applicati per interpretare la mole notevole dei risultati. Da una parte, vi sono ricerche che mostrano la possibilità di ottenere buone performance di classificazione attraverso l'utilizzo di determinati modelli, mentre dall'altra si oppongono studi che sostengono i limiti, e quindi l'insufficienza, di una corretta predizione tramite classificatori statistici. L'attuale lavoro di tesi vuole corroborare il tentativo di quel filone di ricerca che sostiene la vantaggiosa applicazione di metodi di *class prediction* ai profili di espressione genica.

Nell'affronto del problema della classificazione, in casi come la distinzione sano/malato, si presentano tre criticità:

- l'identificazione di nuove e sconosciute classi di tumori. È il caso del *Unsupervised learning* (e.g. *cluster analysis*);
- la classificazione dei campioni in classi note. È il caso del *Supervised learning* (e.g. *discriminant analysis*);
- l'identificazione di geni marker con cui caratterizzare le differenti classi di tumori. Si tratta della selezione di variabili.

Dopo un'introduzione alla teoria dei network, in questo capitolo sono illustrati alcuni tra i più noti metodi statistici di apprendimento supervisionato e di validazione applicabili nella ricerca di predittori per malattie genetiche quali le molteplici forme di cancro.

2.1 Teoria dei network

Un network è un sistema (fisico, biologico, informatico, economico, sociale, ecc.) i cui elementi presentano interazioni di qualche tipo. La schematizzazione di un network si realizza principalmente per mezzo di un grafo, che rappresenta elementi e relative interazioni del sistema con nodi (detti anche vertici) e connessioni tra i nodi (link). Un modo matriciale per descrivere un network composto da N nodi è la matrice di adiacenza $N \times N$, nella quale ogni posizione $A_{i,j}$ vale 1 quando esiste il link tra elemento i ed elemento j (oppure vale $w_{i,j}$ se il network è pesato).

Tra le misure di centralità, che indicano l'importanza dei nodi all'intero del network, si trova il grado di connettività K, ovvero il numero di link entranti (o uscenti) da un nodo; si ottiene facilmente come somma delle colonne (o delle righe) della matrice di adiacenza. Questa grandezza può anche essere vista come il numero di link da rimuovere perché un certo nodo sia disconnesso dal resto del grafo. Se $K = 1$ per un nodo, allora si dice che tale nodo è pendente (*leaf*). La progressiva rimozione dei pendenti porta a rimanere con un network circolare (*core*) oppure ad eliminare l'intero network.

Un'altra misura di centralità è la closeness, definita come l'inverso del valor medio della distanza di un nodo dagli altri:

$$Cl_i = \frac{N-1}{\sum_j P_{ij}} \tag{2.1}$$

con P_{ij} = shortest path del nodo i verso gli altri nodi. Quanto più un nodo è meno distante dagli altri, tanto più viene considerato importante. Se la distanza tra n_i e n_j è infinita, significa che il grafo non è connesso e che i esistono sottografi distinti. Questi sottografi sono detti anche componenti connesse del grafo, all'interno delle quali un vertice è collegato ad ogni altro tramite cammini, mentre non lo è con vertici di altre componenti.

Una terza misura di centralità è la betweenness centrality, definita come il numero di shortest paths che passa per il nodo i-esimo da ogni vertice

verso tutti gli altri:

$$BC_i = \frac{\sum_{m,n} P_{m,n}(i)}{\sum_{m,n} P_{m,n}} \tag{2.2}$$

Un nodo con bassa connettività potrebbe comunque avere grande betweenness centrality nel caso in cui si trovi in una posizione dove il flusso delle interazioni nel network è elevato, come nei nodi che collegano due blocchi (bridge o cutpoint).

2.2 Artificial Neural Networks

Il termine Artificial Neural Networks nel tempo ha incluso al suo interno un gran numero di modelli e di metodi di apprendimento, tra i quali il più utilizzato prende il nome di Single layer perceptron. Nel complesso, si tratta di modelli statistici non lineari. Una rete neurale è un modello di regressione o di classificazione a due fasi. Per quanto riguarda la regressione generalmente c'è un unico output finale Y_1, anche se non vi è nessuna difficoltà per la rete a controllare risposte multiple. Per una classificazione a K-classi ci sono K unità a monte, con la k-esima unità che determina la probabilità della classe k. Ci sono K misure nominate Y_k, $K = 1, ..., K$, ognuna catalogata 0–1 per la k-esima classe. Le features derivate Z_m sono ottenute da combinazioni lineari degli input, e l'output Y_k è costruito come funzione delle combinazioni lineari degli Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0_m} + \alpha_m^T X),\ m = 1, ..., M, \\ T_k &= \beta_{0_k} + \beta_k^T Z,\ k = 1, ..., K, \\ f_k(X) &= g_k(T),\ k = 1, \ldots, K, \end{aligned}$$

dove $Z = (Z_1, Z_2, ..., Z_M)$ e $T = (T_1, T_2, ..., T_K)$). La funzione di attivazione $\sigma(v)$ è generalmente scelta tra la funzione sigmoidea e le funzioni gaussiane a base radiale. A volte i diagrammi delle reti sono rappresentati con un'unità di bias aggiuntivo che ha effetto su ciascuna unità del layer nascosto e finale. La funzione di output $g_k(T)$ permette una trasformazione finale del vettore di output T. Per la regressione, la funzione $g_k(T)$ si sceglie della forma di una funzione softmax:

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}. \tag{2.3}$$

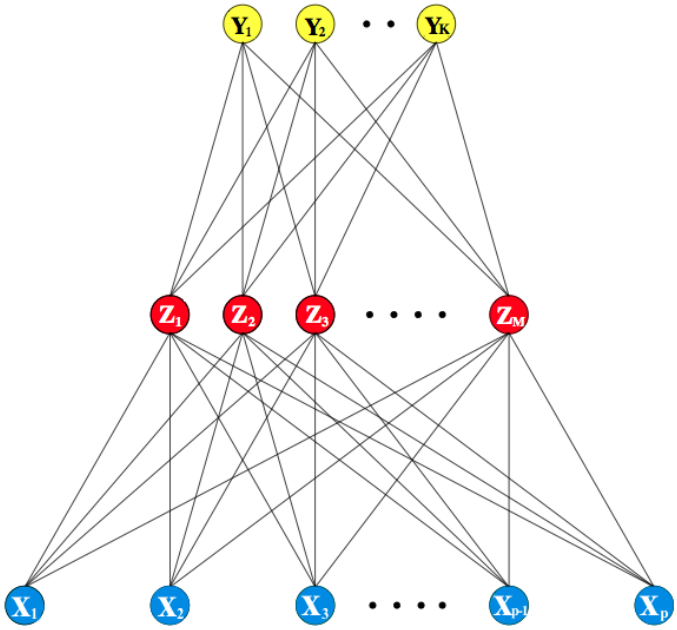

Figura 2.1: Schema di una rete neurale a singolo layer nascosto

Le unità invece che calcolano le feature Z_m sono chiamate *hidden layers* in quanto i valori Z_m non sono direttamente osservabili. Come accennato sopra, ci possono essere più di un livello nascosto; si possono pensare gli Z_m come un'espansione degli input X. Così la rete neurale sarebbe un modello lineare standard, utilizzando queste trasformazioni come input. C'è, però, un potenziamento al di là delle tecniche di espansione-base in cui i parametri delle funzioni base sono appresi direttamente dai dati. Si osservi come nel caso in cui la funzione σ fosse la funzione identità il metodo diventerebbe un metodo lineare nelle variabili in input. Per questo una rete neurale viene considerata una generalizzazione non lineare di modelli lineari, sia per la regressione che per la classificazione. La comodità della funzione σ rispetto anche ad altri metodi è che è una funzione relativamente semplice con tre parametri liberi. La rete neurale ha parametri non noti detti pesi, di cui si cercano i valori che adattino al meglio il modello con i dati di training. Chiamando θ il set completo dei pesi e $R(\theta)$ l'errore che rappresenta la misura del fit, generalmente si tenta di minimizzare $R(\theta)$ con la discesa del gradiente, qui chiamata *back-propagation*. I vantaggi di questa tecnica sono la sua natura semplice e locale. Nell'algoritmo relativo, ogni unità nascosta passa

e riceve e dà informazioni solo da e verso unità con cui è connessa. Anche il nome Neural Netowrk non è casuale ma deriva dal fatto che fu sviluppato per le prime volte come modello del cervello umano. Ogni unità rappresenta un neurone e le connessioni le sinapsi, rappresentate matematicamente dalle funzioni σ e g_m, che poi sono adattate ai processi di modellizzazione statistica non lineare, con le dovete modifiche necessarie ai fini del metodo.

2.3 k-Nearest Neighbors

Il metodo Nearest neighbors si basa sul confronto tra dati all'interno di un training set; in particolare, preso una campione di dati –come può esserne uno di mRNA–, con profilo di espressione genica $x = (x_1, \ldots, x_p)$, la miglior stima per il k-nearest neighbor è definita nel seguente modo:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x \in N_k(x)} y_i \tag{2.4}$$

dove con $\hat{Y}(x)$ si denota una buona previsione dell'output che si ottiene dal dato X, e con $N_k(x)$ l'intorno di x definito dai k punti x_i più vicini tra quelli del training set preso in esame. Il concetto di vicinanza necessita, e in un certo senso induce, la definizione di una funzione distanza che permetta di confrontare più dati e quindi di ottenere determinati qualificatori tramite i quali verificare l'andamento del metodo considerato. In generale, con le convenzioni già utilizzate in precedenza, la distanza tra due campioni di mRNA con espressione genica $x = (x_1, ..., x_p)$ e $x' = (x'_1, ..., x'_p)$, esprime la correlazione tra le loro due espressioni ed è definito nel seguente modo:

$$\pi_{x,x'} = \frac{\sum_{j=1}^{P} (x_j - \bar{x})(x'_j - \bar{x})}{\sqrt{\sum_{j=1}^{P} (x_j - \bar{x})^2} \sqrt{\sum_{j=1}^{P} (x'_j - \bar{x}')^2}} \tag{2.5}$$

dove $\bar{x}$ rappresenta un valore medio rispetto a quelli presenti nel singolo profilo di espressione genico e in generale nell'intero dataset. Determinare il k-nearest nighbor permette di proseguire nella classificazione del test set di osservazioni in base al training set usato come campione. Prima di tutto, per ogni elemento appartenente al test set individua le k osservazioni più vicine presenti nel set prova e di seguirò in base a questo confronto predice la classe con voto di maggioranza, cioè più precisamente sceglie la classe con

più elementi in comune tra le k possibilità. Il numero k di interni viene determinato in base ad un controllo incrociato, in cui ogni osservazione del test di prova viene trattata a sua volta come se fosse nel test set: viene così calcolata e in definitiva classificata la sua distanza da ogni altro elemento del set di prova, ovviamente escludendo se stesso. La classificazione viene poi confrontata con le vere classi per produrre un certo tasso di errore, che viene usato come indice di correttezza della valutazione svolta. Il processo viene ripetuto per un certo numero di k, e il valore di k per cui questa stima del tasso di errore rispetto al controllo incrociato è la più piccola viene mantenuto per essere utilizzato nello studio del test set.

2.4 Support Vector Machine

Sotto il nome di Support Vector Machines vanno quelle tecniche che trattano il caso di due classi non separabili i cui elementi si sovrappongono. In particolare si tratta di classi i cui *boundaries* non siano lineari. Le Support Vector Machines (SVM) affrontano il problema costruendo un boundary lineare in una versione trasformata dello spazio degli oggetti in questione. Una SVM è definita da un Support Vector Machine Classifier (SVMC). Per calcolare il SVC occorre trovare e massimizzare il margine

$$M = 1/||\beta|| \tag{2.6}$$

dell'iperpiano

$$\{x : f(x) = x^T\beta + \beta_0 = 0\} \tag{2.7}$$

che separi le N coppie di dati di training $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ con $x_i \in R^p$ e $y_i \in \{-1, 1\}$ e dove β è un vettore normalizzato unitario. Si cerca un iperpiano che dia il più grande margine tra i punti del piano di classe -1 e 1. Il problema di ottimizzazione è rappresentato come

$$\begin{aligned} &\min_{\beta,\beta_0} M \\ y_i(x_i^T\beta + \beta_0) \geq 1,\ &i = 1, \ldots, N \end{aligned}$$

Definendo la variabile di scarto (slack variable) ξ, il vincolo dell'equazione precedente si può modificare come:

$$y_i(x_i^T\beta + \beta_0) \geq M(\xi_i) \tag{2.8}$$

Così viene misurata la sovrapposizione in distanza relativa, che cambia con la larghezza relativa del margine M.

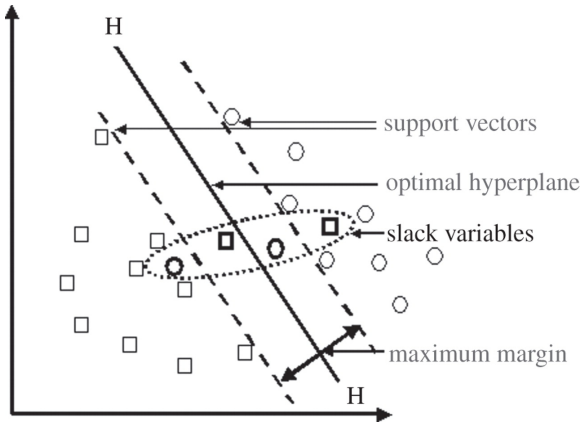

Figura 2.2: Support Vector Classifier nel caso non separabile

Per calcolare il SVMC secondo questa formulazione, occorre risolvere un problema quadratico con condizioni di disuguaglianza lineare, pertanto un problema di ottimizzazione convessa. Utilizzando il metodo dei moltiplicatori di Lagrange, si può arrivare alla seguente soluzione per β:

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \tag{2.9}$$

dove $\hat{\alpha}_i$ sono i vettori di supporto (SV). I SV sono diversi da zero solo per quelle osservazioni per le quali le condizioni imposte in precedenza vengono soddisfatte esattamente. Trovate quindi le soluzioni $\hat{\beta}_0$ e $\hat{\beta}$, la funzione di decisione può scriversi come

$$\hat{G}(x) = \text{sign}[x^T\hat{\beta} + \hat{\beta}_0]. \tag{2.10}$$

Il SVMC trova boundaries lineari nello spazio degli oggetti in questione. Trattandosi di un metodo lineare, il procedimento può essere reso più flessibile allargando lo spazio degli oggetti con espansioni di tipo polinomiale o

spline. Di solito con boundaries lineari nello spazio allargato si ottiene una migliore separazione per le classi nel training; tali boundaries si traducono in boundaries non lineari nello spazio d'origine. Una volta selezionate le funzioni base $h_m(x)$, $m = 1, \ldots, M$ il procedimento è uguale a prima. Si adatta il SVC servendosi delle features di input $h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$, $i_1, \ldots, N$ e si produce una funzione non lineare $\hat{f}(x) = h(x)^T\hat{\beta} + \hat{\beta}_0$, e il classificatore è nuovamente $\hat{G}(x) = \text{sign}(\hat{f}(x))$. Il classificatore SVM è un'estensione di questa idea, dove la dimensione dello spazio allargato può essere molto grande o addirittura infinita.

Negli spazi multidimensionali, per i quali il campionamento risulta più arduo con l'aumentare della dimensione, ci si aspetterebbe che le performances del SVMC aumentino con le dimensionalità del kernel di supporto. I risultati di alcune simulazioni confermano che il SVMC presenta sicuramente risultati migliori di un semplice SVC, in quanto nello spazio degli oggetti originale un semplice iperpiano non è in grado di separare le classi. Tuttavia non è vero che con l'aumentare della dimensione del kernel (i.e., grado del polinomio) aumenti anche la prestazione del SVMC. Molto dunque dipende dalla scelta del kernel rispetto alla classe degli oggetti in questione.

In generale, le SVM si possono estendere a problemi a molte classi risolvendo ripetutamente dei problemi a due classi. Per ogni coppia di classi si costruisce un classificatore; il classificatore finale è quello più importante. In alternativa si può usare la loss function multinomiale con un kernel appropriato. Evidenze empiriche suggeriscono che le SVM abbiano delle buone performances in molti problemi reali di apprendimento. Vi è inoltre una connessione tra le SVM e la structural risk minimisation.

2.5 Analisi del discriminante lineare

Con grande frequenza, nell'ambito della classificazione si incontrano dati i cui elementi presi singolarmente non producono buone stime poiché da soli non veicolano sufficiente informazione sulla classe di appartenenza (1-dimensionale). Come graficato in Figura 2.3, prendendo un set di dati con feature $\mathrm{x} = (\mathbf{x_i}, \ldots, \mathbf{x_N})$, $i = 1, \ldots, N$, la proiezione dei dati sull'asse della feature i-esima produrrebbe una stima con un certo *overlap*, e così succederebbe per ognuna delle caratteristiche considerate singolarmente. L'analisi

del discriminante lineare, invece, ha la sua forza nella classificazione *a coppie* di dati, riuscendo in 2D a risolvere l'*overlap* che si verifica in 1D.

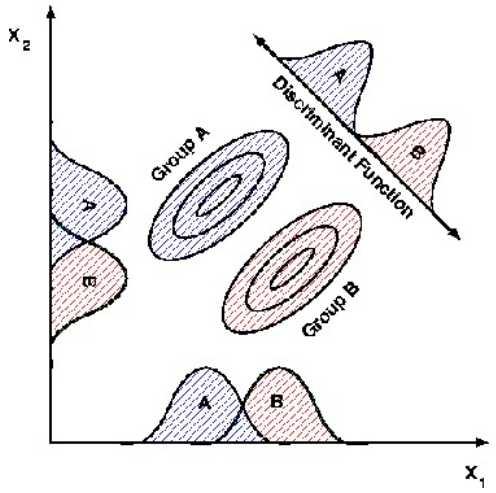

Figura 2.3: Support Vector Classifier nel caso non separabile

Nata negli anni '30 del secolo scorso, l'analisi del discriminante di Fisher, nella sua forma lineare, è basata sulla ricerca di combinazioni lineari xa dei dati $\mathbf{x} = (x_1, \ldots, x_P)$ che massimizzino il rapporto tra le medie della classe e la varianza della stessa classe lungo la direzione a. Per una matrice di dati X di dimensione $n \times p$ (nell'ipotesi che $P \gg N$, con P = numero di feature e N = numero di osservazioni) che funge da learning set, la combinazione lineare Xa delle colonne di X è il rapporto tra la matrice di varianza tra le classi (*between*) e la matrice di varianza nelle classi (*within*) ed è dato da

$$\frac{\boldsymbol{a}'B\boldsymbol{a}}{\boldsymbol{a}'W\boldsymbol{a}} \tag{2.11}$$

dove B e W sono le matrici $P \times P$ delle somme dei quadrati dei *between group* e dei *within group*:

$$B = \frac{1}{n-1}\sum_{k=1}^{K} n_k(\mu_{Y_i} - \bar{X})(\mu_{Y_i} - \bar{X})^T$$

$$W = \frac{1}{n-1}\sum_{i=1}^{n} (\bar{X} - \mu_{Y_i})(\bar{X} - \mu_{Y_i})^T$$

con n_k numero di osservazioni appartenenti alla classe k. I valori estremi di $\boldsymbol{a}'B\boldsymbol{a}/\boldsymbol{a}'W\boldsymbol{a}$ si ottengono dagli autovalori e dagli autovettori di $W^{-1}B$.

La matrice $W^{-1}B$ ha al massimo $s = min(K-1, p)$ autovalori non nulli, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_s$, con i corrispondenti autovettori linearmente indipendenti $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_s$. Le variabili del discriminante sono definite come $u_l = \mathbf{x}\mathbf{v}_l, l = 1, ...s$ e in particolare $\mathbf{v}_1$ massimizza $\boldsymbol{a}'B\boldsymbol{a}/\boldsymbol{a}'W\boldsymbol{a}$. Per una osservazione $\mathbf{x} = (x_1, ..., x_p)$ sia

$$d_k(\mathbf{x}) = \sum_{l=1}^{s}((\mathbf{x} - \bar{\mathbf{x}}_\mathbf{k})\mathbf{v}_l)^2 \tag{2.12}$$

l'equazione che indica la sua distanza euclidea (al quadrato), in termini delle variabili del discriminante, dal vettore $1 \times p$ delle medie $\bar{\mathbf{x}}_k$ per il learning set. La classe predetta per l'osservazione $\mathbf{x}$ è

$$C(\mathbf{x}, L) = \text{argmin}_k d_k(\mathbf{x}) \tag{2.13}$$

cioè, la classe il cui vettore medio è più vicino a $\mathbf{x}$ nello spazio delle variabili del discriminante.

2.6 Cross-validation

I metodi *Cross-validation* sono tra i metodi più semplici e largamente usati per stimare errori di predizione all'interno degli algoritmi di apprendimento.
In particolare, il metodo *K-fold Cross-validation* prevede la divisione del set di dati in k partizioni aventi approssimativamente le medesime dimensioni, utilizzando $k-1$ partizioni per impostare il modello ai fini dell'apprendimento e la rimanente per testarlo. Nel dettaglio, consideriamo una funzione indicizzazione $\sigma : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ che indica la partizione assegnata all'osservazione i, con $i = 1 \ldots, N$; denotiamo inoltre con $\hat{f}^{-k}(x)$ la funzione *fitted*, calcolata rimuovendo la k-esima parte dei dati. La stima dunque dell'errore di predizione risulta

$$CV(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\sigma(i)}(x_i)) \tag{2.14}$$

dove $L(y_i, \hat{f}^{-\sigma(i)}(x_i))$ è la funzione di perdita per la misurazione degli errori tra la variabile y_i e $\hat{f}^{-\sigma(i)}(x_i)$; mentre invece l'errore vero si calcola come la

media aritmetica delle singole stime E_i:

$$E = \frac{1}{k}\sum_{i=1}^{k} E_i \tag{2.15}$$

Dato poi un set di modelli $f(x, \alpha)$ indicizzato dal parametro α di tuning, denotiamo con $\hat{f}^{-k}(x, \alpha)$ l'α-esimo modello ottenuto rimuovendo la k-esima partizione di dati. Dunque definiamo

$$CV(\hat{f}, \alpha) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\sigma(i)}(x_i, \alpha)) \tag{2.16}$$

Tale funzione fornisce una stima della curva degli errori, da cui possiamo ricavare il parametro $\hat{\alpha}$ che lo minimizza. Il nostro modello finale sarà perciò $f(x, \hat{\alpha})$.
Solitamente si scelgono i casi $K = 5$, $K = 10$, per i quali il K-fold Cross-validation stima l'errore atteso, poiché i training set in ogni partizione si differenziano molto dal training set originale.
In questo caso, la cross-validation ha varianza limitata ma il bias può costituire un problema in base a come la performance del metodo di apprendimento varia con la grandezza del trading set. Il caso $K = N$ è noto come *leave-one-out* Cross-validation, caso in cui $\sigma(i) = i$, e per la i-esima osservazione la funzione è calcolata utilizzando tutti i dati tranne l'i-esimo. La stima che fa è approssimativamente unbiased per l'errore di predizione atteso, ma può essere affetta da grande varianza a causa degli N training set molto simili tra loro.

Tra i classificatori qui esposti, l'analisi del discriminante fornisce boundaries lineari nello spazio dei parametri, mentre gli altri forniscono boundaries non lineari. La scelta di sviluppare un algoritmo basato sul discriminante è dovuta anche a queste proprietà, poiché tali classificazioni sono facilmente interpretabili a livello dei dati di partenza. Nel caso di dati biologici, una separazione lineare si traduce nel valutare se i parametri (geni, proteine, ecc.) sono più o meno espressi nei vari casi considerati, permettendo di interpretare i meccanismi biologici sottostanti. La comprensione dei risultati di metodi che producono boundaries non lineari nello spazio dei parametri è invece impossibile, fornendo quindi una cosiddetta *black-box* che può rag-

giungere buone performance statistiche ma senza comprendere i meccanismi soggiacenti.

Capitolo 3

Descrizione dell'algoritmo di classificazione

Un primo elemento caratterizzante questo lavoro di tesi sta nell'utilizzo del discriminante quadratico come classificatore dei risultati dell'espressione genica. Esso si può trovare comunemente implementato nei pacchetti di statistica e *supervised learning* dei principali software di analisi dati. Tuttavia, la considerevole mole dei database trattati ha un costo computazionale gravoso ed è del tutto insopportabile anche da parte di programmi professionali su *workstation*. Così, è stato necessario procedere con la scrittura *ex nihilo* di un software che permettesse un efficiente esecuzione dell'algoritmo in tempi il più ridotti possibile; il linguaggio prescelto per notevoli qualità di potenza e di versatilità è stato il C++. È stato possibile programmare con la possibilità di sfruttare i multipli core eventualmente a disposizione sui server per un calcolo simultaneo e parallelo. La seguenti sezioni illustrano:

- il codice sorgente che calcola il discriminante, realizzato in collaborazione con il dott. Giuseppe Levi del dipartimento di Fisica presso l'università di Bologna, con particolare sottolineatura sui miglioramenti e sulle ottimizzazione specifici di questo lavoro di tesi;
- le operazioni di interpretazione dei risultati, eseguite in MATLAB, intraprendendo un approccio a network nel tentativo di trovare il minor numero di geni possibile che ottenga la massima performance di *class prediction*.

Se la prima parte dell'elaborazione richiede l'utilizzo di macchine ad alte prestazioni (High Performance Computer, *HPC*), la seconda si svolge in tempi rapidi su un comune Personal Computer.

3.1 Algoritmo QDA (C++)

Lo scheletro del codice è costituito dalle seguenti macro-sezioni:

- lettura dei dati e delle relative label;
- calcolo della posizione media dei gruppi di label sul piano determinato dall'espressione genica e classificazione delle coppie di geni in base alla posizione che occupano (ovvero alla distanza dai gruppi di label);
- ordinamento dei vettori dei risultati;
- creazione dei file di output.

L'elaborazione di questi dati rientra nei problemi ad alta dimensionalità, ovvero quelli in cui il numero di geni espressi P è molto maggiore del numero di campioni N in cui essi vengono osservati ($P \gg N$). E di conseguenza la complessità di un tale algoritmo cresce come $O(N \times P^2)$; trattando mediamente con $N \simeq 10^2$ e con $P \simeq 10^3 - 10^4$, si osserva che il costo computazionale è circa $10^8 - 10^{10}$. Poiché in condizioni standard qualunque computer, benché potente esso sia, non riesce ad affrontare il calcolo complessivo, è stato necessario intervenire con la parallelizzazione delle operazioni più onerose per il programma. Nel presente caso, si è trattato di compiere una *data division* in ambito multi-thread, assegnando ai processori (o ai nodi di un cluster) una suddivisione dei dati da analizzare. La programmazione è stata svolta in modo da ottimizzare l'uso della memoria cache attraverso accessi contigui alla memoria.

3.1.1 Librerie OpenMP, Boost e TCLAP

Per ampliare le potenzialità e la facilità d'uso del codice sono state utilizzate importanti librerie aggiuntive. Esse sono: Boost, TCLAP e OpenMP.

Librerie Boost

Le librerie Boost sono sviluppate e raccolte come librerie di alta qualità che si affiancano a quelle standard del C++ per aumentarne la produttività.

Come si legge direttamente dal sito web ufficiale, esse "velocizzano lo sviluppo iniziale, hanno risultati con bug minori, riducono la necessità di riscrivere cose già implementate e tagliano i costi di mantenimento a lungo termine". È sufficiente includere gli appositi header files e usare la direttiva *using namespace boost* per accedere alle funzioni desiderate delle librerie. Nel codice in esame, queste librerie sono utilizzate solo nel caso in cui il programma sia eseguito su server cluster e si fa uso esclusivamente della funzione *boost::mpi::all_reduce*, che unisce i valori appartenenti a ogni singolo processo del cluster in un unico valore disponibile a tutti i processi.

Libreria TCLAP

La TCLAP è una piccola libreria che consente la definizione e l'inserimento di argomenti direttamente da riga di comando; il suo impiego rende possibile, al momento del lancio dell'esecuzione, passare argomenti al calcolatore, quali: il nome del file di dati in input, la percentuale di risultati da salvare, l'ordine in cui salvarli, ecc. Le istruzioni relative alla libreria TCLAP compaiono nelle prime righe del codice, vista la rilevanza delle informazioni necessarie al programma che esse veicolano. Nel caso in cui l'utente finale non immettesse alcun argomento all'eseguibile, la sintassi TCLAP prevede anche l'inserimento di un valore default per ogni variabile.

OpenMP Application Program Interface

Le librerie OpenMP sono costituite da direttive al compilatore, librerie di routine e variabili d'ambiente per la programmazione parallela del flusso di lavoro. Le direttive utilizzate nel codice sono le seguenti:

- *pragma omp parallel*: con questa istruzione si richiede esplicitamente che il compilatore parallelizzi il segmento di codice che segue racchiuso da parentesi graffe. A tale sintassi possono essere aggiunte delle clausole come *private* e *shared*, con relativi nomi di variabili le quali saranno rispettivamente private e condivise tra ogni thread;

- *pragma omp for*: questa istruzione istruisce il compilatore a distribuire il ciclo *for* nel gruppo di thread che incontrano il costrutto, in modo da ripartire il carico dei calcoli ai vari processori a disposizione. Ad essa seguono eventuali clausole, tra cui:

  - *schedule (type)*, dove si specifica come le iterazioni vengono suddivise tra i thread. Il tipo *(dynamics, n)* divide la ripetizione del ciclo in pacchetti di dimensione n fino al completamento di tutto il lavoro;
  - *reduction (operator:list)*, che realizza una riduzione sulle variabili in *list* secondo l'operatore specificato. In ciascun thread è creata una copia privata di ogni variabile, la quale alla fine del ciclo viene ricombinata con le altre usando l'operatore dichiarato, ottenendo un valore unico e condiviso.

- *pragma omp critical*: si specifica che le operazioni lì racchiuse devono essere eseguite un thread per volta;

- *pragma omp barrier*: rappresenta un punto di sincronizzazione, dove si attende che ogni thread arrivi in quel punto prima di procedere nell'esecuzione.

3.1.2 Lettura dei dati

In accordo con gli standard che regolamentano i risultati delle analisi con microarray, i dati che provengono dai profili di espressione genica si presentano tabulati in file di testo semplice, con i geni ordinati per righe e i campioni per colonne. La prima azione che il codice deve svolgere è la lettura delle etichette che determinano a quale delle due (o più) classi appartiene il campione (nei casi analizzati si tratta di pazienti sani o affetti da cancro, spesso segnati con 0 e 1); esse si possono trovare sia numeriche che alfabetiche che alfanumeriche. La versione iniziale del codice, la versione beta, consentiva esclusivamente la lettura di etichette corrispondenti alla cifra indicante la classe, e la prima di esse doveva essere 0. Questo limite è stato superato attraverso l'uso della funzione *strtok* della Standard library <cstring>, la quale scorre la prima riga finché non trova il token stabilito - quindi nel caso presente di tabulazione in tabulazione - memorizzando ogni label incontrata. Questo *parsing* viene effettuato due volte: la prima per osservare quanti tipi di etichette sono presenti e inserirli in una variabile *string* del contenitore associativo *set*, che include soltanto elementi unici; la seconda volta si svolge il confronto (funzione *string::compare*) di ogni label con le classi appena individuate. Infine vengono contati i campioni per classe. Più semplice risulta invece la lettura dei dati veri e propri, per la quale

è sufficiente acquisire ogni riga con la funzione *getline*, sempre dell'header string.h.

3.1.3 Struttura dell'algoritmo

Prima di svolgere i calcoli, avviene l'allocazione di memoria per le matrici dei dati, dei dati medi, delle medie dei quadrati e delle sigma dei dati medi. Queste grandezze, in realtà, sono vettori di puntatori: tale soluzione è alquanto vantaggiosa poiché fa guadagnare maggior velocità nell'esecuzione delle operazioni, dal momento che alle matrici viene passato soltanto l'indirizzo in memoria dell'oggetto e non l'oggetto intero.

Dentro al primo grande ciclo *for* che scorre tutte le righe dei dati, si trovano i cicli interni in cui vengono calcolate le deviazioni standard al quadrato σ_j^i, espresse come differenza tra la media dei quadrati e il quadrato della media, e le medie μ_j^i *leave-one-out* per ogni elemento, con i indice del gruppo e j indice della coppia. Terminati questi calcoli, sulla shell compare il numero di combinazioni eseguite tra ogni dato della matrice: esso risulta da *(numero di colonne)*×*(numero di righe)*^{2}*/2*.

Il codice prosegue con la prima parte scritta per esecuzione parallela che comprende il calcolo effettivo del discriminante quadratico. Le direttive *prama omp parallel* e *pragma omp for* aprono due cicli *for* annidati, per i due geni delle coppie, dove si individuano le coordinate dei geni di cui si calcola la distanza e, a seconda che essa sia minore per una classe o per un'altra, la si "colloca" nel gruppo più vicino, ovvero si aumenta lo *score* di quella classe, ad indicare la performance di classificazione dell'algoritmo per quella coppia di geni. Con questi due cicli *for* annidati si svolge una delle parti più gravose del programma poiché il numero di coppie su cui si svolgono i calcoli è *(numero di geni) + (numero di geni)*×*(numero di geni -1)/2*, cioè si somma il numero di coppie uguali con la matrice triangolare superiore (o inferiore) dei geni. Segue poi la direttiva *pragma omp critical* dove ogni processo inserisce le performance (di singola classe e totale) delle coppie di geni uguali e delle coppie di geni differenti nelle rispettive variabili vettoriali: *singlegene_ results* e *results*. Nel caso il programma sia lanciato su un cluster, sono definite nelle righe successive del codice alcune operazioni per fare in modo che il lavoro dei diversi computer (nodi) sia raccolto in modo corretto nei vettori dei risultati. Alla fine di questo segmento si trova la funzione *all_ reduce()*, la sola utilizzata delle librerie Boost per rendere

condiviso a tutti i processi valori unificati di alcune variabili, ottenuti dalla combinazione dei loro singoli valori.

Un'ultima importante aggiunta al nucleo della sezione dei calcoli ha permesso di ridurre i tempi computazionali: si tratta dell'utilizzo di una funzione lambda, novità introdotta nel C++11. Essa è stata inserita all'inizio del ciclo *for* sul secondo gene delle coppie, rendendo argomento della funzione tutto il calcolo successivo (utilizzando la sintassi [&](){*...calcoli...*}). Nelle parentesi quadre sono "catturate" tramite indirizzo tutte le variabili finora presenti nel codice, mentre tra le parentesi tonde la lista dei parametri rimane vuota.

3.1.4 Sorting parallelo

Una volta ottenuti i vettori *singlegene_results* e *results* dall'unione dei risultati dei singoli threads, occorre procedere all'ordinamento degli stessi. Questa operazione è piuttosto onerosa a causa delle dimensioni dei vettori, in particolare di quello più voluminoso contenente i risultati delle coppie di geni differenti. Proprio per ciò, si è scelto di utilizzare il semplice algoritmo *sort()* già implementato in C++ per il sorting di *singlegene_results*, vettore non particolarmente esteso, invece un nuovo algoritmo è stato scritto per l'ordinamento degli elementi di *results*. Esso, infatti, viene svolto in parallelo per dividere il carico di lavoro sui thread.

I risultati delle coppie uguali possono essere ordinati seguendo l'ordine progressivo dei numeri dei geni (1-1, 2-2, 3-3 e così via) oppure può essere seguito l'ordine per la performance del gruppo 1, del gruppo 2 o per la performance totale, in base alla volontà dell'utente che esegue il software. Il sorting viene effettuato accendendo agli estremi del vettore con le funzioni *vector::rbegin()* e *vector::rend()*, le quali invertono l'ordine di accesso di *sort()*, distribuendo i valori dal maggiore al minore.

Entrando nel merito dell'articolata struttura della sezione in parallelo, per prima cosa si controlla se il numero di processori disponibili, acquisito con la funzione *omp_get_num_threads()*, è una potenza esatta di 2: questo perché le operazioni saranno svolte con un numero dimezzato di thread ad ogni ciclo. Quindi, se ci sono processori in sovrabbondanza, l'esubero non viene utilizzato. Il vettore *results* viene interamente copiato in un vettore "di transizione" *vect1* su cui sono eseguite le operazioni: esso viene diviso per il numero dei processori, con eventuale approssimazione per difetto, cosicché ogni thread elabora un segmento di uguale lunghezza del vettore dei risul-

tati. Se la divisione produce un resto di *vect1*, esso viene messo da parte e utilizzato in seguito, come sarà mostrato.

Il codice prosegue con il ciclo *for* cardine di questa sezione parallela, in cui il vettore è inizialmente diviso in n parti uguali, con n = numero massimo di threads selezionati. Attraverso le coordinate dei rispettivi estremi, ogni segmento è assegnato ad un thread ID, individuato dalla funzione *omp_get_thread_num()*; a questo punto si trova la già menzionata funzione *sort()* che viene eseguita esclusivamente al primo giro e che ordina gli elementi dei segmenti, sempre in parallelo e con evidente minore complessità rispetto all'ordinamento dell'intero vettore di partenza. In base alle esigenze dell'utente, è possibile scegliere di effettuare il sorting secondo la performance del gruppo 1, del gruppo 2 o secondo la performance totale. La direttiva *pragma omp barrier* dà inizio alla sezione seguente del merging: come accennato, a due a due le n parti vengono accoppiate, ottenendone quindi $n/2$ in cui è mantenuto l'ordinamento scelto. Il risultato del merging è inserito in un secondo vettore intermedio *vect2*, al quale al termine del giro del ciclo si aggiunge il resto di *vect1* che avanza dall'iniziale suddivisione. Il vecchio valore di *vect1* viene qui sostituito interamente con il nuovo *vect2*. Il ciclo *for* con queste istruzioni è ripetuto finché non restano che due soli segmenti da fondere in un unico vettore. Quando accade ciò, per l'ultima volta ad esso si aggiunge come appendice il resto iniziale e, terminato il ciclo, si procede con i conclusivi sorting e merging in serie di *vect1*. Finalmente il vettore *results* viene sostituito con il contenuto di quello appena ricavato e la funzione *reverse()* inverte l'ordine dei suoi elementi, mettendo come prime le coppie con punteggio più alto. L'utilizzo delle funzioni OpenMP per il sorting dei risultati costituisce probabilmente il beneficio principale apportato al codice, che ha raggiunto un netto guadagno in termini di costi computazionali: soltanto grazie alla parallelizzazione di queste operazioni, il tempo di esecuzione è stato ridotto del 70-80%, senza considerare il guadagno ottenuto con la funzione lambda, stimabile a una riduzione ulteriore del 10% circa.

3.1.5 Scrittura su file

Dall'esecuzione del programma, sono creati tre documenti di testo, uno dei quali in codifica binaria. Si tratta di:

- *Binarydata*, file che contiene sia *singlegene_ results* che *results*. In ogni riga sono scritti i due indici delle coppie, i due o più valori di singola performance e quello di performance totale. Attraverso un comando TCLAP al momento del lancio del software, è possibile specificare quale percentuale dei risultati salvare in questo documento (oltre al criterio di ordinamento degli stessi). Dal momento che l'interesse principale verte ad analizzare le coppie con miglior performance - come si vedrà nella prossima sezione del capitolo -, è conveniente precisare una percentuale massima di 20-30% perché la dimensione del file non sia inutilmente grande. Come ultimo dato, in coda al file viene inserito il numero di elementi per riga, per facilitare il reshaping in fase di acquisizione da parte di altri software. Al nome "BinaryData" sono aggiunte le sigle che indicano sia la percentuale salvata che il criterio di sorting scelto.

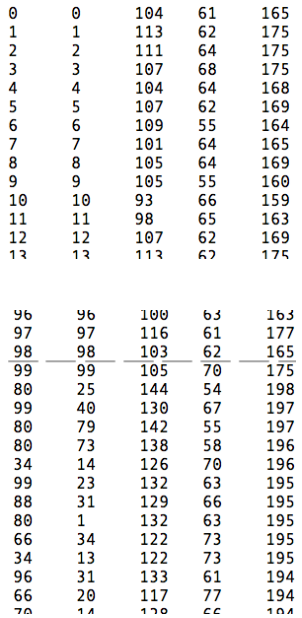

Figura 3.1: Esempio di file BinaryData. La linea tratteggiata evidenzia la fine dei valori delle coppie uguali e l'inizio dei valori delle coppie diverse

Eventualmente, è prevista anche la creazione del file FormattedData, con lo stesso contenuto del file precedente ma formattato, per aprirlo con i comuni editor di testo.

- *OutputData*, file formattato che contiene alcune informazioni su lettura ed elaborazione dei dati. Esse sono:

  - le etichette delle classi;
  - il numero di samples con suddivisione nelle rispettive classi;
  - il numero di geni totali presenti;
  - alcuni calcoli statistici come score medio, massimo e minimo delle performance;
  - il numero di processori utilizzati;
  - i tempi di sorting, sia quello seriale per le coppie uguali che quello parallelo;

- *GenesNames*, file che contiene i nomi dei geni, ovvero la prima colonna estratta dal database di espressione genica.

3.2 Network analysis (MATLAB)

Con il codice appena illustrato è stato possibile elaborare i dati di Gene Expression, ottenendo un file binario con le performance delle coppie di geni. La successiva analisi dei risultati è stata svolta non più utilizzando il linguaggio C++ bensì il software MATLAB, che permette una programmazione snella e gode di molte funzioni e di molti algoritmi già implementati all'interno dei suoi toolbox. Il suo impiego è stato possibile anche per via della dimensione ridotta dei calcoli da affrontare rispetto allo svolgimento del discriminante quadratico.

Scopo di questo studio è la ricerca delle coppie di geni con le più alte performance e la selezione di alcune di quelle formanti un network che conservi prestazioni di classificazione buone, se non addirittura migliori che le singole coppie isolate. L'insieme delle soluzioni qui esposte è da ritenersi un tentativo nuovo ed empirico con il quale approcciare il problema. In particolare, l'ipotesi-guida che i pochi geni costituenti un sotto-network rappresentino signature con performance migliori del network completo è alla base dei procedimenti seguiti: la stessa presenza di nodi fortemente connessi suggerisce che essi siano in qualche modo più significativi di altri.

3.2.1 Operazioni preliminari

Le prime istruzioni che il codice di MATLAB esegue sono l'acquisizione e la lettura del file *BinaryData* e del database contenente i valori di espres-

sione genica. Come già spiegato nella sezione precedente, l'ultimo valore in *BinaryData* rappresenta il numero elementi per riga, ovvero il numero di colonne, che la matrice dei risultati deve avere quando la si vuole visualizzare formattata: questo, quindi, è il dato utilizzato per effettuare l'operazione di *reshape* della matrice.

$$Perf = reshape(BinaryData, [Ncolonne, Nrighe])'; \tag{3.1}$$

Nel caso dei set analizzati in questa tesi, le classi delle etichette sono sempre due, e questo comporta che il numero di colonne sia pari a cinque: due indicano i geni, due indicano le performance di singola classe e l'ultima indica la performance totale. Dal file del database, invece, si estraggono le etichette *Lbl* (la prima riga), gli identificatori dei probe *ProbeId* (la prima colonna) e il corpo principale dei dati *Data*. Dalle label si calcola poi quante classi sono presenti (*LblU*) e quanti elementi ha ogni classe (*Nlbl*). Queste informazioni - numero di geni e di campioni, classi e loro elementi - vengono scritte sulla command window, insieme ai valori massimo e minimo per ogni colonna di *Perf*.

```
DATA:   54676 probes    118 samples
LABELS:
0:      103
1:      15
54676 couples
min:    14      5       90      0       98
max:    54675   54674   103     14      110
```

Figura 3.2: Esempio di output su Command Window

3.2.2 Calcolo delle componenti e classificazione

Il codice MATLAB prosegue con una funzione che include quasi tutte le operazioni successive, *qda_res_comp()*. Dopo aver prodotto su command window le statistiche di media, deviazione standard e massimo di *Perf*, la prima fondamentale azione qui eseguita è la determinazione della soglia su uno dei tre tipi di performance: queste è la "linea di taglio" in base alla quale per i calcoli successivi si selezionano solo le coppie con risultati superiori al valore di *threshold*. Si crea una matrice sparsa quadrata *Net* con dimensione pari alla lunghezza di *ProbeId*; nella matrice vengono individuate le coordinate dei geni scelti e si riempiono di 1 quelle posizioni. La cella *NetMW*, invece, contiene le tre matrici sparse popolate dalle performance 1, 2 e totale.

Come secondo passaggio fondamentale, si trova a questo punto la funzione *components()*, del pacchetto MatlabBGL, che calcola le componenti connesse del grafo rappresentato da *Net*. Il collegamento tra un gene e un altro avviene in base alla presenza di una performance al di sopra della soglia.

Di seguito, una seconda analisi del discriminante si effettua ora su ognuna delle componenti del network per stimare le nuove capacità di performance. La funzione già presente in MATLAB per fare ciò è *classify()*, appartenente alla *Statistics toolbox*, e per analogia al programma C++ essa viene applicata utilizzando la tecnica della *leave-one-out cross validation*. L'utente può scegliere con che tipo di metrica svolgere il calcolo; e come nell'algoritmo in C++, anche in questo caso si decide per *diagquadratic*, cioè viene valutata una matrice di covarianza quadratica diagonale per ogni classe. Terminata l'operazione, la command window presenta i risultati come mostrato di seguito.

```
#1 (7 probes)
        AFFX-DapX-3_at  209065_at       211047_x_at     215807_s_at     226267_at
        12      8621    10553   15242   26137   34069   48652
        70/103 (67.9612%)       11/15 (73.3333%)        81/118 (68.6441%)
#2 (2 probes)
        200603_at       215821_x_at
        193     15256
        46/103 (44.6602%)       10/15 (66.6667%)        56/118 (47.4576%)
#3 (8 probes)
        200728_at       204966_at       228766_at       229555_at       229770_at
        318     4555    28634   29423   29638   53149   53538   54313
        77/103 (74.7573%)       12/15 (80%)     89/118 (75.4237%)
#4 (11 probes)
        201042_at       202311_s_at     203695_s_at     203871_at       204041_at
        632     1901    3284    3460    3630    7088    13273   25973   31104   32899
        84/103 (81.5534%)       11/15 (73.3333%)        95/118 (80.5085%)
#5 (2 probes)
        202216_x_at     218968_s_at
        1806    18394
        58/103 (56.3107%)       12/15 (80%)     70/118 (59.322%)
#6 (23 probes)
        202791_s_at     203142_s_at     204241_at       205466_s_at     205841_at
        2381    2732    3830    5055    5430    6030    9743    12904   14380   17130
        87/103 (84.466%)        11/15 (73.3333%)        98/118 (83.0508%)
#7 (3 probes)
        202943_s_at     221388_at       1555041_a_at
        2534    20813   46836
        77/103 (74.7573%)       13/15 (86.6667%)        90/118 (76.2712%)
```

Figura 3.3: Stralcio dei risultati della classificazione delle componenti visualizzati su command window

In ogni riga, le informazioni riportate rappresentano:

- il numero progressivo indicante la componente;
- la quantità di nodi (geni) da cui è costituita ogni componente;
- i ProbeId di tali geni

- la percentuale di performance di classificazione per le singole classi e totale.

Dopo queste azioni, il codice resta in attesa di sapere quale sia la miglior componente che l'utente deciderà di scegliere. Una volta ottenuto il numero identificativo in input, l'esecuzione prosegue con la creazione del network *SignNet*, matrice sparsa di dimensione pari al numero dei geni della signature appena selezionata, il quale viene visualizzato attraverso il layout "a molle" *Kamada-Kawai* che minimizza l'energia interna del grafo. Su questo nuovo network viene ricalcolato il discriminante quadratico attraverso la tecnica statistica *k-fold cross validation* per ottenere una classificazione ancora più robusta che la precedente. I parametri da immettere ora in input sono il numero k di suddivisioni e il numero di iterazioni della validazione. Nuovamente, sulla command window vengono riportate le performance medie, mimime e massime per le classi singole e per la totale. È ricorrente che così facendo l'efficienza della signature aumenti quando viene classificata con la *k-fold cross validation*.

3.2.3 Operazioni sulla *best signature*

Il codice prevede un'ultima sezione fondamentale in cui si possono effettuare operazioni per provare ad aumentare la performance riducendo il numero di probe che compongono la signature selezionata. Sono elencate quattro possibilità:

- rimuovere i nodi con connettività uguale a 1;
- rimuovere tutti i nodi pendenti;
- rimuovere alcuni nodi specificandoli manualmente;
- non rimuovere alcun nodo e continuare con la signature precedente.

In seguito all'eventuale riduzione appena operata, si applica ancora la funzione *classify()* congiuntamente alla *k-fold cross validation* per ricavare una nuova classificazione e valutare se le performance sono migliorate.

Il codice offre l'opportunità di eseguire la Principal Component Analysis (PCA) con i geni appartenenti alla signature, scegliendo le due componenti con cui graficare i risultati, e di salvare in un file di testo i nomi dei ProbeId conseguiti.

3.2.4 Confronto con altri criteri di scelta della signature

Top scoring pair e Salient link

Nella prosecuzione del codice, è possibile verificare opzionalmente se ci siano coppie di geni che abbiano prestazioni superiori rispetto a quelle della signature scelta. Il semplice e intuitivo confronto viene fatto selezionando un numero a piacere di coppie di geni tra quelle che hanno ottenuto le performance maggiori in assoluto. Queste coppie, le *Top scoring pairs*, non formano quindi un network come nel caso della signature, e la previsione con cui si calcola la loro capacità di performance è che si raggiungano risultati peggiori proprio per via della mancanza di correlazione tra quei geni, seppur aventi i più alti score individuali. Come precedentemente fatto, si utilizza la *k-fold cross validation* per il calcolo del discriminante; si visualizzano su command window le performance e si mostrano l'andamento delle performance al variare del numero di probe selezionati.

Link salience

Anche con un secondo criterio si provano ad ottenere geni significativi che diano alte performance. Si tratta della *link salience*, una misura di network pesati e non topologica, applicabile a network scale-free. Questo approccio differisce dalle tradizionali misure, come la betweenness centrality, poiché non è affetto dalla posizione topologica dei link e non dà più importanza a quelli maggiormente vicini al baricentro del network. La link salience, invece, agisce come un filtro uniforme sui link in base al peso w_{ij} dei link della matrice $N \times N$ dei nodi. Matematicamente, viene espressa come

$$S = \langle T \rangle = \frac{1}{N} \sum_{k} T(k) \tag{3.2}$$

dove $T(r)$ è l'insieme dei shortest paths che minimizzano l'effettiva distanza $d_{ij} = 1/w_{ij}$. Secondo la definizione che ne danno D. Grady et al. [2011], per ogni link essa è "la frazione degli N shortest-path trees *T(r)* di cui il link prende parte". Perciò, se la salience $s_{ij} \approx 1$ allora il link (i, j) è essenziale per i nodi di riferimento, ma se $s_{ij} \approx 0$ significa che non gioca nessuno ruolo di importanza. In molti network scale-free si nota una sorprendente distribuzione bimodale della link salience e questo fa sì che avvenga con successo la classificazione dei link tra quelli con $s_{ij} \approx 1$ e quelli con $s_{ij} \approx 0$. L'insieme

dei link con salience prossima a 1 rappresenta quindi lo *scheletro* del network, come intuitivamente se ne dà definizione. Sempre D. Grady et al. riportano che, per la maggioranza dei network esaminati, solo una piccola frazione della totalità dei link, inferiore al 10% entra a far parte dello scheletro: con questi si forma il network su cui calcolare la classificazione tramite *k-fold cross validation.*

Capitolo 4

Applicazione a dataset reali

L'algoritmo realizzato in C++ è stato utilizzato per elaborare tre dataset di espressioni geniche, di differente natura e dimensione, analizzando successivamente in MATLAB le coppie di geni così ottenute, mediante l'approccio a network per la determinazione e il miglioramento dei sotto-network meglio classificanti. Gli sforzi di ricerca hanno rispettato una duplice tensione: l'individuare signature con le performance migliori e che esse fossero composte dal minor numero di geni possibile.

Di recente pubblicazione su *Leukemia* è un articolo ad opera di S. B. Amin *et al.* [2014] in cui si esamina l'efficienza di differenti tecniche utilizzate per predire la risposta migliore (*Complete Response*, CR) in dataset di pazienti affetti da mieloma multiplo, affermando sia che i profili di espressione portano intrinsecamente uno scarso contenuto informativo ai fini predittivi, sia l'inadeguatezza dei metodi considerati. In totale cinque dataset comprendenti 647 pazienti, tra nuove diagnosi e ricadute, vengono analizzati con diverse tecniche statistiche come Support Vector Machine, decision trees, K-nearest neighbors e Neural networks, validando infine il miglior modello con k-fold (o leave-one-out) cross validation e aplpicandolo ad un set di test. Nell'articolo, le performance ottenute con il procedimento appena descritto si dimostrano insoddisfacenti, rimanendo nel range 56-78%, con specificità e sensitività variabili e quasi mai oltre all'80%.

Con i risultati esposti in questo capitolo si vuole mostrare, al contrario, un modo efficace con cui è possibile individuare signature significative di coppie di geni con performance rilevanti; una di esse si è rivelata valida anche nel caso di applicazione a dataset provenienti da piattaforme di analisi

geniche diverse da quelle di origine. Questo test è stato fatto proprio sui dati analizzati da Amin *et al.*, fornendo così un confronto indiretto tra i metodi adottati e rivelando caratteri di *robustness* delle signature ottenute con il presente algoritmo.

4.1 Mieloma multiplo trattato con VTD

Presso il laboratorio di Biologia Molecolare dell'Istituto di Ematologia L.A.Seràgnoli dell'Università di Bologna è stato misurato il profilo di espressione di 54675 geni (probes) per 118 pazienti affetti da mieloma multiplo e preventivamente trattati con la terapia farmacologica di induzione Velcade-Talidomide-Desametasone (VTD). Tra questi:

- 15 raggiungono una CR;
- 14 raggiungono una non-complete response (nCR);
- 42 raggiungono una partial response (PR);
- 7 raggiungono una stable disease;
- 40 raggiungono una very good partial response (VGPR)

In questi dati originali, l'interesse va per il riconoscimento dei pazienti che, secondo la prognosi, rispondono pienamente alla terapia, quindi si cerca di ottenere una buona classificazione per i 15 campioni CR; i restanti vengono accorpati in un unico gruppo (103 campioni) sotto l'etichetta comune di PR. Al termine dell'esecuzione del programma in C++ per il calcolo delle performance tramite analisi del discriminante, nel file di output contenente le informazioni riassuntive si legge

```
Numero di pazienti 118, di cui:
103 nel tipo 0
15 nel tipo 1
Geni presenti: 54675
```

Questi dati provengono dalla corretta lettura delle label e del numero totale di righe del dataset. Tra le indicazioni dei tempi di calcolo, si trova invece:

```
Tempo impiegato per analizzare tutte le 1494705150 coppie: 2949.88 s
Tempo di sorting delle performance di 1494650475 coppie di geni differenti:
```

154.9 s

Il numero di coppie totali è dato da *(N probes)+(N probes)×(N-1)probes/2*, come spiegato nel capitolo precedente. Tra tutti, i tempi riportati sono i due più rilevanti: si conclude perciò che l'esecuzione è avvenuta nell'arco di circa 50 minuti.

Il file binario *VTD11_118_Q_1_LblCR_BinaryData_frac0.3_p0*, in cui sono stati salvati il 30% dei risultati delle performance, disposti in ordine decrescente per valore di performance totale, viene importato ed esaminato in MATLAB e hanno inizio le operazioni per la determinazione della signature migliore. Con una breve tabella sono mostrati minimo, deviazione standard e massimo delle tre classi.

classe	media	dev.std.	max
0 (103)	101.45	1.55	103
1 (15)	0.79	1.21	14
T (118)	102.24	1.01	110

Tabella 4.1

Su un massimo di 110, basandosi sull'istogramma *numero geni / valore performance* il valore di soglia è stato scelto pari a 109, corrispondente a 123 coppie selezionate con 172 probe. A questo punto, attraverso la funzione *components* del pacchetto MatlabBGL si calcola quante componenti connesse sono presenti nel network formato dai probe scelti, si ottengono 172 nodi e 49 componenti. Su ognuna di queste componenti si applica *classify* con metodo *diagquadratic* cross-validando con leave-one-out. Tra i risultati che compaiono su command window, una delle signature è formata da 12 probe e raggiunge buone performance.

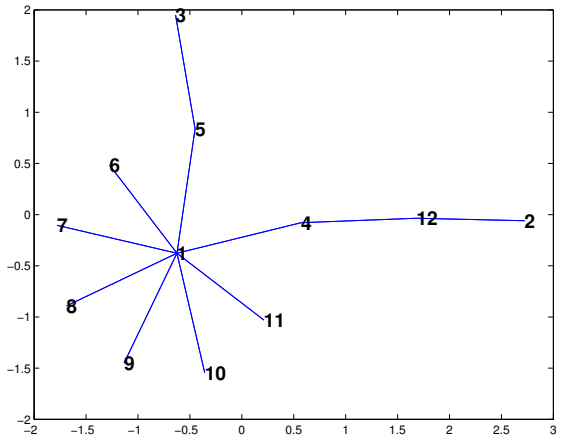

Figura 4.1: Sub-network con 12 nodi

Probe ID
207640_x_at
207640_x_at
212540_at
216762_at
234707_x_at
238238_at
243958_at
1554668_a_at
1560771_at
1561723_at
1561914_at
1569422_at

classe	score
0	93/103 (90.29%)
1	11/15 (73.33%)
T	104/118 (88.14%)

Tabella 4.2: 12 probe e performance della signature selezionata

Risultando la migliore, essa viene scelta per continuare nelle operazioni di "rifinitura". Una nuova classificazione ora è eseguita sulla signature mediante 5-fold cross validation con 200 iterazioni, riuscendo così ad osservare molteplici esiti per lo stesso set e calcolare lo score mediano, minimo e massimo.

classe	0	1	T
score mediano	93 (90.78%)	11 (73.33%)	104 (88.14%)
score min	89 (86.41%)	8 (53.33%)	100 (84.78%)
score max	97 (94.17%)	13 (86.67%)	108 (93.22%)

Tabella 4.3: Nuove performance della signature selezionata (12 probe)

Si nota come lo score mediano conferma quello raggiunto con la classificazione tramite *loocv* e come lo score massimo abbia un alto valore per tutte le tre classi, intorno al 90%.

La riduzione della signature tramite rimozione dei nodi con connettività $K = 1$ porta ad avere 4 probe che ottengono i seguenti score in seguito alla classificazione con *k-fold cross validation*:

classe	0	1	T
score mediano	87 (84.47%)	12 (80.00%)	99 (83.90%)
score min	81 (78.64%)	10 (66.67%)	93 (78.81%)
score max	92 (89.32%)	13 (86.67%)	104 (88.14%)

Tabella 4.4: Performance della signature ridotta (4 probe)

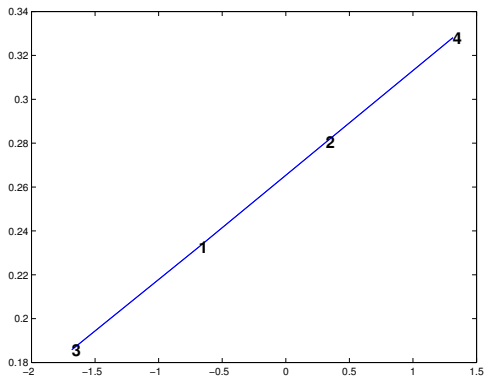

Figura 4.2: Signature ridotta (4 probe)

Con soli 4 probe si arrivano a raggiungere frazioni di veri positivi (*sensitivity*) quasi del 87% e di veri negativi (*specificity*) del 89%. I nomi dei probe con i nomi dei relativi geni sono i seguenti:

Probe ID	Gene symbol
207099_s_at	FAM129C
216762_at	IGLV1-44
234707_x_at	KANK1
1569422_at	CHM

Infine, la PCA è calcolata e graficata nelle sue prime due componenti.

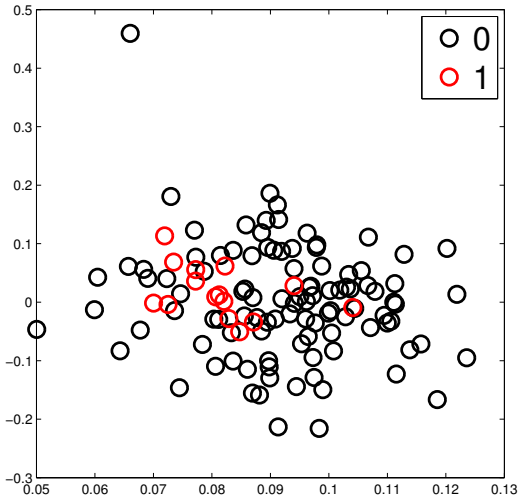

Figura 4.3: PCA con 4 probe

I numerosi elementi del gruppo 0 sono piuttosto diffusi e mascherano quelli del gruppo 1, che compaiono abbastanza sovrapposti agli altri campioni ma restano in maggioranza confinati in un'area ristretta.

4.2 Leucemia mieloide acuta e Leucemia linfoblastica acuta

Il secondo dataset è formato da 38 campioni e proviene dall'unione di 27 pazienti affetti da leucemia linfoblastica acuta (ALL) e 11 da leucemia mieloide acuta (AML). Esso comparve in letteratura quando Golub *et al.* [1999] selezionarono, tra i 7129 geni del profilo di espressione, signature semi-arbitrarie di 10–200 geni, tutte con *accuracy* del 100%. In particolare, 50 geni risultarono altamente correlati e adatti alla distinzione ALL-AML.

Sottoponendo il dataset al programma di classificazione mediante discriminante lineare implementato in questa tesi, si legge dal file di output:

Numero di pazienti 38, di cui:
27 nel tipo 0
11 nel tipo 1
Geni presenti: 7129

Tempo impiegato per analizzare tutte le 25414885 coppie: 31.0297 s
Tempo di sorting delle performance di 25407756 coppie di geni differenti: 44.3455 s

Vista la discreta dimensione dei dati, in poco meno di due minuti l'elaborazione ha termine e l'acquisizione in MATLAB dei risultati nel file binario riferisce che per la classe T si ottiene la performance media pari a 28 e la massima pari a 38.

classe	media	dev.std.	max
0 (27)	22.76	2.31	27
1 (11)	5.75	2.41	11
T (38)	28.51	1.80	38

Tabella 4.5

Situando la soglia direttamente a 38 vengono selezionati 400 geni, 395 dei quali costituiscono una delle tre componenti del network, che presenta performance pressoché ottime.

classe	score
0 (27)	26/27 (96.30%)
1 (11)	11/11 (100%)
T (38)	37/38 (97.37%)

Tabella 4.6: Performance della signature selezionata

La classificazione dell'intero dataset ricalcolata con questi geni non altera gli score mediani ma porta a raggiungere score massimi del 100% per tutte e due le classi.

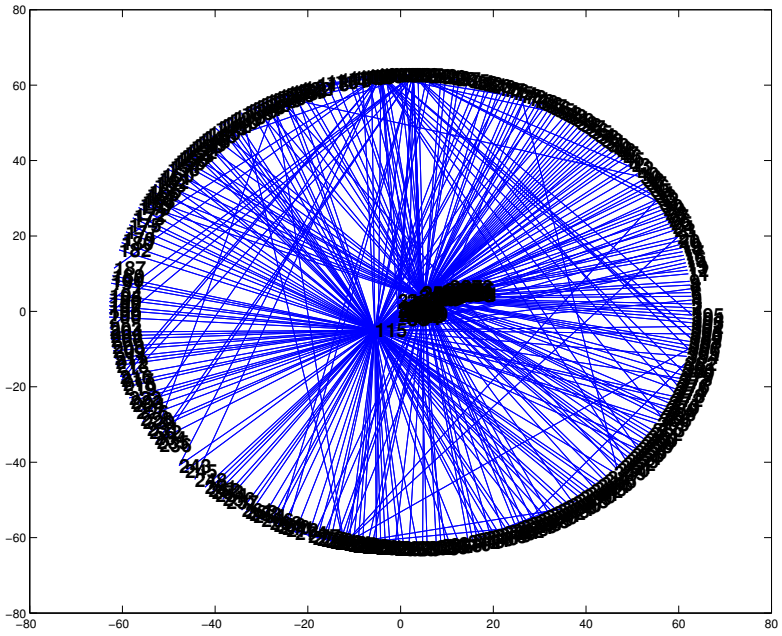

Figura 4.4: Sub-network con 395 nodi

classe	0	1	T
score mediano	26 (96.30%)	11 (100%)	37 (97.37%)
score min	26 (96.30%)	11 (100%)	37 (97.37%)
score max	27 (100%)	11 (100%)	38 (100%)

Tabella 4.7: Performance della signature (395 probe) dopo nuova classificazione con k-fold cross validation

Operando con successo un ridimensionamento della signature, eccessivamente ampia, tramite la rimozione dei nodi con connettività 1, si ottengono performance invariate ma con 68 probe.

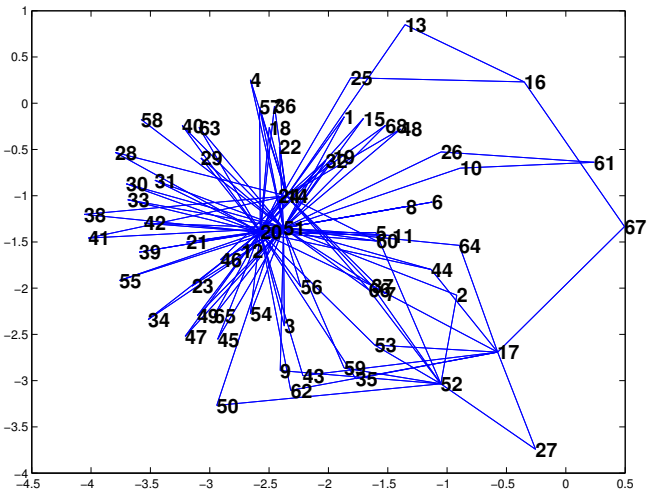

Figura 4.5: Signature ridotta (68 probe)

Come si può osservare, solo il nodo 58 ha ora connettività 1, quindi il successivo tentativo di riduzione viene condotto con inserimento manuale dei nodi che si desidera mantenere: immettendo i sette probe [20 51 24 14 5 52 50], la nuova signature non solo viene largamente ridotta ma accresce al 100% tutte le performance sulle due classi.

Con un ultimo passaggio si eliminano i nodi agli estremi (K=1) e la signature resta composta da soli 5 probe, sempre ottenendo massime percentuali di classificazione.

classe	0	1	T
score	27 (100%)	11 (100%)	38 (100%)

Tabella 4.8: Performance della signature ridotta (7 e 5 probe)

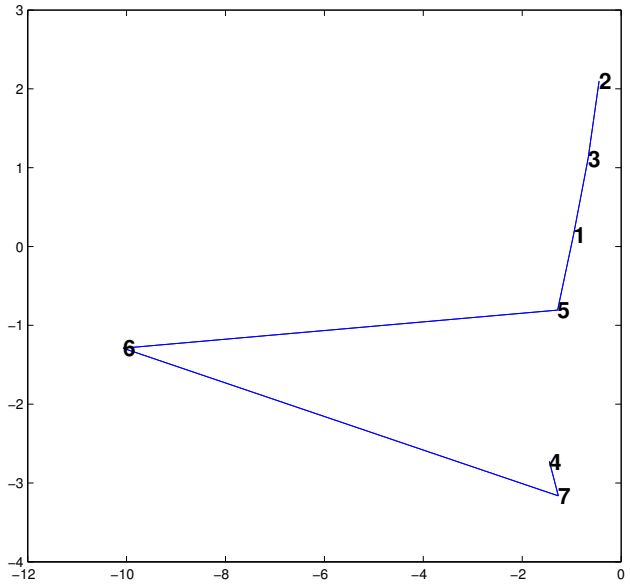

Figura 4.6: Signature ridotta (7 probe)

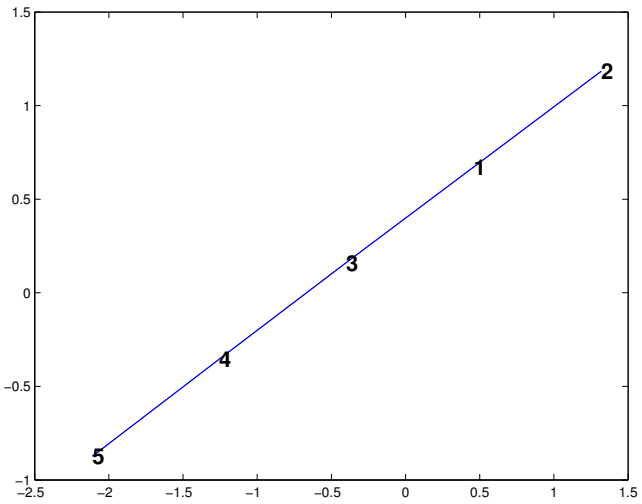

Figura 4.7: Signature ulteriormente ridotta (5 probe)

La PCA sulle prime due componenti rivela la buona distinzione che c'è tra i due gruppi.

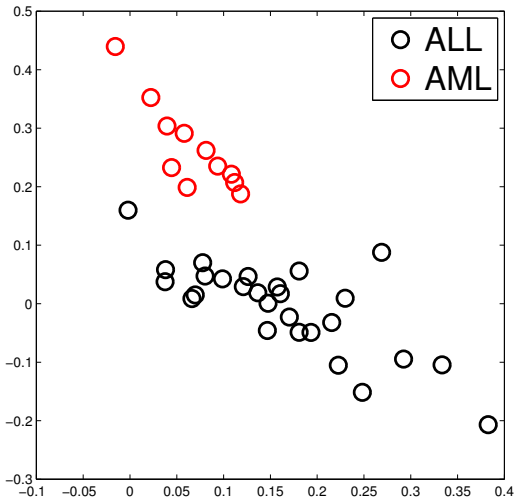

Figura 4.8: PCA con 5 probe

4.3 Tumore primario al seno con linfonodi negativi

Il terzo dataset preso in esame è il medesimo su cui Y. Wang *et al.* [2005] applicarono il modello di Cox di predizione statistica, dopo aver selezionato nel training set di 115 campioni una signature di 76 geni attraverso una suddivisione basata sullo stato dei ricettori estrogeni (60 ER+ e 16 ER-). I risultati del test set di 171 pazienti mostrano una sensitività del 93% (52/56) e una specificità del 48% (55/115). Nel presente studio, training e test set sono stati indagati uniti; tale dataset di 22283 geni per 286 campioni, quindi, è stato elaborato in circa 10 minuti. Numero di pazienti 286, di cui:

179 nel tipo 0
107 nel tipo 1
Geni presenti: 22283

Tempo impiegato per analizzare tutte le 248277186 coppie: 484.715 s
Tempo di sorting delle performance di 248254903 coppie di geni differenti: 155.498 s

Il file binario con gli score delle coppie presenta questi valori riassuntivi:

classe	media	dev.std.	max
0 (179)	131.66	20.75	179
1 (107)	39.62	19.23	95
T (286)	171.29	4.46	214

Tabella 4.9

Impostando la soglia a 199 vengono selezionati 151 coppie con 170 probe, che costituiscono i 170 nodi delle 25 componenti del network così creato. Il sotto-network designato per miglior performance è formato da 109 probe.

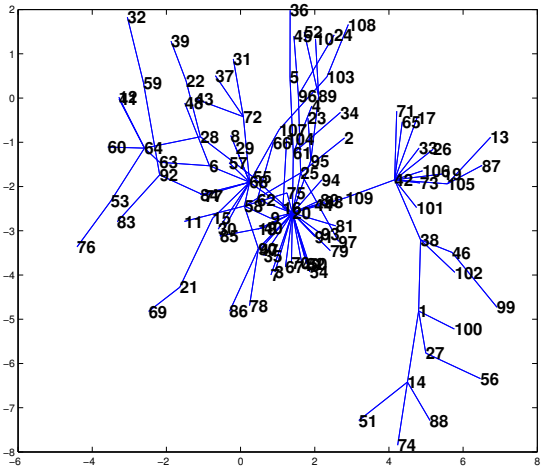

Figura 4.9: Sub-network con 109 nodi

classe	score
0 (179)	136/179 (75.98%)
1 (107)	75/107 (70.09%)
T (286)	211/286 (73.78%)

Tabella 4.10: Performance della signature selezionata (109 probe)

Con cross validation di 5-fold e 200 iterazioni, la signature mostra score massimi vicini o superiori al 75%.

classe	0	1	T
score mediano	135 (75.42%)	77 (71.96%)	212 (74.13%)
score min	127 (70.95%)	73 (68.22%)	202 (70.63%)
score max	141 (78.77%)	82 (76.64%)	220 (76.92%)

Tabella 4.11: Nuove performance della signature selezionata (109 probe)

Rimuovendo i nodi con connettività 1, la signature si riduce a 39 probe, con leggero incremento della performance per tutte le classi.

classe	0	1	T
score mediano	135 (75.42%)	77 (71.96%)	212 (74.13%)
score min	127 (70.95%)	73 (68.22%)	202 (70.63%)
score max	141 (78.77%)	82 (76.64%)	220 (76.92%)

Tabella 4.12: Performance della signature ridotta (39 probe)

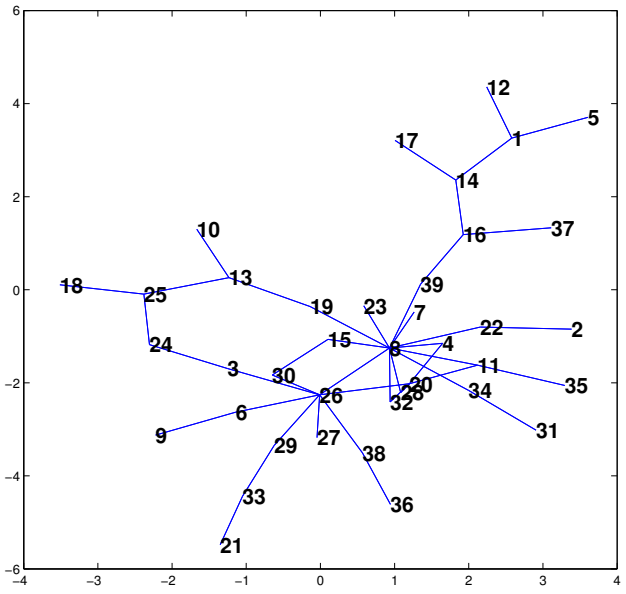

Figura 4.10: Signature ridotta (39 probe)

Procedendo con la riduzione del network non è stato osservato nessun miglioramento significativo degli score, anzi, piuttosto un lieve deteriora-

mento degli stessi. I risultati ottenuti per questa classe sono inferiori a quelli riportati in Wang *et al.* [2005], anche se la classe 0 viene individuata meglio. Bisogna però sottolineare le differenze iniziali tra le due configurazioni sperimentali, ovvero:

- un dataset unificato invece che una suddivisione in training e test set (Wang);
- la mancanza di stratificazione per informazioni biologiche invece che una suddivisione tra pazienti ER+ e ER- (Wang);

Con questi *distinguo*, quindi, il metodo seguito può ritenersi soddisfacentemente attendibile anche in questo caso.

La PCA, calcolata e graficata nelle sue prime due componenti, mostra quanto gli elementi delle due classi siano sovrapposti e perciò non ottimamente classificabili.

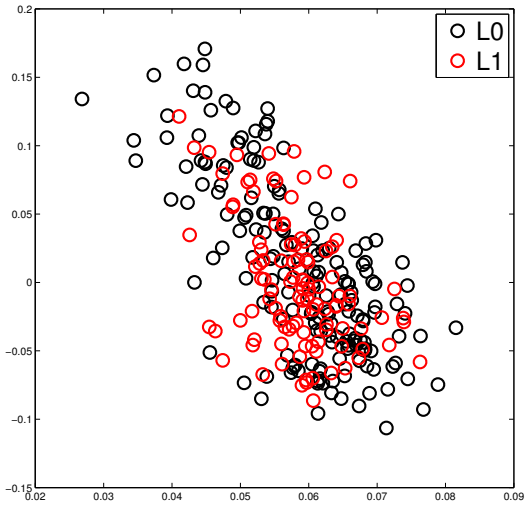

Figura 4.11: PCA con 39 probe

4.4 Classificazione con Top scoring pairs

Si svolge in questa sezione un confronto di performance tra ognuna delle tre signature ottenute precedentemente e la capacità di predizione che può avere un certo numero di coppie di geni tra quelle che hanno ottenuto un punteggio più elevato, senza che esse formino un network. Perciò, nella penultima parte del codice, si inserisce il numero di probe da selezionare all'interno delle coppie più performanti di ogni dataset e se ne osserva l'andamento in relazione allo score della signature corrispondente.

Mieloma multiplo trattato con VTD Per questo dataset sono introdotti i successivi gruppi di probe: [2 3 4 6 8 10 12 15 18 20 24 28 35 50]. Per ogni elemento del vettore vengono trovate le coppie contenenti i relativi probe e si produce una classificazione mediante *k-fold*.

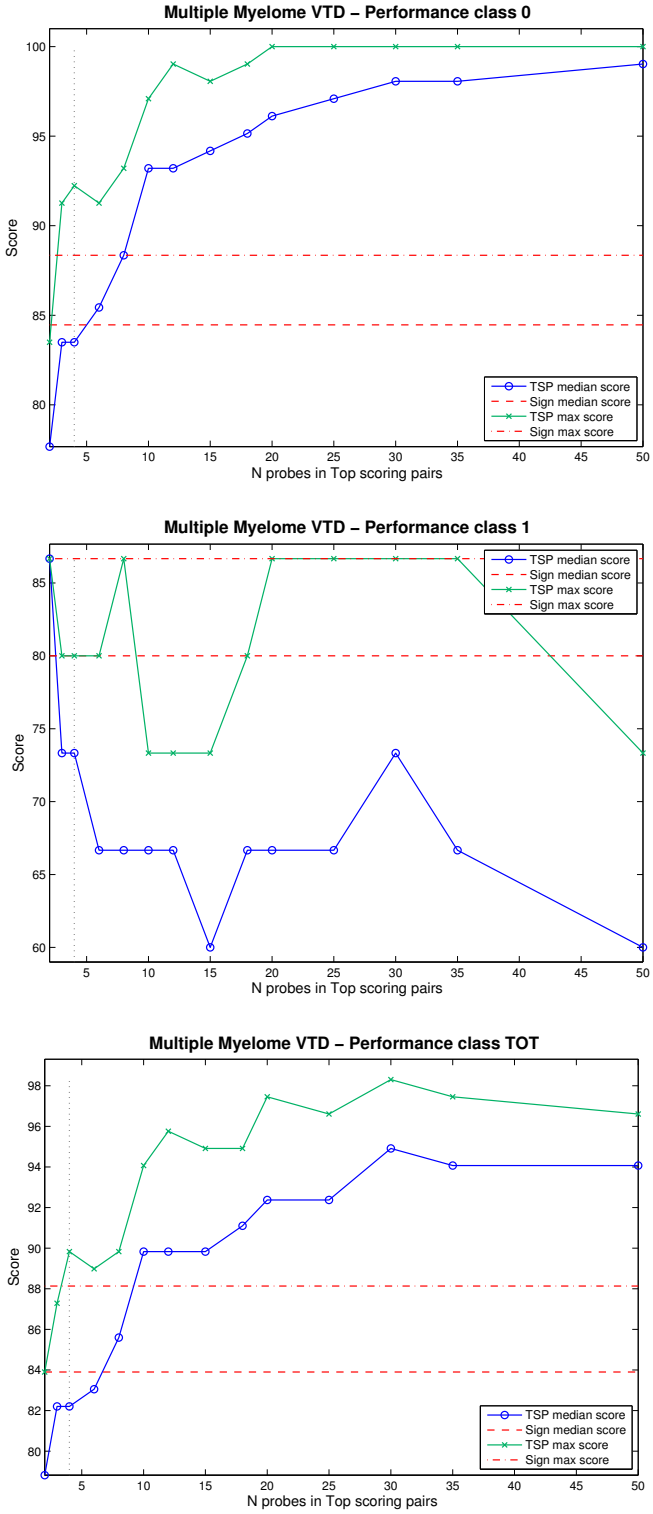

Figura 4.12: Confronto tra best signature e TSP comprendenti 5–50 probe

Si nota che la classificazione di pazienti del gruppo 0 (PR) migliora all'aumentare del numero di Top scoring pairs e già con n probe > 7 la performance sovrasta quella della signature; invece, il gruppo 1 (CR) è ottimamente classificato per mezzo della signature, il cui valore di score non viene mai superato da quello delle Top scoring pairs, il quale ha un andamento decrescente al contrario di come avviene per il gruppo 1. I risultati qui graficati si possono considerare una verifica soddisfacente della bontà dell'approccio a network adottato.

Leucemia mieloide acuta e Leucemia linfoblastica acuta Sul secondo dataset i numeri di probe selezionati sono: [2 3 4 5 6 7 8 10 12 15 20 25 30 50]. Il confronto tra le performance delle signature e delle Top scoring pairs sarebbe quasi superfluo poiché entrambi i metodi danno una risposta pressoché ottima, anche con pochissimi geni.

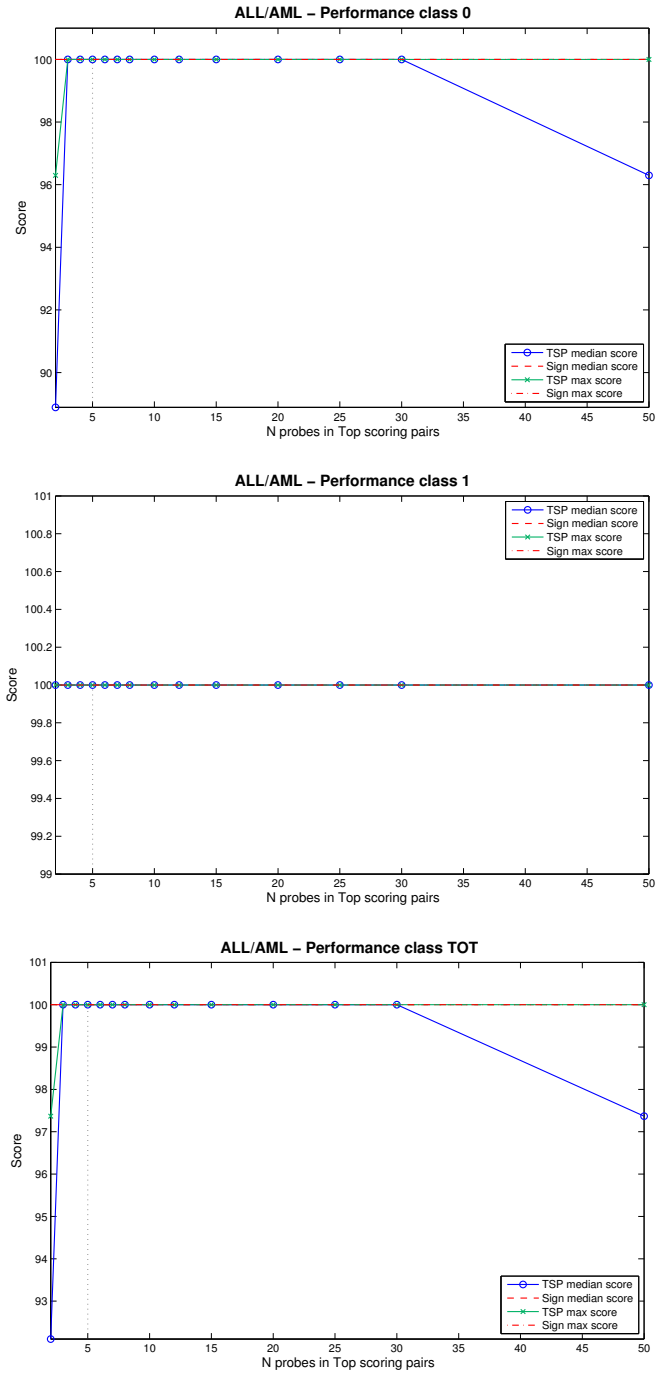

Figura 4.13: Confronto tra best signature e TSP comprendenti 5–50 probe

Tumore primario al seno con linfonodi negativi Per il terzo dataset si considerano le prime coppie contenenti le seguenti quantità di geni: [5 10

15 20 25 30 33 36 38 40 42 45 50 65 80]. Il confronto mostra che le Top scoring pairs restano ad una performance inferiore o uguale alla signature di 39 probe, salvo che avere un picco di crescita tra 50 e 80 probe per la classe 0 e tra 20 e 30 probe per la classe 1.

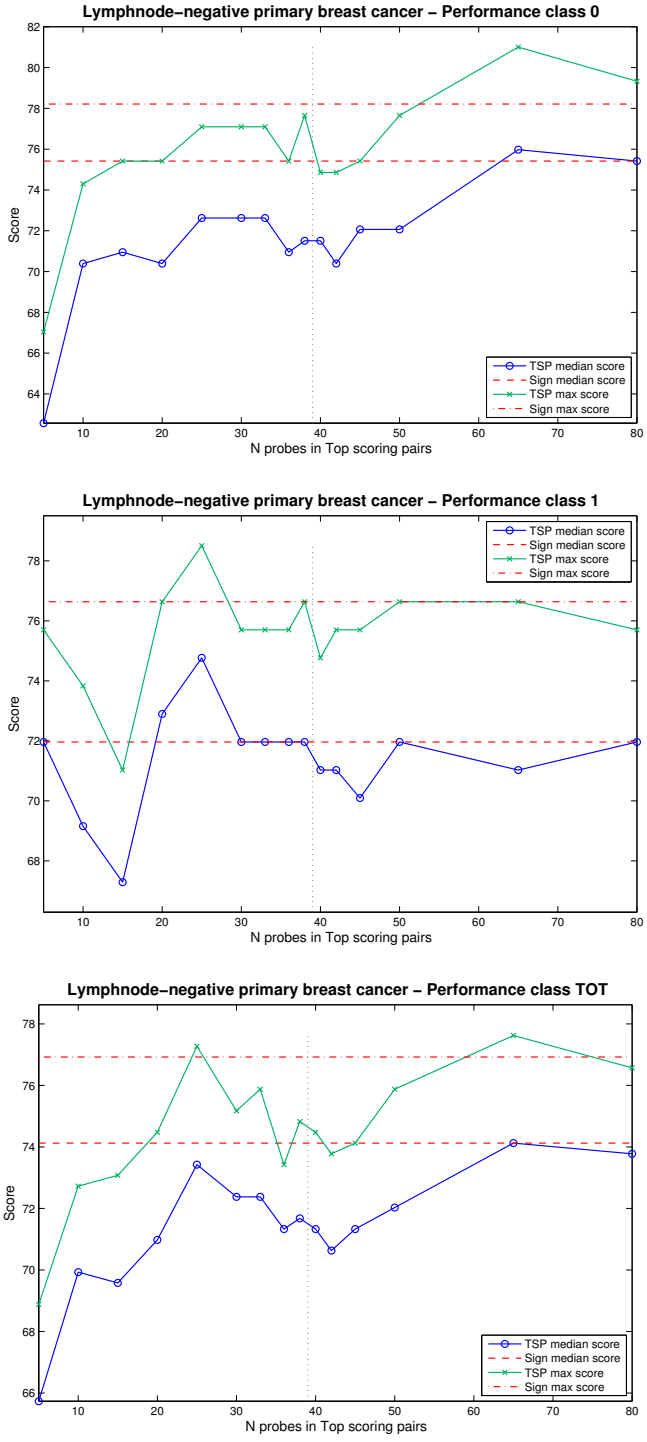

Figura 4.14: Confronto tra best signature e TSP comprendenti 5–80 probe

Questi ultimi andamenti confermano ancora una volta che non c'è superiorità di classificazione nelle prime coppie indipendenti rispetto alle signature individuate a partire da un network, quindi dal fatto che i loro geni siano in relazione.

4.5 Classificazione con Salient link

Un secondo controllo della consistenza degli esiti ottenuti si svolge servendosi della *link salience*. Essa viene calcolata a partire dal network formato dalle coppie al di sopra della soglia di performance stabilita ad inizio esecuzione; per ogni network la distribuzione della salience rispetta l'andamento bimodale che la teoria prevede, e la soglia di "taglio" del network è individuata come media pesata della salience tra 0.5 e 1. In tutti e tre i casi considerati vengono mantenuti quasi solamente i link che raggiungono il valore massimo di salience.

Mieloma multiplo trattato con VTD A partire dai 172 geni del network di partenza, i 23 selezionati per mezzo della soglia sulla link salience non comprendono quelli che fanno parte della signature. Essi presentano le seguenti performance:

classe	0	1	T
score mediano	90.29 (87.66%)	9.29 (61.93%)	99.58 (84.39%)
score min	85 (82.52%)	5 (33.33%)	93 (78.81%)
score max	96 (93.20%)	11 (73.33%)	106 (89.83%)

Tabella 4.13: Performance della signature selezionata per mezzo dei Salient link (23 probe)

Gli score della classe 0 si avvicinano a quelli della signature da 5 probe, ma per la classe 1 si hanno risultati decisamente peggiori, forse giustificabili alla luce della diversità completa di geni alla base delle signature.

Leucemia mieloide acuta e Leucemia linfoblastica acuta Del network originale di 400 geni, ne vengono selezionati 346, ulteriormente ridotti a 19 grazie alla rimozione dei nodi pendenti. Tra questi, sono contenuti 4

dei 5 geni della miglior signature identificata precedentemente. Come presumibile, le performance si attestano ancora una volta al 100% per questi dati.

classe	0	1	T
score mediano	27 (100%)	11 (100%)	38 (100%)
score min	26 (96.30%)	11 (100%)	37 (97.37%)
score max	27 (100%)	11 (100%)	38 (100%)

Tabella 4.14: Performance della signature selezionata per mezzo dei Salient link (19 probe)

Tumore primario al seno con linfonodi negativi Considerando il network iniziale con 170 geni, la soglia sulla link salience isola 106 nodi significativi, che si riducono a 35 rimuovendo il primo livello di pendenti. Si ottiene, quindi, una signature con 4 geni in meno rispetto a quella trovata in precedenza con l'analisi del discriminante, ma con performance essenzialmente invariate. Va notato che tutti i 35 geni di questo sotto-network sono presenti tra i 39 dell'altra signature.

classe	0	1	T
score mediano	135 (75.42%)	78 (72.90%)	213 (74.47%)
score min	128 (71.51%)	73 (68.22%)	205 (71.68%)
score max	141 (78.77%)	80 (77.57%)	222 (77.62%)

Tabella 4.15: Performance della signature selezionata per mezzo dei Salient link (35 probe)

Riassumendo i risultati ottenuti:

- Il dataset di dati originali su mieloma multiplo, analizzati per l'istituto di Ematologia "Seràgnoli" (Dipartimento di Medicina Specialistica, Diagnostica e Sperimentale (DIMES), Università di Bologna), presenta una signature di 4 soli geni che offre capacità di classificazione migliori del ben più vasto network di partenza. In special modo la classe 1, che rappresenta i pazienti con una *complete response* alla terapia VTD,

è accuratamente identificata, sebbene sia costituita da 15 campioni contro ai 103 della classe 0.

Applicando lo stesso metodo a dataset presentati in letteratura, si ottengono risultati alquanto soddisfacenti.

- La classificazione dei dati di Leucemia mieloide acuta e Leucemia linfoblastica acuta (ALL/AML) raggiunge percentuali massime con una signature di 5 geni, anziché di 50 come Golub *et al.* [1999] proposero.
- I dati provenienti dal tumore primario al seno con linfonodi negativi (Breast Cancer ln-) rivelano una signature di 39 geni con performance appena inferiori a quelle osservate da Wang *et al.* [2005] con una signature di 76; si attesta comunque come buon risultato soprattutto considerando che per il presente studio questo dataset non è stato suddiviso negli stessi training e test set dello studio di Wang, né sono state apportate stratificazioni per informazioni biologiche.

Statisticamente parlando, il procedimento impiegato in questo lavoro di tesi presenta caratteristiche di *robustness*, avendo potuto riscontrare buone performance di classificazione anche applicando la medesima signature individuata nei dati di mieloma multiplo a dataset provenienti da differenti piattaforme per analisi di genoma (Affymetrix, Agilent Technologies, ecc.).

		MM (VTD)	AML/ALL	BC ln-
	Signature	4probe	5probe	39probe
QDA-Net	Perf. media	84-80-84%	100-100-100%	75-72-74%
	Perf. max	89-87-88%	100-100-100%	79-77-77%
TSP	Perf. media	84-74-82%	100-100-100%	71-72-72%
	Perf. max	92-80-90%	100-100-100%	76-76-75%
	Signature	23probe	19probe	35probe
SL	Perf. media	88-62-84%	100-100-100%	75-73-75%
	Perf. max	93-73-90%	100-100-100%	79-78-78%

Tabella 4.16: Prospetto riassuntivo delle performance ottenute con metodi: Quadratic Disciminant Analysis con approccio a network, Top scoring pairs, Salient Link. Per le Top scoring pairs sono considerati gli stessi numeri di probe delle signature QDA-Net.

Capitolo 5

Conclusioni

Con il lavoro esposto in questa tesi ci si propone di sviluppare un algoritmo per la classificazione di dati ad alta dimensionalità, caratterizzato da un ottimo compromesso tra elevata performance e bassa dimensionalità della signature risultante. L'algoritmo si basa sul metodo dell'analisi del discriminante per classificare i campioni tramite coppie di variabili, e successivamente esegue uno studio topologico del network creato dalle coppie di variabili con le migliori performance. Sfruttando le proprietà di tale network si ottengono signature ottimali a bassa dimensionalità, ridotte almeno di un fattore 10^3 rispetto alle variabili di partenza. La robustezza di questi risultati è controllata in varie fasi dell'algoritmo mediante metodi di cross-validazione. L'approccio a network permette anche di operare sulla signature per un ulteriore rifinitura sia in termini del numero di variabili che di performance totale o su specifiche classi di campioni.

L'applicazione di questo metodo a dati reali, nel nostro caso dati di espressione genica relativi a patologie oncologiche (sia prodotti in collaborazione con l'Istituto di Ematologia del pollicinico S. Orsola di Bologna che provenienti da database pubblici), ha mostrato risultati ottimi in termini di performance di classificazione. Le signature di geni trovate sono costituite da un ridotto numero di geni e raggiungono valori di accuracy confrontabili con quelli riportati in letteratura. Inoltre, sono stati svolti confronti con altri metodi statistici comunemente utilizzati in letteratura (k-NN, SVM, ANN) i quali hanno fatto emergere la superiorità, o nei casi peggiori la comparabilità, dei risultati dell'algoritmo, a fronte però di una minore dimensionalità delle signature trovate e di una migliore interpre-

tabilità dei risultati grazie alla linearità delle boundaries di classificazione ottenute dal nostro metodo.

I risultati di questa tesi possono fornire un contributo allo sviluppo di applicazioni diagnostiche e prognostiche in ambito medico-clinico. Le signature ottenute con questo metodo possono essere facilmente interpretate in semplici termini di up e down regolazione dei geni individuati, permettendo così una semplice interpretazione biologica dei meccanismi che possono sottostare ai fenomeni osservati (come nel nostro caso per la resistenza a farmaci tumorali). La bassa dimensionalità delle signature, infatti, può permettere lo sviluppo di kit diagnostici scalabili per grandi numeri di pazienti mediante tecnologie a più elevata precisione rispetto agli array.

Bibliografia

[1] T. R. Golub et al., *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*, Science, 1999.

[2] Y. Wang et al., *Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer*, Lancet, 2005.

[3] W. Etienne et al., *Comparison of mRNA gene expression by RT-PCR and DNA microarray*, BioTecniques, 2004.

[4] . Dudoit et al., *Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data*, Journal of the American Statistical Association, 2002.

[5] . Shi et al.,*Top scoring pairs for feature selection in machine learning and applications to cancer outcome*, MC Bioinformatics, 2011.

[6] . Terragna, D. Remondini et al., *Correlation between eight-gene expression profiling and response to therapy of newly diagnosed multiple myeloma patients treated with thalidomide–dexamethasone incorporated into double autologous transplantation*, Annals of Hematology, Springer-Verlag Berlin Heidelberg, 2013.

[7] . B. Amin et al., *Gene expression profile alone is inadequate in predicting complete response in multiple myeloma*, Leukemia, 2014.

[8] . Geman et al., *Classifying Gene Expression Profiles from Pairwise mRNA Comparisons*, Statistical Application in Genetics and Molecular Biology, 2004.

[9] . Wu et al., *Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data*, Bioinformatics, 2003.

[10] astie, Tibshirani and Friedman, *The Elements of Statistical Learning*, seconda edizione, Springer-Verlag, 2009.

[11] . Grady et al., *Robust classification of salient links in complex networks*, Nature Communications, 2012.

[12] einze, von Löwis, Polze, *Feature Salience for Neural Networks: Comparing Algorithms* in Neural Information Processing, Springer, 2012.

[13] . Mozer et al., *Using Relevance to Reduce Network Size Automatically*, Connection Science, 1989.

[14] . M. Shekhtman et al., *Robustness of skeletons and salient features in networks*, Journal of Complex Networks, 2014.

[15] . Menichetti, D. Remondini et al., *Weighted Multiplex Networks*, Physics and Society, 2013.

[16] . Zhu et al., *Proteomics*, Annual Review of Biochemistry, 2003.

[17] . Tyers et al., *From genomics to proteomics*, Nature, 2003.

[18] . Dettmer et al., *Metabolomics–a new exciting field within the omics sciences*, Environ Health Perspect, 2004.

[19] .L. Wong et al., *Real-time PCR for mRNA quantitation*, BioTecniques, 2005.

[20] . Brazma et al., *Minimum information about a microarray experiment (MIAME) - Toward standards for microarray data*, Nature genetics, 2001.