Alma Mater Studiorum · Università di Bologna

Scuola di Scienze

Corso di Laurea Magistrale in Fisica Applicata

Implementazione e benchmarking dell'algoritmo QDANet PRO per l'analisi di Big Data genomici

Relatore:

Prof. Daniel Remondini

Presentata da:

Nico Curti

Correlatore:

Prof. Gastone Castellani

Sessione II

Anno Accademico 2015/2016

"*Ears cannot sleep hungry.*"

Salutary John

Abstract

Dato il recente avvento delle tecnologie Next-Generation Sequencing, che permettono di sequenziare interi genomi umani in tempi e costi accessibili, la capacità di estrarre informazioni dai dati ottenuti gioca un ruolo fondamentale per lo sviluppo della ricerca. Al giorno d'oggi i problemi computazionali connessi a tali analisi rientrano nel topic dei Big Data, con databases contenenti svariati tipi di dati sperimentali di dimensione sempre più ampia. Questo lavoro di tesi si occupa dell'implementazione e dell'ottimizzazione dell'algoritmo QDANet PRO, sviluppato dal gruppo di Biofisica dell'Università di Bologna: il metodo consente l'elaborazione di dati ad alta dimensionalità per l'estrazione di una Signature a bassa dimensionalità di features con un'elevata performance di classificazione, mediante una pipeline di analisi che comprende algoritmi di Dimensionality Reduction. Il metodo è generalizzabile anche all'analisi di dati non biologici, ma caratterizzati comunque da un elevato volume e complessità, fattori tipici dei Big Data in generale. L'algoritmo QDANet PRO, valutando la performance di tutte le possibili coppie di features, ne stima il potere discriminante utilizzando un Naive Bayes Quadratic Classifier per poi determinarne il ranking. Una volta selezionata una soglia di performance, viene costruito un network delle features, da cui vengono determinate le componenti connesse. Ogni sottografo viene analizzato separatamente e ridotto mediante metodi basati sulla teoria dei networks fino all'estrapolazione della Signature finale. Il metodo, già precedentemente testato su alcuni datasets disponibili al gruppo di ricerca con riscontri positivi, è stato messo a confronto con i risultati ottenuti su databases omici disponibili in letteratura, i quali costituiscono un riferimento nel settore, e con algoritmi già esistenti che svolgono simili compiti. Per la riduzione dei tempi computazionali l'algoritmo è stato implementato in linguaggio C++ su HPC, con la parallelizzazione mediante librerie OpenMP delle parti più critiche.

Indice

Introduzione

Sulla scia del *Progetto Genoma Umano* (HGP, 2003), l'interesse per i dati di natura biologica e lo sviluppo tecnologico per la loro estrazione sta subendo una crescita esponenziale. A partire dai dati di sequencing, la diffusione delle tecnologie *High-Throughput*, i.e. ad alto tasso di produzione di dati, ha determinato un netto aumento dei risultati sperimentali. Tempi ridotti di acquisizione e bassi costi sono fattori che ci hanno permesso l'ingresso in una nuova era della ricerca scientifica, l'era dei *Big Data*.
Il progetto TCGA (*The Cancer Genome Atlas*) costituisce un emblema di questo settore di ricerca. All'interno del database sono presenti un gran numero di campioni di espressione genica, proteomica, trascrittomica ed epigenetica di pazienti affetti da diverse $(1-6)$ tipologie di tumore. I dati di profili molecolari possono essere informativi e significativi per molteplici aspetti della pratica oncologica. Uno degli obiettivi primari per il trattamento di questo genere di patologie è una prognosi accurata per i pazienti, al fine di poterli suddividere in gruppi in relazione alla gravità. Tradizionalmente la prognosi è redatta mediante variabili cliniche quali l'età e lo stadio del tumore. La ricerca è, invece, sempre più indirizzata verso una possibile incorporazione delle informazioni molecolari.
Alla luce di questi presupposti, il presente lavoro si pone come obiettivo quello di fornire un nuovo metodo per l'analisi dei Big Data, con particolare applicazione ai dati di natura biologica. Il metodo QDANet PRO (*Quadratic Discriminant Analysis with Network PROcessing*) è stato ideato dal gruppo di ricerca in biofisica del Dipartimento di Fisica ed Astronomia dell'Università di Bologna. Dall'analisi di campioni suddivisi in due classi di appartenenza, l'algoritmo consente l'estrazione di un sotto-campione ad elevata efficienza di classificazione correlata ad una bassa dimensionalità. In questo modo vengono uniti gli obiettivi di estrazione delle informazioni maggiormente significative con quelli relativi alla classificazione.
Come test per il metodo sono stati utilizzati i dati disponibili nel TCGA, focalizzando lo studio su quattro tipologie di tumore: *kidney renal clear cell carcinoma* (KIRC), *glioblastoma multiforme* (GBM), *ovarian serous cystadenocarcinoa* (OV) e *lung squamous cell carcinoma* (LUSC). Per ognuno sono stati analizzati differenti tipologie di dati: *miRNA expression*, *mRNA expression*, *methylation*, *reverse phase protein array* e *copy-number variation*. Questi dati sono depositati sul cloud di Amazon e disponibili online mediante registrazione gratuita.
Un'intensa analisi del medesimo dataset è stata eseguita da Yuan et al. [1], in cui

sono state integrate informazioni cliniche con quelle molecolari e testate le efficienze di numerosi classificatori. Per questo si è scelto di considerare il loro studio come termine di paragone per i risultati del QDANet PRO.
Nei successivi capitoli verrà presentato il settore di studio dei Big Data attraverso le potenzialità che portano con loro e le problematiche per l'estrazione dei risultati, con riferimento alle infrastrutture di analisi necessarie per la loro manipolazione (Capitolo 1). Nel Capitolo 2 sono trattate le metodologie di analisi, con riferimento ai metodi di Machine Learning ed una breve presentazione delle tecniche legate alla modellizzazione a complex networks: per ognuno è riportata una introduzione teorica seguita da un'implementazione del metodo in linguaggio Python e C++. Nella parte 3 è spiegato nel dettaglio l'algoritmo QDANet PRO, il metodo utilizzato per l'analisi in questo lavoro di tesi e che coinvolge ognuno degli argomenti precedentemente esposti. Infine sono presentati i risultati ottenuti dalla sua applicazione a databases reali (Capitolo 4) e le conclusioni in merito (Capitolo 5). Nelle appendici sono riportati alcuni dei codici sviluppati in questo lavoro di tesi.

Capitolo 1

Big Data

Con il termine *Biological Big Data* ci si riferisce a grandi e complessi datasets, tipicamente impossibili da memorizzare, manipolare ed analizzare tramite un computer standard [13]. Come ordini di grandezza, si parla di circa 140 Gb per il sequencing di un singolo individuo [5] e si consideri che l'*Array Express*, un compendio di dati pubblici di gene expression, contiene più di 1.3 milioni di genomi raccolti in più di 45000 esperimenti [3].

La crescita dei dati a disposizione ha portato la richiesta di progettazione e realizzazione di numerose banche dati capaci di organizzare, classificare ed estrarre informazioni e risultati. L'Istituto Europeo di Bioinformatica (EBI) a Hinxton (UK), uno dei repository di dati biologici più grandi del mondo, attualmente contiene 20 petabytes di dati e back-ups genici, proteici e di piccole molecole. I dati genomici costituiscono il 10%, quantità che raddoppia ogni anno: tali quantità rappresentano circa un decimo dei dati provenienti dal CERN [5]. Considerando la crescita annuale di generazione dei dati, è previsto il raggiungimento dei 44 zettabytes entro il 2020, 10 volte il valore raggiunto nel 2013 [11]. D'altra parte, la capacità di elaborazione e le tecniche computazionali d'analisi stentano a tenere questo ritmo elevato di crescita. Stiamo assistendo quindi ad un allargamento del gap tra il gran numero di dati a disposizione e la nostra capacità di utilizzarli.

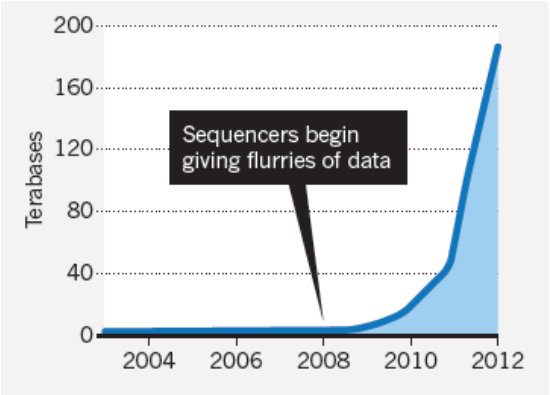


Figura 1.1: Crescita del volume di dati immagazzinati nell'European Bioinformatics Institute in funzione del tempo [5].

Con la presenza di questo enorme quantitativo di informazioni ancora da elaborare ed integrare, gli esperimenti biologici *in silico* sono la nuova frontiera di ricerca. A discapito di quanto possa accadere in altre branche di ricerca, l'applicazione di differenti metodi di analisi ai databases attualmente in rete ci permette di carpire molte informazioni

senza dover ripetere acquisizioni o procedere ad esperimenti. I Big Data in biologia, quindi, aggiungono possibilità agli scienziati in quanto i risultati che cerchiamo sono spesso già presenti nei databases a disposizione [5].

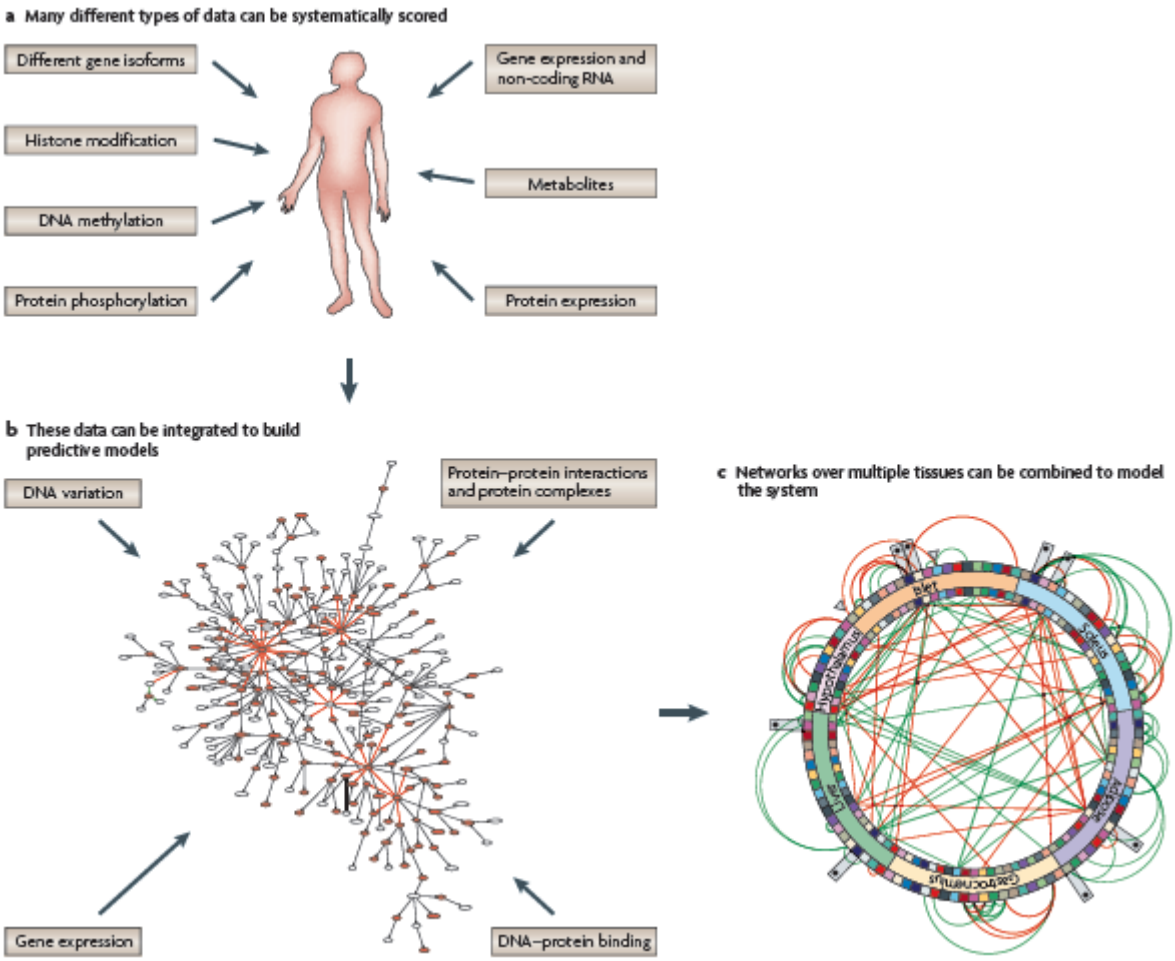


Figura 1.2: **Generazione ed integrazione di diverse tipologie di dati**. La modellizzazione dei sistemi viventi richiede la generazione **(a)** e l'integrazione **(b)** di dati multidimensionali. In **(b)** sono mostrati complessi datasets come networks, in cui i nodi rappresentano le variabili biologiche di interesse, come variazioni del DNA, variazioni di RNA, livelli delle proteine, stati delle proteine, livelli di metaboliti e caratteristiche associate alla malattia, e gli edges tra questi nodi rappresentano relazioni causali tra le variabili. Questi networks più granulari (a livello dei geni) possono essere efficacemente riassunti in subnetworks **(c)** che interagiscono tra loro. In questo modo, visualizzando solamente i core dei networks si ottiene una rappresentazione di come i processi biologici interagiscano tra loro per definire stati fisiologici associati alle patologie. [2]

Negli ultimi anni molte agenzie governative americane, come il National Institutes of Health (NIH) e il National Science Foundation (NSF), hanno accertato l'utilità dei Big Data per il processo decisionale e la loro profonda influenza per i futuri sviluppi di ricerca [7]. Una caratteristica specifica della post-genomica è, infatti, la correlazione

tra le informazioni immutabili risiedenti nel corredo genetico (genotipo), con quelle manifestate dall'individuo (fenotipo).
A livello computazionale, fisica, statistica e bioinformatica si stanno occupando della ricerca di metodi per l'analisi di voluminosi datasets. Per questo, gli algoritmi di Machine Learning per l'analisi di grandi quantità di dati non manipolabili con un approccio statistico diretto [3], necessitano di ottimizzazioni e modifiche per aumentarne l'efficienza di calcolo. Per l'ottimizzazione dei tempi di calcolo è necessario estendere gli algoritmi esistenti ed elaborare tecniche di Dimensionality Reduction. Un approccio standard, infatti, porta all'analisi di problemi catalogabili come NP Complessi. Negli ultimi anni si stanno diffondendo metodi legati al calcolo parallelo e di computazione distribuita, cloud computing e distributed machine learning. Questo ha segnato l'ingresso dei Big Data nella *e-Science*[1], di cui fanno parte da tempo la fisica delle particelle e le scienze della terra [7].
É importante operare anche sui metodi di rappresentazione grafica: i metodi *Linear Mapping* (PCA, Discriminant Analysis etc.) sono i più utilizzati a tale scopo. Ad essi vengono affiancati metodi basati sulla Cluster Analysis, tra cui CLARA (Clustering LARge Applications) e BIRCH (Balanced Iterative Reducing using Clustering Hierarchies) [11], e su Complex Networks [9] per ottenere una Dimensionality Reduction. Inoltre, combinando vari classificatori il metodo acquista più potenza, robustezza, resistenza, accuratezza e generalità: tale innovazione colma il deficit dato dall'utilizzo di singole funzioni discriminanti che producono buoni risultati solo in casi specifici [12].
Un'ulteriore problematica legata ai Big Data risiede nell'eterogeneità dei dati biologici, diversamente da quanto accade nel settore della fisica delle particelle o dell'astrofisica. Mentre il CERN possiede un solo *supercollider* che fornisce dati in un solo luogo, le ricerche biologiche che generano grandi volumi di dati sono distribuite tra più laboratori sconnessi tra loro, comportando lo storage di tipologie di dati differenti sia in formato che in acquisizione [5]. Questo grava sulle analisi computazionali che spesso utilizzano databases eterogenei ed indipendenti per analisi di inferenza e validazione.
Attualmente ci sono diversi progetti per lo sviluppo di strumenti per gestire, integrare e analizzare dati biomolecolari, basati sulla cooperazione tra diverse università e centri specialistici, per coniugare competenze specialistiche nei diversi domini delle *Life Sciences* con analoghe competenze avanzate nell'utilizzo degli strumenti della *Information Technology*.

1.1 Infrastrutture di Analisi

Per risolvere le necessità di analisi computazionali, la tecnologia *cloud computing* è sempre più utilizzata [10] ed è particolarmente efficiente sia per lo storage dei dati che per la loro elaborazione. Con il termine cloud computing ci si riferisce alla fornitura

[1] Scienze ad alto grado di livello di computazione che vengono implementate su sistemi di calcolo distribuito.

di risorse informatiche tramite internet. In particolare il sistema cloud permette ad individui ed aziende di utilizzare software ed hardware che sono gestiti da terze parti in locazioni remote. Il modello consente l'accesso ad informazioni e risorse da ogni zona in cui sia disponibile un'accesso alla rete, fornendo un pool condiviso di risorse, inclusi spazio di archiviazione dati, reti, potenze di calcolo elevate per l'elaborazione ed applicazioni aziendali.
L' *U.S. National Institute of Standards and Technology* (NIST) ha fornito questa definizione di cloud computing [26]:

Il cloud computing è un modello per un conveniente accesso di rete on-demand ad un pool condiviso di risorse di calcolo configurabili (es. reti, server, storage, applicazioni e servizi) che possono essere rapidamente forniti e rilasciati con il minimo sforzo di gestione o interazione con il fornitore del servizio.

ll Beijing Genomics Institute (BGI), uno dei centri di sequenziamento genico più famoso al mondo, è un noto esempio di applicazione del modello: utilizzando cloud computing l'analisi dei dati genici viene eseguita attraverso calcolo parallelo su centinaia di migliaia di core [2]. L'ottimizzazione computazionale per svariate applicazioni è evidente: usando un computer standard, viene impiegato un tempo di circa 30 s per confrontare i dati di un solo gene all'interno di un database e di 4 giorni se il numero di geni sale a 6000. Distribuendo la ricerca su 1000 CPUs, il lavoro può essere compiuto in meno di 10 minuti [2]. Da notare che questo tipo di problema rappresenta un tipico calcolo di moderata complessità.

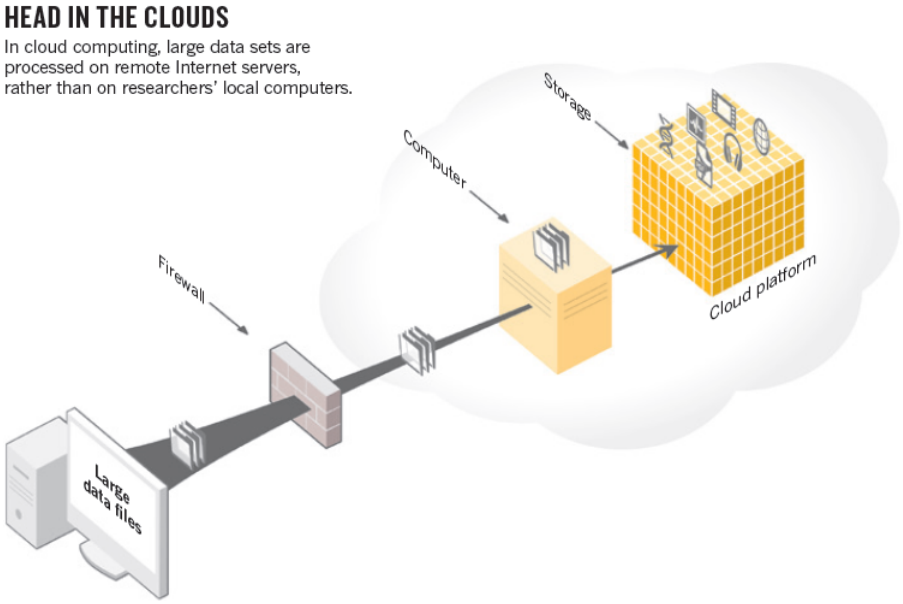


Figura 1.3: Schema di trasferimento dei dati secondo la tecnologia cloud computing [5].

Se a livello computazionale è evidente l'efficienza di queste strutture, la proliferazione delle infrastrutture cloud può portare a risultati controproducenti qualora i dati vengano

distribuiti su più piattaforme distinte e ne risulti comunque necessario lo spostamento e scambio. Inoltre, occorre cambiare radicalmente la mentalità di analisi, scoraggiando lo storage dei dati su hardware locali ed incentivando l'utilizzo del cloud sia per l'analisi che per lo stoccaggio dei databases.

Nonostante la possibilità di utilizzare computer cloud per il calcolo a distanza, la scrittura degli algoritmi deve riuscire a supportare alti livelli di parallelizzazione per la distribuzione su più macchine, il cosiddetto *High Performance Computing* (HPC). Le tecnologie HPC fondano la loro efficienza proprio sulla parallelizzazione su computer cluster. Attualmente, oltre all'ormai consolidato metodo di calcolo parallelo su CPUs stanno prendendo piede anche metodi innovativi fondati sulla cooperazione di componenti hardware differenti. Al riguardo citiamo il metodo *MapReduce* e l'*Heterogeneous Computational Environments* (GPU Computing).

1.1.1 MapReduce

Diversi anni fa emerse un paradigma di calcolo distribuito detto MapReduce per semplificare lo sviluppo di applicazioni di calcolo massivamente parallelizzate. Il metodo MapReduce consente, infatti, lo splitting di un problema in più sotto-problemi omogenei in un *map step*, seguito da un *reduce step* che combini l'output delle piccole elaborazioni nell'intero output cercato.

Un esempio di problema affrontabile con l'utilizzo del MapReduce è l'allineamento delle sequenze raw di *reads* di un genoma, in relazione ad un genoma di riferimento. I sotto-problemi omogenei in questo caso consistono nell'allineare le singole reads al genoma di riferimento. Una volta allineate tutte (map step) si può passare alla loro unione in un unico filamento (reduce step).

La crescita in efficienza legata alla combinazione tra il calcolo distribuito MapReduce e le infrastrutture cloud computing dovrebbe ormai essere evidente. Riprendendo la definizione del paradigma è chiaro, inoltre, come il metodo di splitting possa essere ri-applicato anche ai singoli sotto-problemi, andando a realizzare una griglia ancor più fine di parallelizzazione in relazione alla potenza di calcolo a disposizione. Molti lavori recenti hanno dimostrato come questa sinergia possa costituire un'efficiente metodologia di indagine per l'abbattimento dei tempi di calcolo su ingenti volumi di dati come i Biological Big Data.

Una volta dimostrata l'efficienza del metodo, l'implementazione di un codice che sfrutti questa metodologia di parallelizzazione viene agevolata dall'utilizzo di apposite librerie scaricabili dalla rete. Il più noto esempio è dato dalla *MR-MPI Library*, un'implementazione open-source del metodo MapReduce per il calcolo a memoria distribuita. Realizzata da Steve Plimpton e Karen Devine presso i laboratori Sandia National, la libreria include interfacce C e C++ richiamabili dalla maggior parte dei linguaggio ad alto livello (es. per il linguaggio Python è incluso lo scripting OINK per l'implementazione).

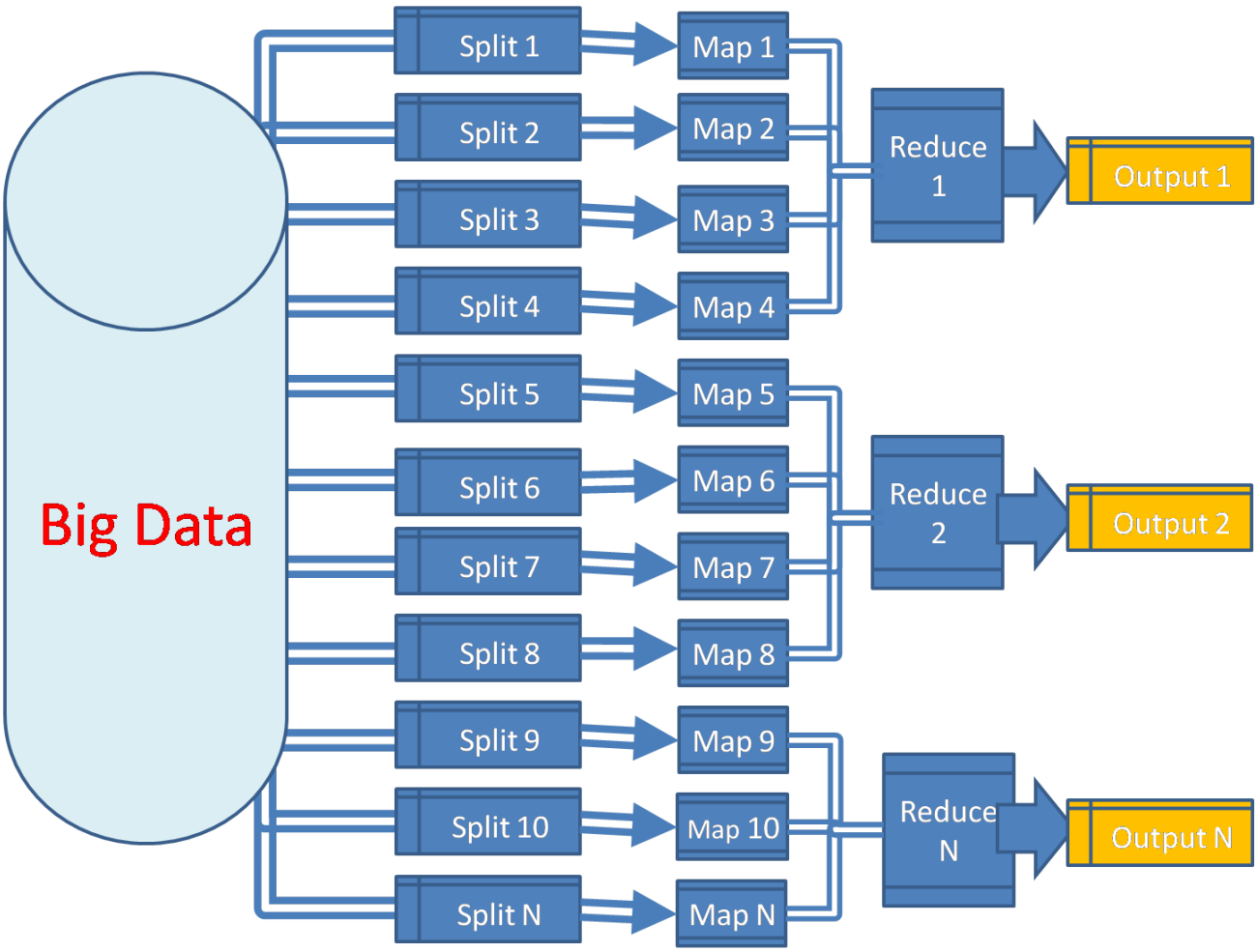


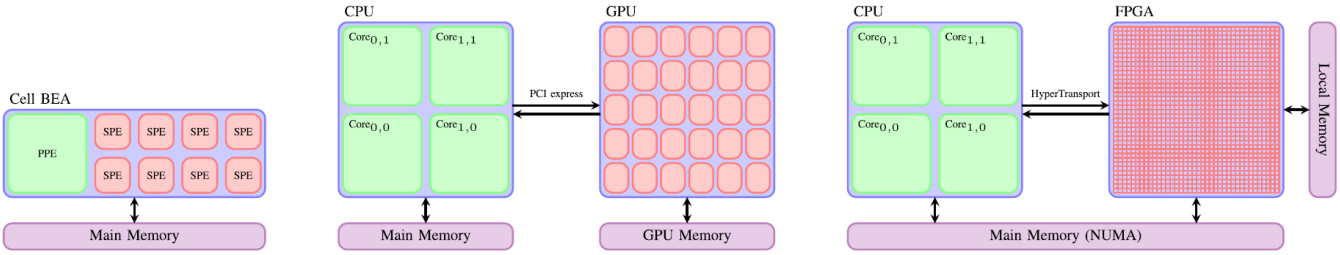
Figura 1.4: Schema di funzionamento del paradigma MapReduce per la distribuzione del calcolo su più CPUs. In un primo step l'analisi del Big Data viene suddivisa in sotto-problemi computazionalmente semplici (*map step*) per poi al termine riunire i risultati in un ridotto numero di output (*reduce step*). Questo tipo di parallelizzazione è già reso disponibile in diverse librerie di uso comune per l'elaborazione di Big Data: un noto esempio è *Apache Hadoop*.

1.1.2 Heterogeneous Computational Environment

Un altro sistema complementare alle infrastrutture basate sul cloud è l'utilizzo di computer a multi-core (CPUs) eterogenei, ovvero integrati con acceleratori specializzati nell'aumentare la potenza di calcolo e la parallelizzazione. Attualmente questi acceleratori sono dati dalle GPU.

Seguendo lo sviluppo dell'industria dei videogiochi, le *unità di elaborazione grafica* (GPUs) sono passate dai mainframe ad essere workstation per PC, arrivando a rappresentare una valida alternativa alle piattaforme di calcolo tradizionali. Le GPUs offrono prestazioni estremamente elevate in calcoli in virgola mobile ed una possibilità di parallelizzazione elevata ad un costo molto basso, promuovendo un nuovo concetto

di mercato per l'HPC. In particolare si parla di *heterogeneous computing* quando processori con diverse caratteristiche lavorano insieme per il raggiungimento delle migliori performances per l'applicazione richiesta.



(a) CBEA (b) CPU in combination with GPU (c) CPU in combination with FPGA

Figura 1.5: Alcune schematiche architetture per l'*heterogeneous computing*: la *Cell Broadband Engine* è un chip eterogeneo (a), una CPU in combinazione con una GPU (b) e una CPU in combinazione con una FPGA (c). La GPU è connessa con la CPU attraverso un PCI e alcune FPGA sono collegabili con i processori AMD e Intel.

Le più famose compagnie di microprocessori come NVIDIA, ATI/AMD o Intel, hanno sviluppato prodotti hardware destinati proprio al mercato dell'heterogeneous computing. In aggiunta sono fornite componenti software per poter accedere facilmente alla potenza di calcolo di queste macchine. CUDA (*Compute Unified Device Architecture*) è la soluzione proposta da NVIDIA, basata su API per consentire una programmazione a blocchi, la quale è attualmente la più utilizzata nel settore.
Nonostante gli sforzi di programmazione per lo sfruttamento delle capacità di queste piattaforme di calcolo, gli sviluppatori devono far fronte ad un'architettura orientata massicciamente sulla parallellizzazione, molto differente da quella di calcolo tradizionale. La possibilità di realizzare cluster eterogenei attraverso l'interazione tra GPUs e CPUs porta ulteriori problematiche per lo sviluppo del codice, nel quale oltre alla parallellizzazione degli algoritmi occore tener conto al meglio anche dei tempi di latenza di comunicazione tra le componenti. Per questi motivi la programmazione su queste piattaforme è ancora una sfida.

1.2 Un esempio di Big Data: Gene Expression Microarray

Le piattaforme sperimentali high throughput, come la spettrometria di massa, microarray e next generation sequencing stanno producendo un volume considerevole di dati appartenenti al settore, cosiddetto, *omico*. Tra le tante discipline omiche, genomica, proteomica ed interatomica stanno guadagnando un interesse crescente nella comunità scientifica per lo studio di patologie a livello molecolare.
In questo lavoro di tesi si è scelto di analizzare datasets ottenuti attraverso la tecnologia microarray, ragion per cui è necessario fare un po' di chiarezza sul metodo.
L'introduzione dei microarray ha consentito una visione completamente nuova sulla biologia del DNA e, quindi, un arricchimento nello studio dei sistemi viventi. Il DNA

è un molecola fisica in grado di codificare informazioni in una struttura lineare. La codifica avviene mediante geni ed elementi regolatori che ne controllano l'accensione o spegnimento. In questo modo il DNA può essere descritto in differenti stati, a seconda della configurazione dei geni. La tecnologia microarray consente appunto la lettura di questi stati.

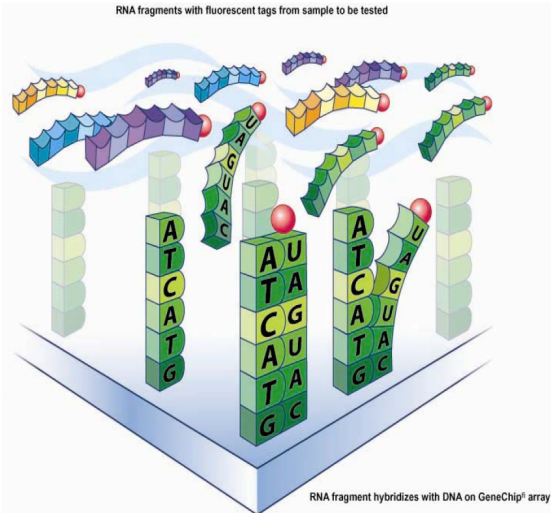


Figura 1.6: Schema del processo di ibridizzazione: la singola catena nucleotidica posta sul vetrino viene completata dal filamento inserito e precedentemente reso fluorescente. L'accoppiamento delle basi soddisfa la regola A-T e C-G per i filamenti di DNA complementari, mentre l'uracile U, presente solo nell'RNA, completa la base A nelle misurazioni di mRNA come in figura.

Mentre fino a qualche anno fa la raccolta dei dati di espressione genica richiedeva lunghi processi ed un duro lavoro per la generazione e lo stoccaggio, l'elevata parallelizzazione del microarray consente un notevole incremento nella velocità di raccolta dati e di svolgimento degli esperimenti. Da un punto di vista statistico, infatti, ogni misura di espressione genica nel microarray costituisce una variabile indipendente in un esperimento ad alta parallelizzazione.

I microarray misurano eventi all'interno del genoma. Un evento può essere dato dalla trascrizione di un gene, dalla codifica di un segmento di DNA in proteina, dalla presenza o assenza di una mutazione, da una variazione del numero di copie in un locus, da un cambio di stato di metilazione del DNA, o da un qualsiasi cambiamento di stato o attività associati alle molecole di DNA o RNA. Come per le *readout* genomiche, i microarray identificano dove avvengono questi eventi.

Lo scopo è quello di esaminare l'espressione di più geni simultaneamente in risposta ad una perturbazione biologica. Più in generale, un microarray viene utilizzato per verificare la concentrazione di molecole in una miscela complessa e, quindi, come potente strumento per molte tipologie sperimentali.

Per comprenderne il funzionamento è necessario rifarsi alla struttura a doppia elica del DNA ed al principio di complementarietà delle basi. Le basi di una catena, infatti, interagiscono con le basi della catena complementare secondo le regole di accoppiamento A-T e C-G. In questo modo, conoscendo la sequenza di una delle due, per definizione, è immediata la sequenza della catena opposta. Questa proprietà di complementarietà è ciò che viene usato per la misurazione della gene expression nei microarray. In particolare, viene utilizzato il processo di *ibridizzazione*, ovvero l'appaiamento complementare dei due filamenti di DNA (oppure di un filamento di DNA e l'altro di RNA).

La misura standard per la misurazione della gene expression è l'ibridizzazione delle basi chiamata *Nothern Blot*. Il Nothern Blotting è una tecnica che permette di visualizzare ed identificare l'RNA purificato da un campione, proprio per studiarne l'espressione genica. L'analoga tecnica per il DNA viene detta Southern Blotting.

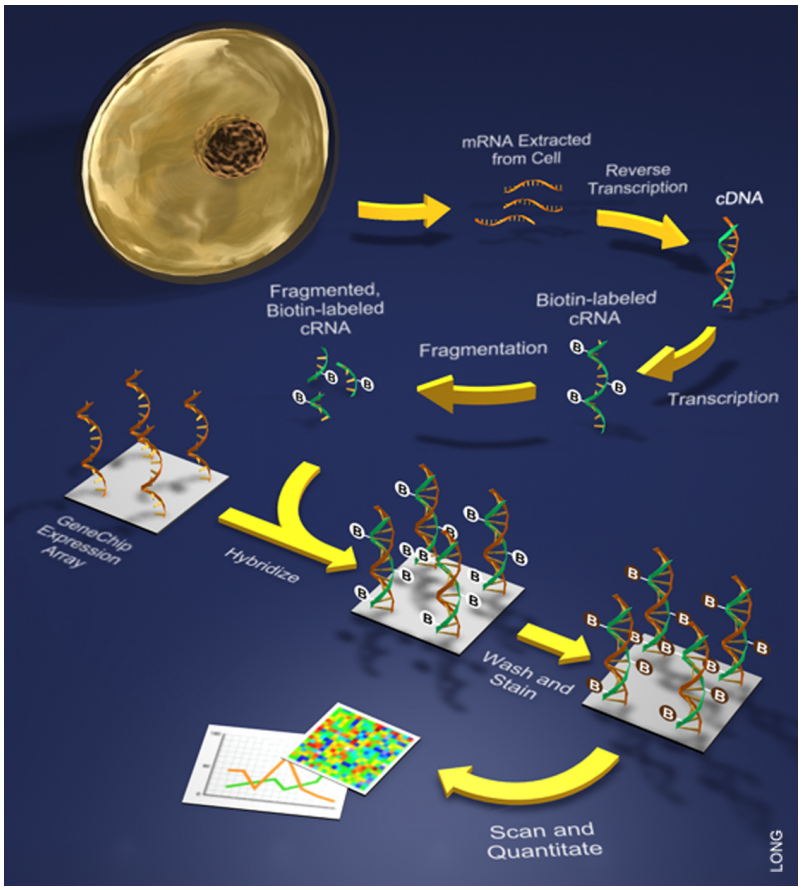


Figura 1.7: Utilizzo dell'Oligonucleotide Array: gli mRNA estratti dalle cellule sono amplificati attraverso un processo che etichetta l'RNA per le analisi. Il campione è poi applicato all'array ed ogni RNA fissato ed evidenziato.

Lo scopo del Nothern Blot è la misuarazione della dimensione ed abbondanza di RNA trascritto da un gene. Per il confronto di espressione genica sotto differenti condizioni, vengono estratti e preparati RNA cellulari provenienti da differenti cellule e tessuti e suddivisi in porzioni. In seguito le porzioni vengono disposte in un gel prima di essere poste su una membrana di nylon. La membrana si lega all'acido nucleico separando la struttura dell'RNA dal gel. Il risultato è una membrana porosa con RNA proveniente da campioni differenti disposti in regioni discrete della membrana.
Per quantificare il livello di espressione di un gene di interesse in ciascun campione viene preparata una sonda di DNA radioattivo complementare al gene. In questo modo l'ibridizzazione avverrà tra la sonda e l'RNA messaggero (mRNA) del gene di interesse, permettendo la visualizzazione sia della posizione che dell'abbondanza relativa. Quantitativamente, quindi, l'ammontare radioattivo di ciascun campione sulla membrana

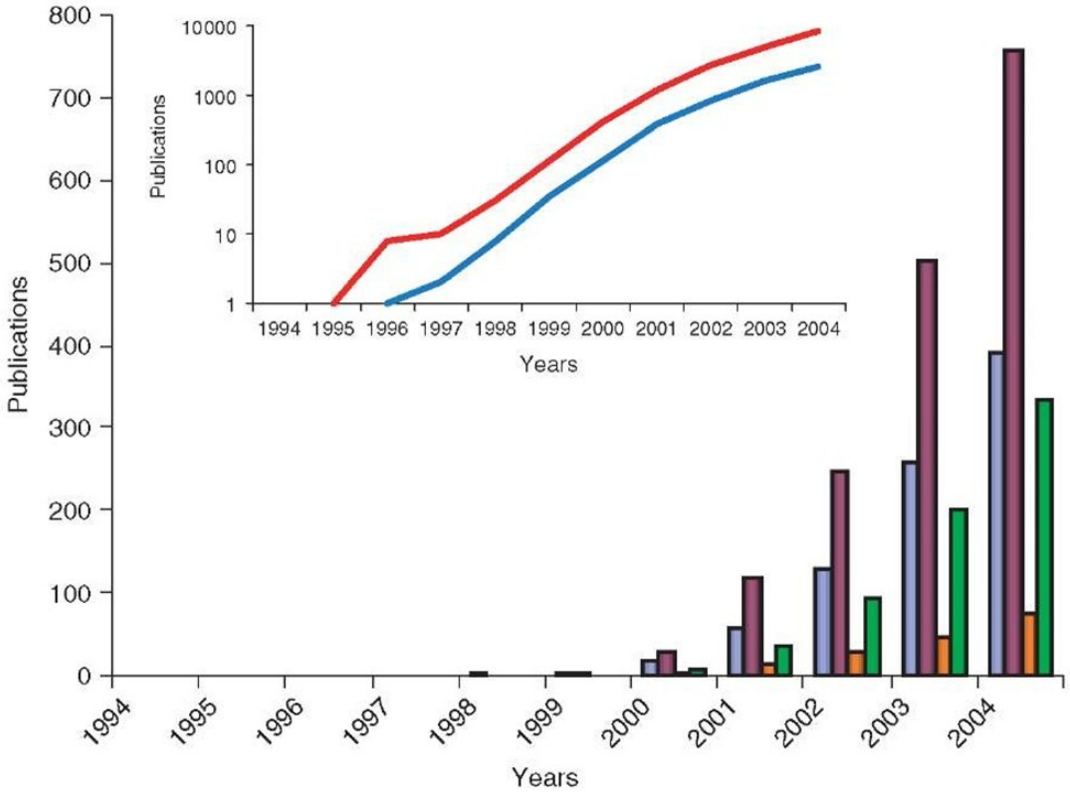


Figura 1.8: Il continuo aumento nell'utilizzo di microarray nella ricerca di malattie infettive, valutata ricercando le keywords 'microarray' (linea rossa, grafico in inserto), 'cancer and microarray' (linea blu, grafico in inserto), 'bacteria and microarray' (barra blu), 'immune and microarray' (barra verde) e '(host and pathogen) AND microarray' (barra arancione) per ogni anno, con il numero di pubblicazioni registrate. Anche se questa rappresenta una misura grezza e relativa, serve ad illustrare il trend di crescita della ricerca biologica sul cancro dovuto all'utilizzo dei microarray, connesso con altre aree della biologia come la ricerca di malattie infettive.

equivarrà al livello di espressione del gene di interesse tra i due campioni.
A livello pratico, il campione di mRNA disposto sulla membrana è dato da una miscela complessa di molecole di concentrazione sconosciuta. Da qualche parte nella regione discreta occupata dal campione di RNA è localizzata la stringa complementare, la cui posizione sarà rivelata in seguito all'ibridizzazione. In questo modo, la sonda è nota ed il target viene rivelato.
Sebbene il processo sia robusto risulta comunque limitato dalla ricerca di un bersaglio alla volta. La tecnologia microarray permette appunto la parallelizzazione di questo processo, utilizzando una superficie bidimensionale di molecole note o locazioni discrete di probes.
Dalla loro nascita sono seguiti numerosi sviluppi che hanno contribuito alla parallelizzazione attraverso la miniaturizzazione della strumentazione. L'uso di substrati vetrosi al posto di filtri porosi come materiali per la cattura dell'acido nucleico in un formato

matriciale ha permesso l'impiego di piccoli volumi di ibridizzazione. Il tasso di ibridizzazione è dipendente dalla concentrazione: volumi più piccoli comportano un aumento del livello di ibridizzazione ed una maggiore sensibilità. Inoltre, le superfici vetrose solide possiedono una bassa fluorescenza intrinseca, consentendo l'utilizzo di coloranti fluorescenti per i campioni noti e microscopi confocali, aumentando l'efficienza di quantificazione della risposta.

Capitolo 2

Metodi di Analisi per Big Data

Uno dei topic principali del settore bioinformatico e biomedico è proprio la ricerca di metodi per l'estrazione di informazioni utili all'interno delle massive raccolte di raw data. I metodi di Machine Learning per la riduzione del numero di variabili in esame ed una loro classificazione hanno contribuito a questo scopo. Al contempo l'utilizzo di avanzati modelli a network e la loro analisi topologica si è rivelata particolarmente efficace nell'integrare differenti tipologie di dati del dominio biomedico e nell'esplorare interazioni tra gli elementi.

Il legame tra la biologia ed il campo del Machine Learning a livello storico può essere ricondotto alla creazione del Single Perceptron come prima modellizzazione computazionale dell'apprendimento nervoso. Da questo si è poi potuto sviluppare tutto il campo delle cosiddette *Artificial Neural Networks* (ANN). L'idea di base del Machine Learning è, infatti, quella di ricreare a livello artificiale ed algoritmico sistemi di apprendimento con cui poter addestrare e far apprendere un calcolatore. Istruire una macchina a compiere scelte si riduce spesso alla dicotomia di una classificazione degli elementi tra due classi. Mediante classificazione è possibile decidere se particolari pattern biologici possono essere infetti oppure no ed identificare il numero di informazioni sufficienti e significative per poter esprimere questa scelta.

L'utilizzo di networks biologici per modellizzare ed integrare dati biomedici eterogenei, invece, sta crescendo, portando l'approccio della System Biology al centro delle ricerche del settore. I networks, infatti, permettono una modellizzazione di varie tipologie di relazione nel campo bioinformatico, partendo dalle interazioni tra geni, interazioni protein-protein, relazioni gene-malattia e correlazioni tra espressioni geniche. Gli studi iniziali sulle relazioni protein-protein mediante networks hanno indicato la presenza di comportamenti statistici simili, spesso legati alle distribuzioni delle misure di centralità, evidenziando una disomogeneità nella significatività degli elementi: solo pochi elementi sono caratterizzati da alti livelli di connessione all'interno del network, mentre la maggior parte hanno funzione marginale nella rete. La presenza di clusters e di *hubs*[1] all'interno delle reti biologiche ha dimostrato l'importanza delle cooperazioni tra

[1] Nodi ad elevata connettività circondati da nodi con basso numero di link.

più elementi per la risoluzione dei problemi e l'individuazione delle cause di numerosi fenomeni.
A livello computazionale l'applicazione ai Big Data implica l'ottimizzazione degli algoritmi standard. In base al linguaggio di programmazione utilizzato sarà necessario manipolare più o meno pesantemente il codice per raggiungere questo obiettivo. Nel seguito faremo riferimento ai linguaggi Python e C++, entrambi utilizzati in questo lavoro di tesi per la fase di prototipizzazione ed implementazione finale, rispettivamente.
Partendo da una spiegazione teorica delle techine di Machine Learning e della fisica dei networks, si prenderanno in considerazione i metodi utilizzati nel presente lavoro, con riferimento anche all'implementazione numerica degli algoritmi.

2.1 Machine Learning

Il problema della classificazione di dati è di rilevante interesse per la ricerca moderna. Con tale metodo, infatti, è possibile non solo associare un determinato dato ad una cosiddetta *classe* di eventi già noti, ma anche sfruttare gli algoritmi di calcolo per numerosi settori di ricerca e sviluppo scientifico.
Tutto ciò è realizzato mediante algoritmi di Pattern Recognition (PR), andando ad *insegnare* alla macchina come comportarsi di fronte a determinate situazioni (per questo motivo il procedimento è detto anche *Machine Learning*). Il campo di applicazione dei PR è connesso con la ricerca automatica di regolarità nei dati a disposizione mediante l'utilizzo di un algoritmo computerizzato e con l'utilizzo di queste regolarità per l'esecuzione di azioni di classificazione dei dati in differenti categorie (o classi).

Figura 2.1: Esempio di *face detection*. (a) Input Image, (b) Output del classificatore, dove vengono individuati 5 volti in posizioni differenti. Questo risultato è stato ottenuto utilizzando la demo online nel sito *http://demo.pittpatt.com/*. Il classificatore è stato addestrato su 1000 immagini di volti etichettati manualmente e non-volti. Successivamente è stata valutata la sua efficienza nella sovrapposizione di patch con immagini di prova. Per la valutazione è stata considerata la probabilità del patch di contenere un viso.

Un classico esempio applicativo è quello del riconoscimento di volti in una sequenza video o del riconoscimento della scrittura manuale, mediante matrici di pixel. Per l'ottimizzazione dei risultati, durante l'insegnamento alla macchina vengono utlizzati set di N dati $\{\mathbf{x_1}, ..., \mathbf{x_N}\}$ detti *training set* al fine di regolare i parametri imposti nell'algorit-

mo per il riconoscimento. Per la classificazione possiamo utilizzare un cosiddetto *target vector* $\mathbf{t}$ che rappresenti l'identità del dato in esame. Il risultato del calcolo dell'algoritmo può essere espresso come una funzione $\mathbf{y}(\mathbf{x})$ che prende in input il dato $\mathbf{x}$ e crea un vettore di output $\mathbf{y}$, codificato nello stesso modo del target vector. La forma della funzione è determinata durante la fase di training, anche detta di *apprendimento*, sulla base dei dati forniti. Una volta che il modello è stato allenato, può essere utilizzato per la classificazione di nuovi dati *mai* visti prima, che chiameremo *test set*. L'abilità di classificazione di nuovi esempi differenti da quelli utilizzati per l'allenamento è detta *generalizzazione*.
Per numerose applicazioni, i dati di input vengono *pre-processati* per trasformarli in un vettore di variabili e poterli analizzare nel loro spazio multi-dimensionale, in modo da facilitare il problema di riconoscimento: prendendo come esempio quello del riconoscimento di volti da frame di una sequenza video, non sarà utilizzata l'intera matrice di pixel delle immagini ma solamente una box di pixel di dimensioni fissata. Questo preprocessing è detto anche *features extraction*: in fase di generalizzazione il dato dovrà essere pre-processato attraverso gli stessi criteri. Inoltre, l'algoritmo di classificazione dovrà possedere una velocità adeguata di calcolo in comparazione con l'utilizzo a cui è preposto: per questo motivo un numero di features inferiore, come può essere l'utilizzo di una box di pixel nell'immagine, è preferibile e consiste in una *riduzione della dimensionalità* del problema.
Attraverso le sezioni esposte nel seguito verranno mostrati i punti salienti ed i metodi principali per l'analisi mediante PR.

2.1.1 Passi del Pattern Recognition

La procedura di PR è composta da numerose fasi, eventualmente dipendenti dall'approccio adottato per la raccolta dei dati. In linea generale, i passi da seguire comuni alla maggior parte dei casi sono:

- **Accentramento dei dati**: In questa parte si accentrano tutti i dati da utilizzare, sia quelli che useremo per l'*addestramento* (*training*) sia quelli per il *testing* (applicazione alla generalità). Questa fase, seppur apparentemente banale è fondamentale: è di importanza cruciale, infatti, quantificare correttamente il numero di esempi da utillizzare come training e quello da utilizzare come test per far sì che l'algoritmo di PR offra performances ottimali (metodi *Houldout*, cfr. Sez. 2.3);

- **Scelta delle features**: É il passo fondamentale del PR. In questa fase entrano in gioco i cosiddetti algoritmi e metodi di *Features Selection* e *Features Extraction* (cfr. Sez. 2.2). Per ogni dato si avranno a disposizione diverse misure caratteristiche. Per ottenere una corretta classificazione sarà necessario capire quali tra queste grandezze (o *features*) siano rilevanti al fine della classificazione, ove per *rilevanti* si intende la capacità di buona discriminazione di classi differenti. Non è detto che una singola feature sia sufficiente per avere una buona separazione: al

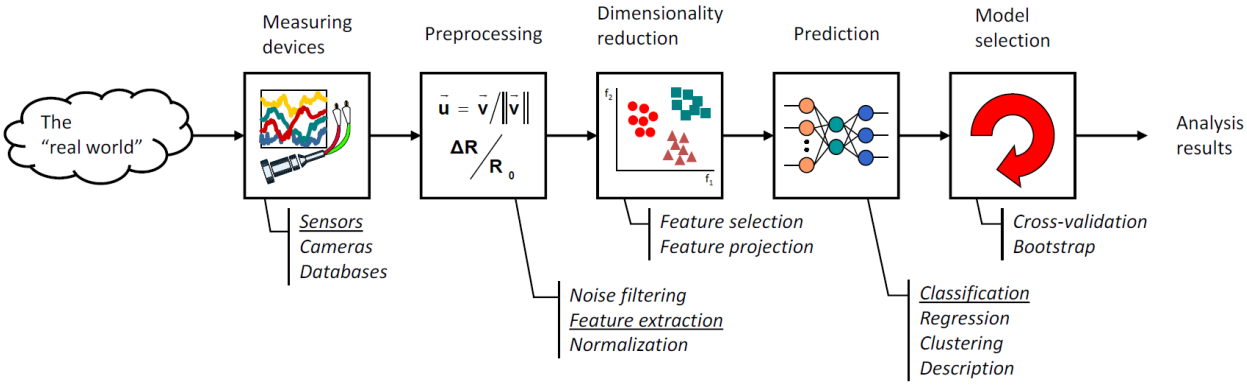


Figura 2.2: Schema degli step del Pattern Recognition. Una volta raccolte le misure delle variabili del sistema, queste vengono preprocessate, fino all'estrazione del numero il più possibile ridotto ed efficiente per la successiva classificazione. Al termine del processo di classificazione, il modello applicato necessita una validazione utilizzando nuovi dati mai forniti durante la fase di addestramento della macchina.

contrario, in molti casi, sono necessarie diverse features oltre alla combinazione di alcune di loro.

- **Scelta del modello**: In questa fase si sceglie quale tra i diversi approcci di PR si adotterà nella costruzione del classificatore, ossia la funzione matematica o la serie di calcoli da utilizzare per stabilire la classificazione dei dati in possesso (o *pattern*). Tale scelta è solitamente forzata dal grado di informazioni che si possiede sui dati: se si ha una grande informazione sulla statistica delle features dei nostri dati (o la distribuzione di probabilità o un numero sufficiente di dati per poterne fornire una stima) si adotta un approccio di tipo statistico, eventualmente Bayesiano (cfr. Sez. 2.1.2). In assenza di tali informazioni, i principali approcci sono mediante *Reti Neurali*, ispirate al funzionamento del sistema nervoso, che operano discriminazioni rigide dei dati, e l'*Approccio Strutturale*, utlizzato per features nominali, cercando di stabilire una struttura tra i dati.

- **Addestramento**: Scelto l'approccio al PR da adottare, occorre regolarne i parametri caratteristici (i parametri della distribuzione per un approccio statistico, i pesi sinaptici per un approccio neurale, la struttura per un approccio strutturale). Per fare ciò, si *addestra* il classificatore. I principali approcci sono due: *supervisionato* e *non supervisionato*.

  Nell'addestramento *supervisionato* al nostro programma viene fornito un pattern, ossia un insieme di coppie (dato, etichetta), ove l'etichetta identifica la classe a cui il dato appartiene. Settati dei parametri iniziali, il programma classifica gli esempi. Finita la classificazione, la bontà del processo viene sondata mediante una *funzione di merito*, come ad esempio l'errore commesso. Se la funzione di merito è accettabile, ossia con un errore al di sotto di una soglia scelta, i pa-

rametri vengono mantenuti, altrimenti vengono riaggiustati per poi ricominciare con la somministrazione dei pattern fino al raggiungimento della condizione di accettabilità.

Nell'addestramento *non supervisionato* i dati sono sprovvisti dell'etichetta, ma vengono distribuiti dall'algoritmo secondo vari criteri: minimizzazione delle distanze intra-classe e massimizzazione delle distanze extra-classe, trial and error, ecc. In generale, questo addestramento è utilizzato nei casi in cui il numero di dati etichettati sia molto esiguo. Alla fine di ogni iterazione viene calcolata una funzione di merito appropriata all'approccio adottato[2] e si setteranno i parametri dell'algoritmo, per poi riprocedere alla classificazione fino alla stabilità della funzione di merito;

- **Test o generalizzazione**: É l'ultima fase del PR. Si danno al programma un ultimo set di dati di esempio *noti a priori*. Questi dati non devono mai essere usati per l'addestramento e devono essere visti in questo momento per la prima volta dal classificatore. Quest'ultima fase serve per verificarne la bontà, valutando al contempo l'eventuale presenza di *overfitting*, ovvero l'eccessiva aderenza del classificatore ai dati utilizzati nel training.

2.1.2 Teorema di Bayes

In un approccio di tipo statistico al PR, è possibile utilizzare il *Teorema di Bayes* al fine di effettuare la procedura di classificazione: in quest'ottica, infatti, per la determinazione della classe w_i di appartenza di un elemento $\mathbf{x}$ (in generale questo elemento sarà descritto da un set di caratteristiche e questo gli conferisce la natura vettoriale) si utilizza la seguente *funzione di merito* alla ricerca della classe i-esima che la **massimizzi**:

$$g_i(\mathbf{x}) = P(w_i|\mathbf{x}) \propto P(\mathbf{x}|w_i)P(w_i)$$

dove $P(w_i|\mathbf{x})$ è la *probabilità a posteriori*, $P(\mathbf{x}|w_i)$ la *verosimiglianza* (ossia la distribuzione delle grandezze $\mathbf{x}$ all'interno dell'i-esima classe) e $P(w_i)$ la *probabilità a priori* di $\mathbf{x}$. La nostra funzione di merito è la forma ridotta del teorema di Bayes.
In altre parole l'approccio Bayesiano al PR consiste nella *massimizzazione della probabilità a posteriori* che l'elemento $\mathbf{x}$ appartenga alla classe w_i. Tale approccio tuttavia richiede un'elevata conoscenza della statistica del problema: le distribuzioni $P(w_i)$ e $P(\mathbf{x}|w_i)$ devono essere conosciute in modo soddisfacente (o a priori o stimate), altrimento il classificatore Bayesiano *non è applicabile* (*condizione necessaria*).
Si può dimostrare che un classificatore siffatto è quello dotato del minor tasso di errore in confronto a qualunque altro tipo di classificatore. L'errore di classificazione in questo

[2] Si noti che in questo caso non sarà possibile utilizzare l'errore commesso in virtù del fatto che non si dispone a priori delle etichette delle classi.

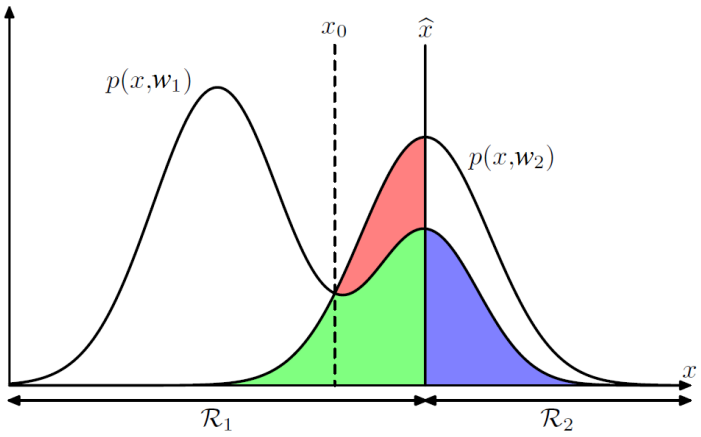


Figura 2.3: Illustrazione schematica della probabilità congiunta $p(x, w_i)$ per ognuna delle due classi rispetto ad x, insieme con la curva di discriminazione $x = \hat{x}$. I valori di $x \geq \hat{x}$ sono classificati nella classe w_2 e quindi appartengono alla regione $\mathcal{R}_2$, mentre i punti con $x < \hat{x}$ sono classificati in w_1 e appartengono alla regione $\mathcal{R}_1$. Gli errori insorgono nelle regioni blu, verde e rossa, poiché per $x < \hat{x}$ gli errori sono causati dai punti di w_2 misclassificati in w_1 (rappresentati dalla somma delle regioni verde e rossa), ed al contrario per i punti nella regione $x \geq \hat{x}$ che sono classificati in w_1 mentre appartengono a w_2 (rappresentati dalla regione blu). Variando la posizione della discriminante $\hat{x}$, la combinazione delle regioni verde e blu rimane costante, mentre la dimensione della regione rossa varia. La scelta ottimale di $\hat{x}$ è dove le curve $p(x, w_1)$ e $p(x, w_2)$ si incrociano, ovvero nel punto $\hat{x} = x_0$, poiché in questo caso la regione rossa scompare. Questo è equivalente alla regola di minimo errore di classificazione, la quale assegna ad ogni valore di x la classe che possiede la più alta probabilità a posteriori $P(w_i|x)$.

caso sarà dato dalla probabilità a posteriori di misclassicazione, quindi $Err(w_i|\mathbf{x})$, la quale per i lemmi della probabilità equivale a

$$Err(w_i|\mathbf{x}) = 1 - P(w_i|\mathbf{x})$$

da cui si dimostra che la funzione discriminante dedotta dall'approccio Bayesiano è quella che ne minimizza la quantità. Pertanto nella scelta dell'algoritmo da utilizzare è importante vedere se è possibile usare un metodo PR riconducibile ad un classificatore Bayesiano: in tal modo le performances della macchina miglioreranno sensibilmente.

2.1.3 Funzioni Discriminanti

Obiettivo del PR è assegnare ogni punto (feature) dello spazio iniziale ad una particolare classe. Questo comporta la suddivisione dello spazio (*input space*) in *regioni di decisione* $\mathcal{R}_w$, tali che un punto appartenente ad $\mathcal{R}_w$ è assegnato alla classe w. In totale generalità, una regione $\mathcal{R}_w$ non deve necessariamente essere continua, ma può essere descritta da

una serie di regioni disgiunte, tutte associate alla classe w. I confini (*boundaries*) tra queste regioni sono dette *decision boundaries*.
Per praticità consideriamo uno spazio delle features (x) 1-dimensionale e due sole classi w_1 e w_2. Un criterio ragionevole per il posizionamento delle *decision boundaries* è ottenuto calcolando il minimo della probabilità di misclassificazione. Per stimare questa probabilità occorre considerare i due possibili casi che portano alla classificazione errata del punto:

1. Assegnazione di x a w_1 mentre appartiene a w_2 (x cade nella regione $\mathcal{R}_1$ quando appartiene a w_2).
2. Assegnazione di x a w_2 mentre appartiene a w_1 (x cade nella regione $\mathcal{R}_2$ quando appartiene a w_1).

La probabilità totale di errore è data dalla somma di queste due casistiche possibili:

$$P[errore] = P(x \in \mathcal{R}_2, w_1) + P(x \in \mathcal{R}_1, w_2)$$

Espandendo i termini sulla destra mediante le probabilità condizionate si ottiene:

$$P[errore] = P(x \in \mathcal{R}_2|w_1)P(w_1) + P(x \in \mathcal{R}_1|w_2)P(w_2)$$

Possiamo adesso ottenere la probabilità $P(x \in \mathcal{R}_2|w_1)$ integrando $p(x|w_1)$ in $\mathcal{R}_2$ ed analogamente per $P(x \in \mathcal{R}_1|w_2)$, riscrivendo:

$$P[errore] = \int_{\mathcal{R}_2} p(x|w_1)P(w_1)dx + \int_{\mathcal{R}_1} p(x|w_2)P(w_2)dx$$

La minimizzazione della probabilità di misclassificazione è equivalente alla minimizzazione di $P[errore]$. Questo ci porta a concludere che, per un dato x:

- se $p(x|w_1)P(w_1) > p(x|w_2)P(w_2)$, allora il punto x dovrebbe essere in $\mathcal{R}_1$;
- se $p(x|w_1)P(w_1) < p(x|w_2)P(w_2)$, allora il punto x dovrebbe essere in $\mathcal{R}_2$.

La probabilità di misclassificazione è quindi minimizzata assegnando ogni punto alla classe con il massimo di probabilità a posteriori.
Nel caso in cui non si abbia a disposizione l'informazione inerente la distribuzione statistica dei dati in possesso, ma solamente un'informazione parametrica sommaria, è possibile ottenere una buona classificazione dei dati mediante lo studio delle cosiddette *funzioni discriminanti*, ovvero delle funzioni che non contemplino la statistica Bayesiana (anche se è possibile poi ricondurre gli stessi concetti ricavati ad interpretazioni Bayesiane) ma che sappiano dividere lo spazio delle features.

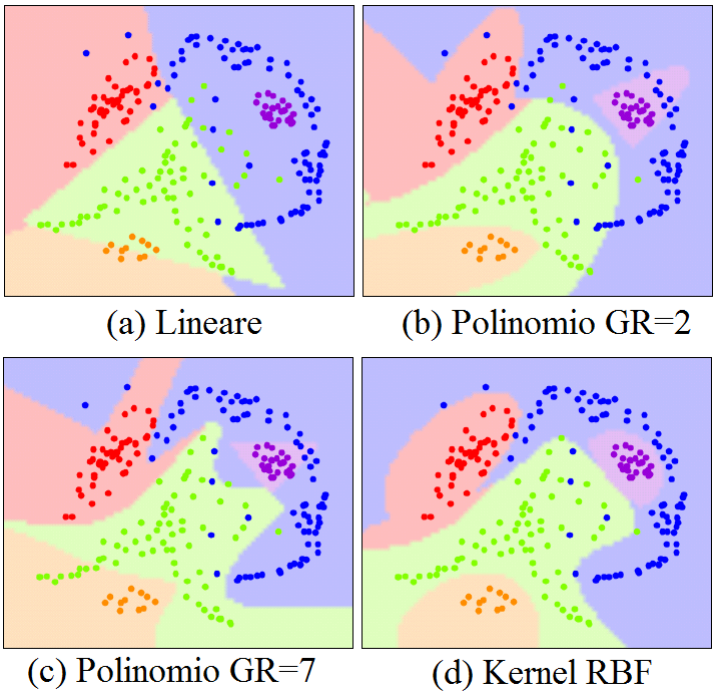


Figura 2.4: Esempi di funzioni discriminanti applicate per la classificazione di dati appartenenti a più classi: funzione discriminante di tipo lineare (a), polinomiale di secondo grado (b), polinomiale di settimo grado (c) e di tipologia più complessa utilizzando le Radial Basic Function. Con l'avanzare dell'ordine e della complessità della funzione, la curva separatrice tende ad adattarsi sempre meglio alla disposizione dei dati, consentendo una migliore classificazione. Al contempo però, l'eccessiva flessibilità della curva può condurre a problemi di generalizzazione del modello, in quanto troppo aderente alla disposizione dei dati di training.

Se abbiamo a disposizione K classi è necessario definire K *funzioni discriminanti* $y_k(\mathbf{x})$, una per ogni classe. La feature $\mathbf{x}$ sarà assegnata alla classe w_i se

$$y_{w_i}(\mathbf{x}) > y_{w_j}(\mathbf{x}) \quad \text{per tutti i } j \neq i$$

In altre parole: assegniamo $\mathbf{x}$ alla classe w_i la cui funzione discriminante $y_{w_i}(\mathbf{x})$ è più grande.

Da un punto di vista Bayesiano, questo equivale alla classificazione basata sul valore della probabilità a posteriori (o il suo logaritmo). Quindi la probabilità a posteriori è interpretabile come una possibile funzione discriminante. Anche in questo caso, la scelta di classificare i punti in relazione al più alto valore della probabilità a posteriori equivale alla minimizzazione della probabilità di misclassificazione. Si dimostra, inoltre, che tale regola costituisce la miglior funzione discriminante possibile.

Una *decision boundary* si trova in quei punti dell'*input space* in cui le funzioni discriminanti sono uguali.

$$y_{w_i}(\mathbf{x}) = y_{w_j}(\mathbf{x}) \quad \textit{decision boundary}$$

Le *decision boundaries* rimangono invariate per trasformazioni monotone (come il calcolo del logaritmo) delle funzioni discriminanti.
Il metodo di classificazione mediante funzione discriminante riesce senz'altro a semplificare il calcolo dell'algoritmo, non dovendo contemplare funzioni di distribuzioni, pagando il prezzo di avere un minor numero di informazioni sul problema ed una classificazione meno accurata. L'algoritmo denominato Support Vector Machine sfrutta proprio tale metodologia, evitando il problema di Bayes, focalizzandosi su quella che è l'effettiva classificazione dei dati (in questo caso con la curva di maggior margine tra le classi). Il metodo è utilizzato quindi quando si ha una conoscenza non completa sui dati, fattore che lo rende applicabile ad un'elevata casistica di problemi.

2.1.4 Classificatori Bayesiani per classi Distribuite Normalmente

La classica equazione mono-dimensionale della distribuzione di Gauss per l'applicazione all'ambito del Machine Learning necessita una generalizzazione al caso d-dimensionale (Gaussiana multivariata), ossia

$$G(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma|^{1/2}} \cdot exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right]$$

dove $\mathbf{x}$ è un vettor colonna d-dimensionale, $\boldsymbol{\mu}$ è il vettore media della distribuzione, Σ è la matrice di covarianza $(d \times d)$, $|\Sigma|$ e Σ^{-1} sono rispettivamente il determinante e l'inversa di Σ. Si noti che la dipendenza della funzione G dal vettore $\mathbf{x}$ è di tipo *quadratico*,

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

dove la quantità che compare ad esponente (Δ^2) è detta **distanza di Mahalanobis** del vettore $\mathbf{x}$ da quello di media, definizione che è ricondotta alla nota distanza euclidea nel caso in cui la matrice di covarianza corrisponda all'identità $\mathbf{I}$.
La matrice di covarianza è sempre simmetrica e definita positiva, pertanto ammette inversa. Se la matrice di covarianza è *diagonale*, la disribuzione multidimensionale è definita come semplice prodotto di d Normali monodimensionali. In tal caso gli assi principali sono paralleli agli assi cartesiani.
Partendo dall'espressione della distribuzione gaussiana multivariata (che nel caso di Pattern Recognition corrisponderà alla densità di probabilità condizionata), la regola di Bayes per la classificazione viene riscritta come

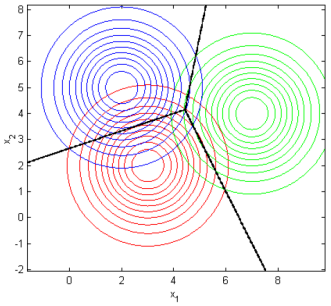
$$g_i(\mathbf{x}) = P(w_i|\mathbf{x}) = \frac{p(\mathbf{x}|w_i)P(w_i)}{p(\mathbf{x})} = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_i|^{1/2}} \cdot exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T {\Sigma_i}^{-1} (\mathbf{x} - \boldsymbol{\mu_i})\right] \frac{P(w_i)}{p(\mathbf{x})}$$

dalla quale, eliminando i termini costanti (fattori π e densità di probabilità assoluta $p(\mathbf{x}) = \sum_{i=1}^{s} p(\mathbf{x}|w_i) \cdot P(w_i)$) e sfruttando la monotonia della funzione, è possibile estrarre la relazione logaritmica, più facile da manipolare,

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T {\Sigma_i}^{-1}(\mathbf{x} - \boldsymbol{\mu_i}) - \frac{1}{2}\log|\Sigma_i| + \log P(w_i)$$

espressione che viene detta **funzione discriminante quadratica**.
La dipendenza della funzione è espressa dalla forma della matrice di covarianza, la quale può dar vita a 5 differenti casi:

- **Caso 1: $\Sigma_i = \sigma^2 I$ - DiagLinear Classifier**



Questo caso si verifica quando tutte le features sono statisticamente indipendenti con uguale varianza per tutte le classi. Questa ipotesi permette di semplificare la funzione discriminante che viene ad assumere la forma (dopo la banale sostituzione dell'espressione di Σ_i)

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x^T x} - 2\boldsymbol{\mu_i}^T\mathbf{x} + \boldsymbol{\mu_i}^T\boldsymbol{\mu_i}) + \log P(w_i)$$

ed eliminando i termini $\mathbf{x^T x}$ che risultano costanti per tutte le classi

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(-2\boldsymbol{\mu_i}^T\mathbf{x} + \boldsymbol{\mu_i}^T\boldsymbol{\mu_i}) + \log P(w_i) = \mathbf{w_i}^\mathbf{T}\mathbf{x} + \mathbf{w_0}$$

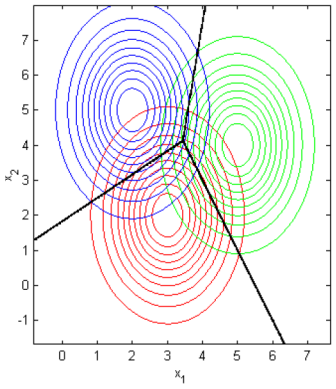
Attraverso queste semplificazioni si è giunti ad una funzione discriminante **lineare**, in cui le superfici di separazione ($g_i(\mathbf{x}) = g_j(\mathbf{x})$) sono iper-piani.

Se invece si sceglie di assumere uguale probabilità a priori, la funzione viene riscritta come

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x} - \boldsymbol{\mu_i})^T(\mathbf{x} - \boldsymbol{\mu_i})$$

detta *nearest mean classifier* (ovvero un classificatore Mahalanobis), in cui le superfici di equiprobabilità sono iper-sfere e nel caso in cui la varianza sia unitaria si ritorna alla semplice distanza euclidea.

- **Caso 2: $\Sigma_i = \Sigma$ (diagonale) - Linear Classifier**



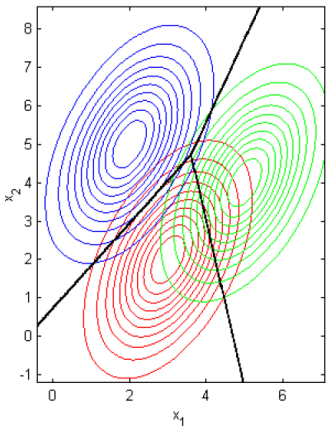
Le classi in questo caso continuano ad avere la stessa covarianza ma le features presentano ognuna una diversa varianza. Procedendo come in precedenza con il calcolo della funzione discriminante sviluppando il prodotto scalare dopo la sostituzione di Σ, si ottiene

$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{s}\frac{(\mathbf{x_k} - \boldsymbol{\mu_{i,k}})^2}{{\sigma_k}^2} - \frac{1}{2}\log\prod_{k=1}^{s}{\sigma_k}^2 + \log P(w_i)$$

nella quale è possibile ancora eliminare il termine $\mathbf{x_k}^2$ in quanto costante per tutte le classi, per arrivare ancora una volta ad una funzione discriminante di tipo **lineare** con superfici di separazione date da iper-piani e contorni di equiprobabilità dati da iper-ellissi.

Si noti che l'unica differenza rispetto al caso precedente risiede nel fatto che la distanza in ogni asse in questo caso viene normalizzata per la sua varianza.

- **Caso 3: $\Sigma_i = \Sigma$ (non diagonale) - Mahalanobis Classifier**



In questo caso si assume che le classi abbiano un uguale matrice di covarianza ma che essa non sia semplicemente diagonale. La funzione discriminante viene ad assumere la forma

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu_i}) - \frac{1}{2}\log|\Sigma| + \log P(w_i)$$

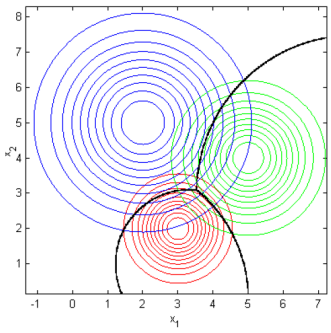
dalla quale è ancora possibile eliminare il termine $\log|\Sigma|$, costante per tutte le classi, ed assumere un uguale probabilità a priori, ottenendo la forma semplificata

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu_i})$$

Il termine quadratico è detto **distanza di Mahalanobis** e rappresenta una distanza normalizzata per l'inverso della matrice di covarianza, la quale opera uno stretching dello spazio.

Espandendo il prodotto scalare e rimuovendo il termine costante $\mathbf{x^T\Sigma^{-1}x}$ si dimostra che la funzione discriminante in questo caso risulta di tipo **lineare** con conseguenti superfici di separazione date da iper-piani e contorni di equiprobabilità dati da iper-ellissi di assi allineati agli autovettori della matrice Σ.

- **Caso 4: $\Sigma_i = {\sigma_i}^2 I$ - DiagQuadratic Classifier**

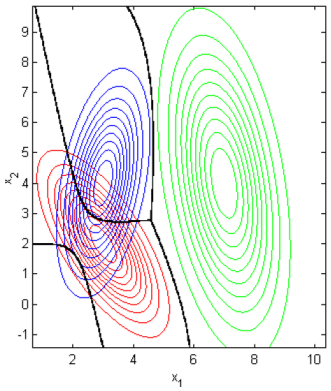


In questo caso ogni classe possiede una differente matrice di covarianza, la quale è però proporzionale all'identità e quindi diagonale. La funzione discriminante si semplifica nella forma

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_i})^T {\sigma_i}^{-2}(\mathbf{x}-\boldsymbol{\mu_i}) - \frac{1}{2}s\log\left|{\sigma_i}^2\right| + \log P(w_i)$$

la cui espressione non può essere ulteriormente ridotta e rendendo il classificatore di tipo **quadratico** con superfici di separazione date da iper-ellissi e contorni di equiprobabilità iper-sferici, allineati agli assi delle features.

- **Caso 5: $\Sigma_i \neq \Sigma_j$ (caso generale) - Quadratic Classifier**



Dall'espressione più generale della funzione discriminante, precedentemente ottenuta dalla sostituzione della densità di probabilità gaussiana, è possibile rinominare le varie componenti e mettere in evidenza la sua forma quadratica

$$g_i(\mathbf{x}) = \mathbf{x^T W_{2,i} x + w_{1,i}^T x + w_{0,i}} \quad \text{with} \quad \begin{cases} \mathbf{W_{2,i}} = -\frac{1}{2}{\Sigma_i}^{-1} \\ \mathbf{w_{1,i}} = {\Sigma_i}^{-1}\boldsymbol{\mu_i} \\ \mathbf{w_{0,i}} = -\frac{1}{2}\boldsymbol{\mu_i}^T{\Sigma_i}^{-1}\boldsymbol{\mu_i} - \frac{1}{2}\log|\Sigma_i| + \log P(w_i) \end{cases}$$

In questo caso, avendo ogni classe una propria matrice di covarianza Σ_i, i contorni di equiprobabilità sono iper-ellissi orientati con gli autovettori della matrice di ogni classe, mentre le superfici di separazione sono ancora di tipo **quadratico** e date da iper-ellissi o iper-paraboloidi.

Molto spesso si fanno ipotesi azzardate sulla "normalità" delle distribuzioni delle classi del problema senza aver sperimentalmente eseguito nessuna verifica. Ciò porta ad ottenere *cattivi risultati* di classificazione.

Pertanto, dato un problema con s classi e dato un training set (significativo), deve essere innanzitutto valutata la rispondenza alla "normalità" delle s distribuzioni. Questo può essere fatto in **modo formale** utilizzando un test statistico (es: test di *Malkovich-Afifi* basato sull'indice di Kolmogorov-Smirnov), oppure in **modo empirico** visualizzando in vari modi le nuvole dei dati o gli istogrammi sulle diverse componenti e confrontandoli con le curve teoriche.

2.1.5 Implementazione numerica del classificatore

Per quanto riguarda l'implementazione numerica di questa tipologia di classificatore, qualora si scelga di utilizzare linguaggi di programmazione come Matlab o Python, vengono già forniti funzioni altamente performanti a livello di tempistiche di calcolo e precisione numerica[3]. Tuttavia è utile fare un po' di chiarezza sul loro funzionamento ed il loro legame con la teoria precedentemente esposta.
Per l'implementazione in C++, invece, il classificatore deve essere implementato necessariamente manualmente, utilizzando al meglio le librerie a disposizione per ottimizzare i tempi di calcolo e la parallellizzazione. Di seguito analizzeremo il codice sviluppato in questo lavoro di tesi nel quale vengono utilizzate le librerie *Eigen* per l'ottimizzazione del calcolo matriciale.

Python - Classifier with Sklearn

L'utilizzo della funzione di classificazione Bayesiana risulta a prima vista non banale, in quanto la libreria Sklearn non mette a disposizione una sola funzione a più possibili parametri, ma bensì ogni tipologia di classificatore necessita un apposito *import*. Da una rapida ricerca all'interno del Reference [25] si nota subito l'assenza delle funzioni *diaglinear*, *diagquadratic* e *mahalanobis*[4], mentre persistono le nomenclature *linear* e *quadratic*, anche se con pratiche abbreviazioni (cfr. *sklearn.lda.LDA* e *sklearn.qda.QDA*). Questa apparente mancanza può essere facilmente colmata mediante l'aggiunta di poche righe di codice.
Prima di procedere è utile precisare meglio l'implementazione dei classificatori lineari e di Mahalanobis. Partendo dai casi lineari, avendo dati che difficilmente presenteranno una uguale matrice di covarianza tra classi, l'imposizione di questa condizione viene eseguita a livello numerico attraverso la stima della matrice media di covarianza di tutte le classi. Una volta calcolata, essa potrà essere utilizzata interamente (classificatore *linear*) oppure considerandone solo i valori diagonali (classificatore *diaglinear*). A proposito del classificatore *mahalanobis*, invece, il problema della scelta della matrice di

[3] Nota: mentre per Matlab le funzioni di classificazione statistica, *classify*, è presente senza l'aggiunta di particolari Toolbox, per Python faremo riferimento alle funzioni presenti nella libreria *Sklearn* [25], attualmente una delle migliori e più complete nell'ambito del Machine Learning.

[4] La funzione "Mahalanobis" in realtà è presente nella libreria ma non viene previsto un classificatore che la utilizzi. La funzione infatti, restituisce solamente il calcolo della distanza di Mahalanobis stimata rispetto alla matrice di covarianza.

covarianza non viene posto, andando a considerare l'effettiva matrice di covarianza di ogni classe. Questa scelta differisce dal *Caso 3* proposto, costituendone una generalizzazione. Come già anticipato, infatti, la *distanza Mahalanobis* non rappresenta a livello teorico di per sè un classificatore, ma solamente una metrica di distanza.

In Appendice A è riportato il codice relativo all'implementazione del classificatore, come funzione che riceve in input il vettore di training dei dati, il vettore di test, il vettore di label del training ed un numero $n \in [0, 4]$ per la scelta della tipologia di classificatore. Si noti che i classificatori LDA e QDA presenti in Sklearn permettono la creazione di funzioni discriminanti solamente di tipo lineare e quadratico, con matrici di covarianza complete: nel primo caso, avendo classi con inevitabile matrice di covarianza differente, viene considerata la matrice media delle varie classi, mentre nel secondo questa viene calcolata come previsto dalla teoria. Per i casi diagonali e Mahalanobis, invece, il metodo viene implementato seguendo esattamente la teoria esposta, ricavando i vettori di media e matrice/i di covarianza/e attraverso le funzioni della libreria Numpy[5].

Prima di concludere è doveroso ricordare che all'interno di Sklearn vi è un gruppo di metodi, denominati *Naive Bayes Classifier* che effettuano una classificazione Bayesiana particolare. Come già anticipato in precedenza, all'atto pratico non si è quasi mai nelle possibilità di fare assunzioni certe sulla forma della distribuzione e ancor più è difficile la conoscenza assoluta della distribuzione dei dati. Per questi motivi spesso l'utilizzo del classificatore di Bayes "rigoroso" risulta difficile se non impossibile ed è preferibile effettuare ulteriori approssimazioni. I metodi Naive Bayes sono un insieme di algoritmi di classificazione supervisionata basati sull'applicazione del teorema di Bayes con però l'assunzione di un'indipendenza tra ogni coppia di features. Questa assunzione porta, quindi, ad una forma diagonale della matrice di covarianza. Questo metodo "diagonale" viene implementato attraverso le funzioni NB di Sklearn, lasciando all'utente la libertà di scelta tra tre differenti densità di probabilità condizionata: *GaussianNB*, *MultinomialNB* e *BernoulliNB*. Per il nostro caso (GaussianNB) il metodo prevede un classificatore *diagquadratic*, tralasciando la possibilità del caso lineare. Per tale ragione è stato preferibile implementare il metodo manualmente.

C++ - Classifier with Eigen Library

L'utilizzo della Libreria Eigen per la manipolazione delle matrici aumenta l'efficienza di calcolo dell'algoritmo e ne facilita l'implementazione. Numerosi metodi implementati nella libreria, con l'attivazione della libreria OpenMP in fase di compilazione, sono già predisposti per applicazioni di calcolo parallelo multi-core. Inoltre, la gestione di memoria è ottimizzata per la manipolazione di ampie matrici, sia per allocazioni statiche che dinamiche. Riguardo l'algoritmo di classificazione, in particolare, la libreria trova applicazione nel calcolo della matrice inversa di covarianza ed anche per la stima di media e varianza.

[5] É pratica comune importare la libreria Numpy con l'abbreviazione *np*.

Ricalcando l'implementazione proposta in Python, l'algoritmo in C++ applicherà la tipologia di classificatore in relazione ad un valore numerico fornito in input. Per la generalizzazione del metodo, le label dei dati saranno considerate delle variabili stringa. Poiché numerose sezioni dell'algoritmo sono le medesime per tutte o alcune tipologie di classificatori, il codice presenta una struttura ramificata a partire dai calcoli comuni. La principale diramazione sarà tra le tipologie quadratiche (QDA, diagQDA e Mahalanobis) e quelle lineari (LDA e diagLDA), in relazione alla diversa matrice di covarianza che è necessario valutare.
In Appendice B è riportato il codice sviluppato in questo lavoro di tesi e successivamente utilizzato nell'algoritmo QDANet PRO (cfr. Cap. 3 per la spiegazione completa dell'algoritmo).

2.2 Riduzione della dimensionalità

Spesso nell'ambito del Machine Learning si lavora con dati ad alta dimensionalità. Tuttavia spesso le informazioni immagazzinate nei dati tendono ad essere ridondanti e racchiuse in un set molto più limitato di dati. Matematicamente questo si traduce in una correlazione tra le dimensioni dell'*input space* delle features: le informazioni più significative saranno racchiuse in solo alcune direzioni, mentre le altre risulteranno superflue. Questo costituisce l'obiettivo essenziale del Machine Learning, ovvero la ricerca del numero minimale di dati rilevanti per la descrizione il più possibile accurata del pattern.
La riduzione della dimensionalità è il metodo utilizzato per ridurre appunto il numero delle variabili, al fine di migliorare le performances dell'algoritmo in termini di tempistica computazionale e capacità discriminante. Per effettuarla è possibile operare su due passi del PR:

- **Features Selection**: ovvero la scelta delle caratteristiche dei dati a disposizione più performanti per la discriminazione.
- **Features Extraction**: ovvero l'insieme di calcoli di pre-processing che viene effettuato sulle features prima della somministrazione all'algoritmo.

2.2.1 Features Selection

Si tratta semplicemente di scegliere un sottoinsieme *ottimale* di features, tra tutte quelle di partenza, che massimizzi l'informazione contenuta e l'accuratezza di predizione. Corrisponde allo scegliere quelle che sono le caratteristiche dell'oggeto in esame che meglio lo descrivono e differenziano dagli oggetti delle altre classi, in termini di disposizione spaziale in quello che è lo spazio multi-dimensionale delle features.
In uno spazio in generale n-dimensionale, i metodi di feature selection cercano di trovare il migliore sottoinsieme, tra i 2^n sottoinsiemi candidati, in accordo con uno specifico

criterio. É immediato notare come questa pratica risulti costosa a livelli di tempistiche computazionali all'aumentare di n.

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \rightarrow \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix}$$

Per ovviare alla ricerca sistematica tra tutti i candidati, è possibile utilizzare tutta una serie di algoritmi, suddivisi in tre diverse classi:

- **Filter**: Selezionano un sottinsieme di variabili in fase di pre-processing, senza tener conto del metodo di classificazione che si implementerà in seguito. Ne è un esempio la *Funzione di Scoring* che permette una classificazione a ranking delle features mediante il valore assunto dalla funzione. Dato un insieme di vettori di training x_i con $i = 1, ..., m$, si indica il numero di istanze positive e negative rispettivamente con n_+ ed n_-, calcolando quello che è il valore *F-Score*, in cui il numeratore indica la discriminazione tra le due classi mentre il denominatore la discriminazione per singola classe.

- **Wrapper**: Utilizzano una determinata macchina per l'apprendimento come *black box*, per determinare il potere predittivo di un sottoinsieme di variabili. Con questo metodo però si necessita di una strategia di ricerca efficente e si contempla un costo dipendente dal metodo. Questo metodo predispone quindi quella che è una ricerca di tipo *Sequenziale* (metodi *Greedy*) andando ad includere/eliminare progressivamente nuove features in funzione delle loro capacità predittive.

- **Embedded**: Selezione delle variabili come parte del processo di training, integrando le features come variabili della funzione discriminante implementata nel classificatore. In questo modo minimizzando la funzione si scartano e migliorano le efficenze delle features.

2.2.2 Features Extraction

Dato uno spazio di features $x_i \in \Re^N$, la ricerca di una funzione $\mathbf{y} = f(\mathbf{x})$ che mappi lo spazio $\Re^N \rightarrow \Re^M$ con $M < N$ (ossia che permetta il passaggio da uno spazio a dimensione maggiore ad uno a dimensione inferiore), tale per cui il vettore delle features del nuovo spazio ($\mathbf{y} \in \Re^M$) preservi la maggior parte delle informazioni o comunque la struttura di $\Re^N$, è definito come problema di *features extraction*.
Nella pratica occorre quindi trovare quel set di caratteristiche, elaborate anche mediante calcoli di pre-processing, valutando combinazioni di features, che descriva al meglio il pattern da identificare, alla luce dell'intero set di caratteristiche ritrovate.
Naturalmente la funzione ottimale $\mathbf{y} = f(\mathbf{x})$ sarà quella che non accresce la probabilità di errore $P[errore]$. Quando si eseguono delle analisi di dati complessi, uno dei maggiori

problemi deriva dal numero di variabili coinvolte. L'analisi di un grosso numero di variabili spesso richiede un algoritmo di classificazione che provoca overfitting dei dati di training e quindi comporta errori di classificazione in fase di generalizzazione.
La ricerca delle dimensioni indipendenti tra loro all'interno delle features a disposizione è certamente un problema molto complesso: alcune dimensioni dovrebbero essere mantenute mentre altre sarebbe opportuno rimuoverle. In generale, la funzione ottimale sarà di tipo non lineare, ma data la difficoltà nel generare trasformazioni non lineari in maniera sistematica, ci occuperemo del solo caso lineare. Consideriamo allora un punto dello spazio multi-dimensionale iniziale $\mathbf{x}$ rappresentato da $\mathbf{y} = Fx$, dove $\mathbf{F}$ è una matrice non quadrata di dimensioni $dim(\mathbf{y}) \times dim(\mathbf{x})$, con $dim(\mathbf{y}) < dim(\mathbf{x})$. La matrice $\mathbf{F}$ rappresenta una proiezione lineare da uno spazio a dimensione maggiore $\mathbf{x}$ ad uno dimensione minore $\mathbf{y}$. La forma della matrice determina la tipologia di proiezione e, classicamente, sono possibili diverse scelte. Le scelte più popolari corrispondono alla *Principal Components Analysis* (PCA) ed alla *Linear Discriminant Analysis* (LDA o *Fisher Analysis*).
Si noti che questi metodi non descrivono nessun tipo di modello con cui generare dati e sono per questo non-probabilistici. Esistono tuttavia metodi alternativi e versioni differenti dei casi sopra citati a cui è associata un'interpretazione statistica.

Principal Component Analysis - PCA

Se i dati appartengono ad uno spazio multi-dimensionale, è auspicabile che questi giaciano il più vicino possibile ad un iperpiano. Possiamo allora approssimare ogni punto utilizzando i vettori che coprono l'iperpiano (*span*). Questi vettori costituiscono una "base" dello spazio in cui sono concentrati i dati. A livello pratico, stiamo cercando di scegliere un sistema di coordinate di uno spazio a più bassa dimensionalità che rappresenti in modo approssimato i dati. Matematicamente

$$\mathbf{x} \approx \mathbf{c} + \sum_{i=1}^{M} \alpha_i b^i$$

I vettori $\mathbf{b}^i$, con $i \in 1, \dots M$ sono scelti ortonormali. Se la dimensione dello spazio dei dati, $dim(\mathbf{x}) = N$, l'obiettivo è poter rappresentare i dati utilizzando solamente il più piccolo numero M di vettori.
Si può dimostrare che la rappresentazione ottimale in uno spazio a dimensione inferiore (ottimale nel senso di minimizzazione dell'errore quadrato di ricostruzione) è ottenuta proiettando i dati lungo gli autovettori della matrice di covarianza con gli M più grandi autovalori. A livello algoritmico, il metodo consiste in:

1. Trovare media e matrice di covarianza dei dati:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^i \qquad \mathbf{S} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}^i - \boldsymbol{\mu})(\mathbf{x}^i - \boldsymbol{\mu})^T$$

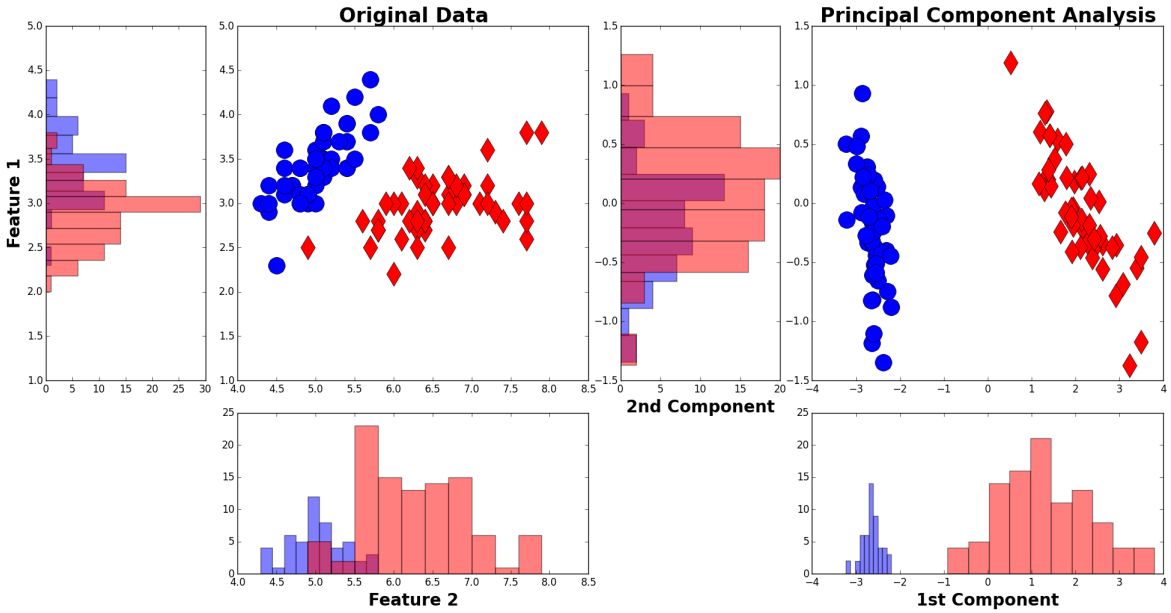


Figura 2.5: Calcolo delle prime due componenti principali di un set di dati. Il database di riferimento è l'*Iris Dataset* presente nei linguaggi di programmazione Python e Matlab. Nel grafico di sinistra sono riportate le features distribuite in uno spazio 2-D con relative proiezioni sugli assi: le proiezioni mettono in evidenza la difficoltà nel separare le due classi. A destra il grafico delle prime due componenti principali: in questo caso la proiezione dei dati lungo la 1st Component permette una separazione ottimale delle due classi, consentendo la riduzione della dimensionalità del problema ad 1 solo grado di libertà.

2. Trovare gli autovettori $\mathbf{e}^1, \ldots, \mathbf{e}^M$ della matrice di covarianza $\mathbf{S}$ con i più grandi autovalori. Costruire la matrice $\mathbf{E} = [\mathbf{e}^1, \ldots, \mathbf{e}^M]$ che ha i maggiori autovettori per colonna.

3. La minore dimensione di rappresentazione di ogni feature $\mathbf{x}^i$ è

$$\mathbf{x}^i \approx \boldsymbol{\mu} + \mathbf{E}\mathbf{y}^i$$

4. L'errore quadrato totale su tutti i dati di training dato dall'approssimazione è

$$Errore = (N-1) \sum_{j=M+1}^{P} \lambda_j$$

dove λ_j, con $j = M+1, \ldots, P$ sono gli autovalori scartati nella proiezione.

Si noti che di per sé gli autovettori calcolati non hanno alcun significato esplicito ma agiscono solo nella definizione del sottospazio lineare su cui sono proiettati i dati. Questo si può verificare considerando che ogni base che copre lo stesso sottospazio degli autovettori della matrice di covarianza è ugualmente valida per la rappresentazione dei dati.

Per quanto riguarda l'errore di ricostruzione è immediato verificare la sua forte dipendenza dai più grandi autovalori della matrice di covarianza. Graficando l'intero spettro degli autovalori, la speranza è che vi siano solo pochi autovettori grandi e la maggior parte piccoli: in particolare ci aspettiamo di riscontrare M autovalori grandi. Questi indicherebbero l'intrinseca dimensionalità dei dati ed i gradi di libertà significativi. Le direzioni corrispondenti ai minori autovalori sono interpretate come rumore.
Si noti che comunque un processo del genere comporta un'inevitabile perdita di informazione sulla effettiva variabilità del dato originale.

Linear Discriminant Analysis - LDA

Il *Fisher Linear Discriminant Analysis* (LDA) è un classico metodo che unisce l'obiettivo della riduzione della dimensionalità con quello della classificazione dei dati. L'LDA ha trovato molte applicazioni nell'analisi multivariata e nel Machine Learning: ne sono esempi il *face recognition* (Belhumeur et al., 1997; Martìnez & Kak, 2001), la *text classification*, la *microarray data classification*, ecc. Lo scopo della LDA è quello di ridurre le dimensioni e allo stesso tempo cercare di preservare quanta più informazione possibile sulla discriminazione delle classi. Per farlo, questa tecnica esamina le direzioni nelle quali i dati hanno massima varianza e successivamente proietta i dati in questa direzione. In questo modo, vengono rimosse le direzioni che producono solo "rumore", e otteniamo una rappresentazione dei dati con meno dimensioni. Matematicamente equivale alla ricerca della trasformazione di proiezione lineare che minimizzi la distanza totale delle classi e massimizzi la distanza tra le classi. É ben noto che questo problema di ottimizzazione possa essere risolto applicando la decomposizione secondo autovalori alle matrici di scatter.
Sia data allora la matrice $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]^T \in \mathbb{R}^{n \times p}$ dove gli $\mathbf{x_i}$ sono vettori di features p-dimensionali. Supponiamo che i dati iniziali siano classificabili in m classi, tali che $\mathbf{X}^T = [\mathbf{X_1}^T, \mathbf{X_2}^T, \ldots, \mathbf{X_m}^T]$ dove $\mathbf{X_i} \in \mathbb{R}^{n_i \times p}$ contiene n_i istanze della i-esima classe e $\sum_{i=1}^{m} n_i = n$. Il metodo LDA permette di trovare la trasformazione lineare ottimale $F \in \mathbb{R}^{p \times q}$ che preserva la struttura delle classi in un sottospazio tanto quanto nello spazio originale. In questo senso F mappa ogni $\mathbf{x_i}$ di $\mathbf{X}$ nello spazio p-dimensionale in un vettore $\mathbf{y_i}$ in uno spazio q-dimensionale.
Le matrici di scatter tra classi e totale sono definite come

$$S_w = \frac{1}{n} \sum_{i=1}^{m} \sum_{\mathbf{x} \in \mathbf{X_i}} (\mathbf{x} - \boldsymbol{\mu_i})(\mathbf{x} - \boldsymbol{\mu_i})^T$$

$$S_b = \frac{1}{n} \sum_{i=1}^{m} n_i (\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T$$

$$S_t = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x_i} - \boldsymbol{\mu})(\mathbf{x_i} - \boldsymbol{\mu})^T$$

dove $\boldsymbol{\mu_i} = 1/n_i \sum_{\mathbf{x} \in \mathbf{X_i}} \mathbf{x}$ è la media della j-esima classe e $\boldsymbol{\mu} = 1/n \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x}$ è la media

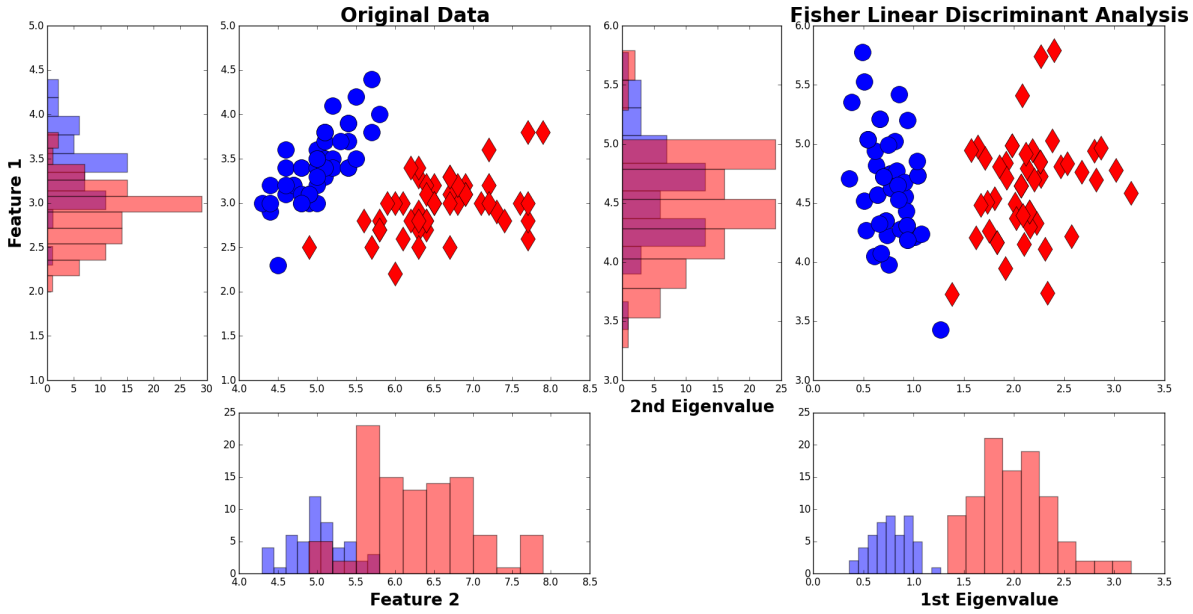


Figura 2.6: Calcolo del Fisher Discriminant di un set di dati. Il database di riferimento è il medesimo della Figura 2.5. A destra il grafico ottenuto dalla Linear Discriminant Analysis: in questo caso la proiezione dei dati lungo il 1st Eigenvalue permette una separazione ottimale delle due classi, consentendo la riduzione della dimensionalità del problema ad 1 solo grado di libertà.

dell'intero set di dati. Nel sottospazio definito dalla trasformazione F, le matrici di scatter diventano ${S_b}^F = F^T S_b F$ e ${S_t}^F = F^T S_t F$.
Una trasformazione F ottimale si può ottenere risolvendo il problema di ottimizzazione:

$$\arg\max_F \left\{ J(F) \triangleq trace\left(({S_t}^F)^{-1} {S_b}^F \right) \right\}$$

La soluzione dell'ottimizzazione può essere ottenuta attraverso la decomposizione in autovalori della matrice ${S_t}^{-1} S_b$ ovunque S_t è non singolare (Fukunaga, 1990). Comunque, quando S_t è non singolare possiamo utilizzare la decomposizione in autovalori di ${S_t}^{\dagger} S_b$, dove ${S_t}^{\dagger}$ è l'inversa di S_t secondo il criterio di Moore-Penrose.
L'applicazione di questo metodo richiede che la matrice totale di scatter sia non singolare, caso non sempre verificato nelle applicazioni reali. Ne è un esempio l'applicazione del metodo ai dati di microarray in cui si hanno a disposizione molte probes in relazione ai samples (*"large p but small n" regime*). Per risolvere il problema delle singolarità sono utili i metodi basati sulle matrici pseudoinverse e quelli di regolarizzazione (Hastie et al., 2009; Zhang et al., 2010). Recentemente sono stati proposti approcci di approssimazione a due step, come l'algoritmo PCA+LDA (Belhumeur et al., 1997) e l'algoritmo LDA/QR (Ye & Li, 2005). Tipicamente l'approccio a due step implica una piccola riduzione della dimensionalità iniziale prima dell'applicazione del convenzionale algoritmo LDA.

2.3 Houldout Methods

La maggior parte delle tecniche di PR hanno uno o più parametri liberi. In quest'ottica sorge il problema sul modello da utilizzare, ovvero sulla scelta del modello (*model selection*) e quindi sui parametri ottimali con cui applicarlo, ed, in seguito, sulla sua convalida (*validation*).
Se avessimo accesso ad un numero illimitato di campioni, i suddetti problemi si risolverebbero facilmente. Nella realtà, tuttavia, si ha accesso soltanto ad un numero ristretto. La soluzione più immediata (ma *errata*) è quella di utilizzare l'intero set di dati per l'addestramento, al fine di ottenere un classificatore maggiormente allenato. Questo approccio è scorretto per due principali ragioni:

- Il modello così ottenuto non potrebbe essere applicato a dei nuovi set di dati e dunque non è possibile verificarne la validità alla vista di esempi nuovi (impossibilità di test di generalizzazione);
- L'errore stimato risulterebbe inferiore rispetto all'errore reale (la macchina è allenata al meglio su determinati esempi ed ha potuto affinare i suoi parametri al meglio).

Un migliore e corretto approccio prevede la suddivisione dei dati per l'addestramento in sottoinsiemi disgiunti. Questo metodo è detto *Holdout Method* e consiste nel suddividere il set di dati in due sottoinsiemi etichettati come *Training Set*, ovvero i dati utilizzati per addestrare il classificatore, e *Test Set*, ovvero i dati utilizzati per stimare l'errore del classificatore addestrato durante la generalizzazione.
Anche tale metodo presenta tuttavia inconvenienti, provenienti principalmente da due sorgenti di problema:

1. Se abbiamo un set di dati molto *piccolo* potremmo non essere in grado di metterne da parte alcuni per comporre il Test Set.
2. Poiché addestramento e test avvengono in un singolo esperimento la stima dell'errore potrebbe essere non molto veritiera.

Queste limitazioni possono essere superate con altri metodi di ricampionamento aumentando però il costo computazionale. Tali metodi comprendono la *Cross Validation* (a sua volta divisibile nei metodi di *K-Fold Cross Validation* e *Leave-One-Out Cross Validation*) e *Bootstrap*.

2.3.1 K-Fold Cross Validation:

Qualora il set di dati di esempio sia limitato (magari a causa di difficoltà sperimetali o impossibilità di ripetere l'esperimento per ottenere un maggior numero di esempi con cui allenare l'algoritmo) viene spesso utilizzato un metodo di Cross Validation, ossia la suddivisione dei dati a disposizione in differenti categorie. Nella metodologia

denominata *K-Fold*[6] si procede alla ripartizione del set di dati a disposizione ad inizio esperimento in K gruppi, di cui K-1 vengono utilizzati per l'addestramento (verifica del/i modello/i in esame) ed il restante gruppo per il test di generalizzazione. Questa procedura è ripetuta per tutti i K gruppi scelti, andando a variare ogni volta il gruppo scelto per la generalizzazione[7]. In questo modo si ha il vantaggio che tutti gli esempi siano utilizzati, almeno una volta, sia per l'addestramento che per il test. L'errore reale è stimato come la media degli errori

$$Errore = \frac{1}{K}\sum_{i=1}^{K} Errore_i$$

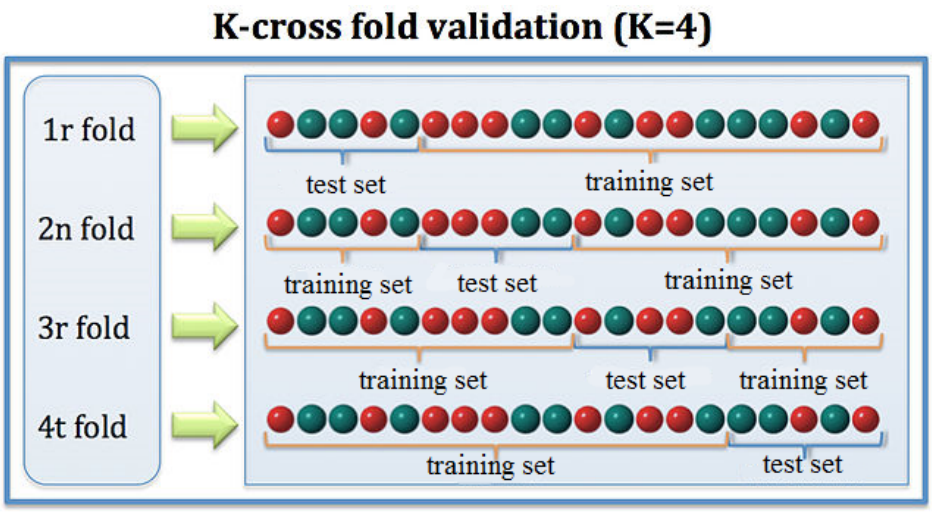


Figura 2.7: Schema del K-Fold Cross Validation con K=4. I dati sono suddivisi in due classi (verde e rosso) e vengono partizionati in training set e test set ad ogni iterazione. Il training set verrà utilizzato per l'addestramento della macchine ed il test set sarà accessibile solamente in fase di validazione.

A livello pratico questo metodo risulta molto simile a quello di tipo Bootstrap ed ha il vantaggio di poter stimare i parametri caratteristici del PR mediante la distribuzione dei risultati ottenuti. Problematica insita nel metodo (stessa poi del metodo Bootstrap) risiede nella velocità di calcolo, in quanto spesso potrebbe essere dispendioso andare a valutare un numero elevato di run del programma per tutti i K possibili.

2.3.2 Leave-One-Out Cross Validation

Rappresenta il caso limite del Metodo K-Fold Cross Validation, dove K è scelto pari al numero totale di esempi e non più come suddivisione arbitraria in gruppi (si passa

[6] Si noti che in alcuni testi la nomenclatura con cui viene identificato il metodo può cambiare ma il principio di funzionamento rimane il medesimo.

[7] Ad ogni run dell'algoritmo i parametri caratteristici del PR dovranno essere ri-settati ai valori di default in modo che si abbia un set di K training differenti e K risultati differenti per la generalizzazione, altrimenti la macchina allenerebbe i propri parametri anche in base ai dati di test

quindi da gruppi a singoli dati, in modo che la generalizzazione concerna 1 solo dato).

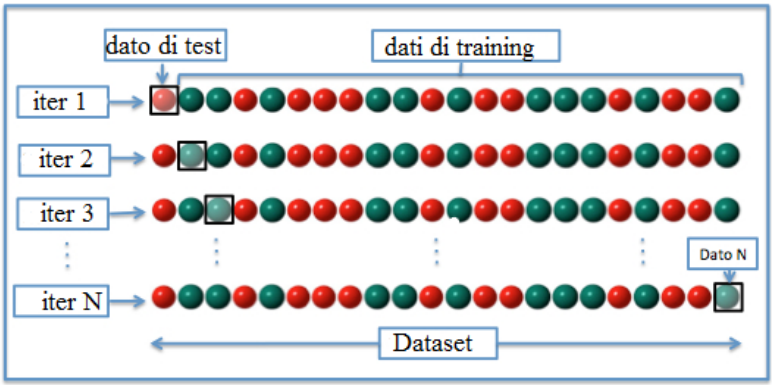


Figura 2.8: Schema del Leave-One-Out Cross Validation. I dati sono suddivisi in due classi (verde e rosso) e vengono partizionati in training set e test set ad ogni iterazione. Il training set verrà utilizzato per l'addestramento della macchine ed il test set sarà accessibile solamente in fase di validazione.

Dato un set di N esempi, si eseguono quindi N ripetizioni del metodo. Come per il Metodo K-Fold, per ogni ripetizione si utilizzano N-1 esempi per l'addestramento ed il restante esempio per il test di generalizzazione. Anche in questo caso l'errore reale è stimato come la media degli errori, una volta ricavata la distribuzione dei risultati delle N ripetizioni.
Per l'ottenimento di un buon classificatore è necessario quindi avere un sufficiente numero di sottoinsiemi (in questo caso di esempi). Con un elevato numero di sottoinsiemi si ha:

- Bias dell'errore piccolo $\rightarrow$ il classificatore è *accurato*.
- La Variance dell'errore è grande.
- Elevato tempo richiesto alla computazione.

Al contraio un ristretto numero di sottoinsiemi comporta:

- Bias dell'errore grande.
- Variance dell'errore piccola.
- Basso tempo richiesto alla computazione.

In pratica si può dire che la scelta del numero di suddivisioni da applicare al set di dati a disposizione debba andare fatta in relazione alla grandezza del set di dati. Per il K-Fold Cross Validation è sufficiente la scelta di K=10.

2.3.3 Implementazione numerica della Cross Validation

L'implementazione algoritmica dei metodi di Cross Validation necessita l'utilizzo di generatori di numeri random per l'estrazione degli indici ed una formula di controllo

per la verifica delle possibili ripetizioni. Per definizione, infatti, al termine di tutte le estrazioni dovranno esser stati considerati tutti i dati in possesso (training set). Per la generalizzazione del metodo e per consentire un campionamento statistico è utile sfruttare estrazioni random, in modo che eseguendo più *run* del codice si possa ottenere una distribuzione degli output e, quindi, un risultato più robusto.

Come anticipato per l'implementazione del classficatore, in questo lavoro sono presi in considerazione solamente i linguaggi Pyhton e C++. Mentre per il primo viene già proposta nella libreria Sklearn [25] un efficiente algoritmo per l'implementazione sia del metodo Leave-One-Out che per il K-Fold, in linguaggio C++ è necessaria un'implementazione manuale. Quest'ultima può essere eseguita in differenti modi, tra cui quello implementato in questo lavoro di tesi ed esposto nel seguito. Per completezza precisiamo che la versione di Python del metodo necessita, appunto, l'inclusione della libreria Sklearn con riferimento alla sezione di Cross Validation (from sklearn import cross_validation). Un esempio di implementazione del metodo K-Fold è il seguente:

```
from sklearn.cross_validation import Kfold
def Kfold(Data, Label, Nfold, Nit):
    Ndati, Nsample = Data.shape
    for i_p in range(Nit):
        CV = KFold(len(Label), Nfold)
        for train_index, test_index in CV:
            Test = Data[:, test_index].T
            Training = Data[:, train_index].T
            Label_training = Label[train_index]
```

Per la versione in C++ è utile sfruttare un vettore di indici fisso a cui fare riferimento (idx_temp) e due matrici vuote per il salvataggio degli indici di test e training (idx_test e idx_train). Volendo estrazioni random, ma singole per ogni indice, è sufficiente applicare un algoritmo di Shuffle al vettore idx_temp e per ogni fold estrarre k indici che verranno inseriti in idx_test. Per il riempimento del vettore di training, invece, sarà sufficiente un ciclo for sugli indici ordinati, con un controllo per gli indici già estratti[8]. Poiché la realizzazione del metodo risulta banale se non per un'attenzione particolare agli indici, l'unica cosa che occorre precisare è la funzione di Shuffle utilizzata. Di seguito è riportato un esempio di questa tipologia di funzione, la quale è stata utilizzata all'interno dell'algoritmo QDANet PRO (ref. cap. 3 per la spiegazione completa dell'algoritmo) sviluppato in questo lavoro di tesi.

[8]NOTA: in questo passaggio è necessario tenere sott'occhio le variabili di ciclo e le posizioni in cui inserire gli elementi per non andare in overflow. Al proposito è utile introdurre una variabile ausiliaria che sia associata alla posizione di inserimento e incrementi una volta effettuato.

```
void shuffleArray(int* array, int size)
{
  int n = size;
  while(n > 1)
  {
    int k = rand()%n;
    n–;
    int temp = array[n];
    array[n] = array[k];
    array[k] = temp;
  }
}
```

Il metodo Leave-One-Out, in quanto caso estremo del K-Fold, risulta nettamente più facile da implementare poiché la scelta degli indici viene eseguita in modo ricorsivo, senza necessità di estrazioni random o particolari accortezze nell'indicizzazione.

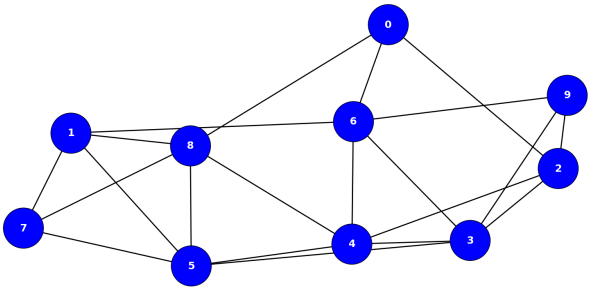
2.4 Complex Networks

Negli ultimi anni, lo studio delle interazioni in sistemi *large-scale* ha ricevuto un grande interesse e sviluppo, dovuto alla crescente disponibilità di grandi datasets e all'aumento della potenza di calcolo, sia per lo storage che per la manipolazione dei dati. A partire dalla mappatura del World Wide Web, gradualmente sono seguiti altri importanti progetti che hanno portato alla creazione di modelli di interazione appartenenti ai più svariati campi applicativi, partendo dalle scienze sociali, passando per l'economia ed il business, fino ad arrivare alla biologia. Nelle ricerche volte all'estrapolazione delle leggi che governano le dinamiche e l'evoluzione di questi sistemi complessi, i ricercatori hanno iniziato un'analisi sistematica seguita dalla caratterizzazione di queste rappresentazioni a network. Un importante risultato è che reti di grandi dimensioni sono generalmente caratterizzate da topologie complesse e strutture molto eterogenee. Queste caratteristiche possono essere quantificate analizzando i modelli statistici di connettività, studiando correlazioni non banali come la presenza di strutture a cluster o ricavando un'ordinamento gerarchico degli elementi della rete.
In totale generalità, un *network* può essere descritto da un grafo i cui *nodi* (vertici) identificano gli elementi del sistema e l'insieme di *links* di connessione (edges) rappresentano le relazioni di interazione tra questi elementi. Chiaramente un tale livello di astrazione è applicabile ad una vasta gamma di sistemi. In questo senso, quindi, i networks forniscono uno strumento teorico che permette una rappresentazione concettuale conveniente di interrelazioni in sistemi complessi, dove la caratterizzazione del sistema implica la mappatura di interazioni tra un gran numero di individui.
Lo studio dei networks deve il suo progresso teorico grazie allo sviluppo di branche come la sociologia, il calcolo discreto, la ricerca in comunicazione e solo recentemente

si è infiltrato anche nei campi fisici e biologici. Al di là di quella che può essere la nomenclatura utilizzata nei vari settori, le fondamenta analitiche rigorose sono legate alla *Teoria dei Grafi*. La nascita della Teoria dei Grafi - un vasto campo di studio della matematica teorica - può essere ricondotta agli studi pionieristici di Leonhard Euler con il suo lavoro sul problema dei ponti di Konigsberg (Euler, 1736).

Matematicamente, un *grafo non orientato* G è definito come una coppia di insiemi $G = (V, \varepsilon)$, dove V è un set di elementi numerabili non vuoto, detti vertici o nodi, e ε un set di coppie non ordinate di vertici differenti, detti edges o links. L'edge (i, j) unisce i vertici i e j, che sono detti per questo *adiacenti* o *connessi*. Il numero totale di vertici del grafo (ovvero la cardinalità dell'insieme V) è denotata con N e definisce l'ordine del grafo. La dimensione del grafo è invece data dal numero m dei suoi edges. Poichè in questo lavoro di tesi verrà considerata solo questa tipologia di grafo non faremo riferimento ad altri.

Per un grafo di dimensione N, il numero totale di edges è dato dal binomiale $\begin{pmatrix} N \\ 2 \end{pmatrix}$. Un grafo con $\varepsilon = \begin{pmatrix} N \\ 2 \end{pmatrix}$, i.e. con tutte le possibili coppie di vertici connesse da links, è detto *completamente connesso*.



Esempio di Matrice di Adiacenza con corrispettiva rappresentazione a network.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Poichè fondato essenzialmente sul calcolo algebrico, lo studio dei networks può essere affrontato sfruttando la notazione matriciale. Considerando ogni riga come un nodo del grafo, la matrice equivalente sarà necessariamente quadrata e simmetrica (links non orientati). Tale matrice $N \times N$ è detta *matrice di adiacenza*: l'elemento A_{ij} assumerà valore non nullo qualora esista il link tra i vertici i e j. Nel presente lavoro si opererà sempre con grafi non pesati, ovvero considerando un valore unitario per la presenza del link. In generale, invece, è possibilie associare un peso all'interazione tra i nodi ponendo un qualsiasi valore positivo w_{ij} a rappresentazione del link.

Considerando due grafi $G(V, \varepsilon)$ e $G'(V', \varepsilon')$, possiamo definire un nuovo grafo dato da $G \cap G'$ i cui vertici sono dati dall'insieme di intersezione $V \cap V'$ e gli edges da $\varepsilon \cap \varepsilon'$. Se $V \cap V' = \emptyset$ i due grafi sono detti *disgiunti*. Al contrario, se $V' \subseteq V$ e $\varepsilon' \subseteq \varepsilon$, allora $G'(V', \varepsilon')$ è un *sottografo indotto* di $G(V, \varepsilon)$.

Una problematica importante legata alla struttura dei grafi è data dalla raggiungibilità dei vertici, i.e. la possibilità di andare da un vertice ad un altro seguendo le connessioni date dagli edges del network. In un grafo completamente connesso ogni vertice è raggiungibile da un qualsiasi altro vertice. Le *componenti connesse* del grafo, quindi, definiscono le proprietà della sua struttura fisica. Al fine di analizzare le propietà di connettività, si definisca un *path* P_{i_0,i_n} in un grafo $G(V,\varepsilon)$ come una collezione ordinata di $n+1$ vertici $V_P = \{i_0, i_1, \ldots, i_n\}$ ed n edges $\varepsilon_P = \{(i_0,i_1),(i_1,i_2),\ldots,(i_{n-1},i_n)\}$, tali che $i_\alpha \in V$ e $(i_{\alpha-1},i_\alpha) \in \varepsilon$, per tutti gli α. La lunghezza del path P_{i_0,i_n} che connette i vertici (i_0,i_n) è n. Il numero $\mathcal{N}_{ij}$ di paths di lunghezza n tra due nodi i e j è dato dall'elemento ij della potenza n-esima della matrice di adiacenza del grafo. Un grafo è detto *connesso* se esiste un path che connette ogni coppia di vertici. Una *componente* $\mathcal{C}$ del grafo è definita come un sottografo connesso. Due componenti $\mathcal{C}_1 = (V_1,\varepsilon_1)$ e $\mathcal{C}_2 = (V_2,\varepsilon_2)$ sono disconnesse se è impossibile costruire un path P_{ij} con $i \in V_1$ e $j \in V_2$. Il concetto di path comporta l'introduzione nella teoria di una metrica che valuti la distanza tra i vertici. In particolare, alla luce delle precedenti definizioni, viene naturale definire una distanza tra due vertici i e j data dal numero di edges attraversati mediante i più piccoli path di connessione tra i due. Questa distanza, equivalente alla distanza utilizzata in chimica (*percolation theory*, Bunde and Havlin, 1991), è detta *lunghezza shortest path* ed è denotata con ℓ_{ij}. Quando due vertici appartengono a due componenti disconnesse del grafo, definiamo $\ell_{ij} = \infty$. Utilizzando questa metrica possiamo definire il *diametro* del grafo come $d_G = \max_{i,j} \ell_{ij}$.

2.4.1 Misure di centralità

Spesso il gran numero di elementi considerati negli studi di complex networks comporta l'utilizzo di analisi statistiche per consentire una caratterizzazione del grafo. In grandi sistemi, infatti, non è possibile individuare regolarità asintotiche cercando elementi e proprietà locali. Questo ha portato la ricerca, soprattutto nel campo della fisica e dell'informatica, a spostare l'attenzione su metodi di descrizione statistica su larga scala. In questo modo è possibile tener conto delle caratteristiche di aggregazione delle unità interagenti.
Durante lo studio di un network una delle più importanti informazioni che è necessario estrarre è l'importanza dei suoi elementi (Freeman, 1977). L'importanza di nodi o edges è definita comunemente attraverso la loro *centralità*. Esistono differenti misure per la rappresentazione della centralità di un nodo nel network: la significatività di un nodo viene valutata a seconda del valore della misura all'interno della distribuzione dedotta sull'intero grafo. Di seguito è riportata una breve descrizione di alcune di esse, utili per la comprensione dell'algoritmo proposto in questo lavoro.

Degree Centrality

La misura *degree* k_i associata al vertice i del grafo è definita come il numero di edges che si diramano dal vertice. Si noti che nel caso di grafo non orientato il numero di edges

entranti nel vertice coinciderà con il numero di edges uscenti, comportando l'equivalenza tra le quantità di *in-degree* $k_{in,i}$ e *out-degree* $k_{out,i}$. Considerando la struttura della matrice di adiacenza, il calcolo di queste due quantità risulta banalmente dato da

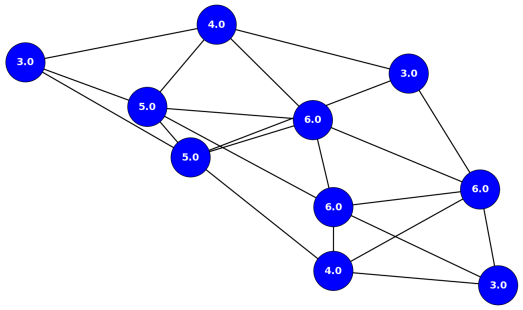


Figura 2.9: Esempio di Degree Centrality stimata per i nodi di un piccolo grafo.

$$k_{in,i} = \sum_j A_{ji} \qquad k_{out,i} = \sum_j A_{ij}$$

Il significato della quantità di degree centrailty è a questo punto evidente: quantitativamente definisce quanto bene un elemento del grafo risulti essere connesso con gli altri elementi. Computazionalmente la sua stima è banale e rapida.
Considerando la totalità dei vertici del grafo si otterrà una distribuzione relativa di degree $P(k)$, definita come la probabilità che una qualsiasi scelta random del vertice abbia degree k. La forma di tale distribuzione permette la classificazione dei networks reali. In genere si identificano due casi nettamente differenti e contrapposti di distribuzione: il primo riferito a grafi, cosiddetti, *omogenei*, con una forma simile ad una distribuzione di Poisson, ovvero con code sottili; il secondo riferito a grafi *eterogenei*, con distribuzioni a grosse code (es. power law), tipica dei grafi *scale-free*[9].

Betweenness Centrality

Mentre la precedente misura permette di stilare un ranking dei nodi in relazione alla loro condizione topologica di connessione, la misura di *Betweenness centrality* (BC nel seguito) tiene in considerazione l'importanza dei nodi nel connettere differenti regioni. Concettualmente introdotta da Freeman e Newman, la BC è definita dal numero di *shortest path* che connettono coppie di vertici, passando attraverso il vertice dato. Analiticamente, identificando con σ_{hj} il numero totale di *shortest path* dal vertice h a j e con $\sigma_{hj}(i)$ il numero di quelli che verificano il passaggio per il vertice i, la BC del nodo i è definita come

[9]Grafi con un'intrinseca invarianza di scala: la relazione tra il numero di nodi ed il numero degli edges associati correla in modo esponenziale negativo. In questo senso, quindi, la distribuzione della degree centrality risulta essere inevitabilmente una legge power law.

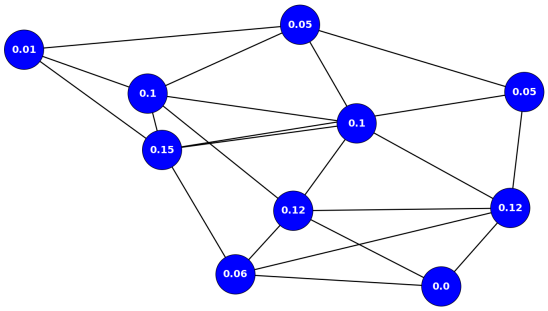


Figura 2.10: Esempio di Betweenness Centrality stimata per i nodi di un piccolo grafo.

$$BC_i = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}}$$

Secondo questa definizione, è evidente che i nodi centrali del grafo compariranno nella maggior parte degli *shortest path* a discapito dei nodi *pendenti*[10]. Nodi ad elevata BC comportano inevitabilmente maggiore importanza all'interno del network ed una loro rimozione spesso comporta la rottura in più componenti sconnesse. A livello computazionale la BC risulta non banale e altamente dispendiosa.

Clustering Coefficient

Il concetto di *clustering* per l'analisi dei networks, anche chiamato *transitività* nel contesto sociologico, si riferisce alla tendenza, osservata in molti grafi reali, a formare *cliques* nell'intorno di particolari vertici. Con il termine *clique* si definisce un sottografo completo $K_{i,j}$ di dimensione n, con $n < N$: la definizione porta ognuno dei nodi i all'interno del sottografo ad avere un edge di connessione verso ogni altro nodo j.
In questo senso, un alto *coefficiente di clustering* (C, nel seguito) implica che, se il vertice i è connesso con j e j è connesso con l, allora molto probabilmente i sarà connesso con l (rappresentazione matematica del concetto sociologico "gli amici dei miei amici sono anche amici miei"). Consideriamo allora un vertice i con degree k_i e denotiamo con e_i il numero di edges che compaiono tra i k_i vicini di i. Il coefficiente di clustering C_i di i è definito come il rapporto tra l'attuale numero di edges e_i ed il massimo valore possibile $k_i(k_i - 1)/2$, i.e.

$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

Statisticamente C_i misura la probabilità media che due nodi vicini del vertice i siano connessi anche tra loro (Watts and Strogatz, 1998). Si noti che la formulazione di C_i risulta consistente solamente per nodi con almeno $k_i > 1$, mentre per $k_i \leq 1$ si definisce $C_i = 0$.

[10] Nodi il cui degree entrante ed uscente è pari ad 1: graficamente consistono proprio nei nodi connessi al resto del grafo con un solo link e per questo definiti pendenti.

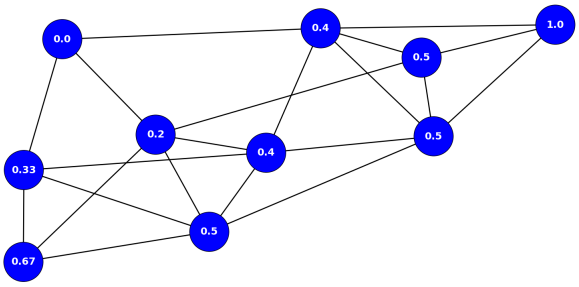


Figura 2.11: Esempio di Clustering Coefficient stimato per i nodi di un piccolo grafo.

Dalla definizione di e_i è facile verificare che il numero di edges nell'intorno di i può essere stimato nei termini della matrice di adiacenza come

$$e_i = \frac{1}{2}\sum_{il} A_{ij}A_{jl}A_{li}$$

Una definizione alternativa ma equivalente alla precedente è data dall'analisi della funzione di triplette[11] nel network

$$C_\Delta = \frac{3 \times \text{numero di triplette completamente connesse}}{\text{numero di triplette}}$$

dove il fattore 3 è legato al fatto che ad ogni triangolo sono associati tre nodi. Questa definizione, introdotta da Wasserman e Faust (1994) viene spesso utilizzata in ambito sociologico.

2.4.2 Implementazione numerica delle misure di centralità

Seppur estremamente performante, la modellizzazione dei Biological Big Data mediante networks può risultare computazionalmente dispendiosa. Per la realizzazione di un network di dimensione n, il numero di possibili relazioni è di ordine $O(n^2)$, mentre la ricerca di cluster o sottografi densi è di ordine $O(2^n)$. Per comprendere le tempistiche di calcolo ci si rifaccia alla Figura 2.12, in cui viene considerato l'impatto di 1000 processori per creare ed analizzare un network di correlazione utilizzato per modellizzare il tessuto cerebrale durante l'invecchiamento. Procedendo con un elevato livello di parallellizzazione associato alle operazioni da compiere sul grafo è possibile ottenere un ingente *speedup* aumentando il numero di processori: in questo caso da 2 settimane a pochi minuti. Come esposto nel capitolo precedente, la necessità di utilizzare HPC o tecnologie ad alto livello computazionale risulta indispensabile.
Le ricerche del settore Biomedico hanno mostrato che spesso i networks biologici a livello topologico presentano una distribuzione power law della connettività, sintomo di una forte asimmetria nel numero di link associati ai nodi. Anche per questo motivo è

[11] Una tripletta è definita come un sottografo contenente esattamente tre nodi.

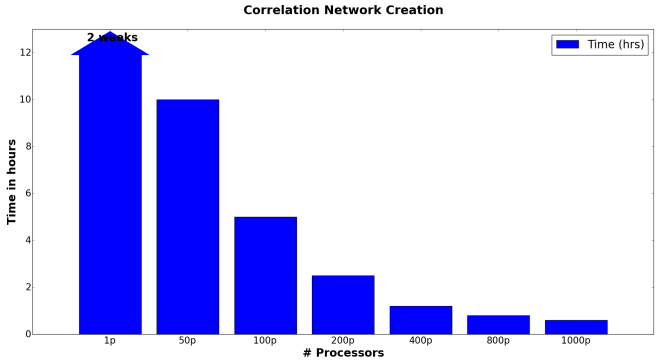


Figura 2.12: Impatto dell'utilizzo di 1000 processori per la creazione ed analisi di un Correlation Network per la modellizzazione di tessuti cerebrali in studi sull'invecchiamento.

sufficiente ed incentivato l'utilizzo di matrici sparse per la rappresentazione dei grafi, in modo da ottimizzare la quantità di memoria.

Per quanto riguarda l'implementazione vera e propria degli algoritmi, come già visto per i metodi di Machine Learning, sono già poste a disposizione numerose librerie a seconda del linguaggio scelto. Per la programmazione in Python citiamo la libreria *NetworkX* [24], mentre per un'implementazione a più alta efficienza in C++ la libreria *Boost Graph Library* mette a disposizione algoritmi già parallelizzati per l'analisi dei grafi.

Python - NetworkX

NetworkX[12] è una libreria del linguaggio Python per la creazione, manipolazione e studio delle strutture, dinamiche e funzioni delle complex networks. Al suo interno sono presenti numerosi algoritmi per la generazione di differenti grafi e funzioni per l'estrapolazione di informazioni da essi.

Di interesse per lo sviluppo dell'algoritmo QDANet PRO (cfr. Cap. 3 per la spiegazione completa dell'algoritmo) sono i metodi per la determinazione delle componenti connesse del grafo, la stima delle misure di centralità e la rimozione dei nodi pendenti. Mentre per le prime due classi è possibile sfruttare funzioni già implementate, per l'ultima è stato necessario sviluppare una funzione a parte.

Partendo dalla determinazione delle componenti connesse, una volta creato il grafo di studio[13] la funzione nx.connected_components restituisce un *dizionario* delle componenti connesse. Per una pratica visualizzazione e manipolazione di esse è utile eseguire un ulteriore passaggio, come segue

[12] É pratica comune importare la libreria NetworkX con l'abbreviazione di scrittura nx (import networkx as nx).

[13] Per l'utilizzo di NetworkX è indispensabile che il grafo sia rappresentato da un *dizionario* in cui le *keys* identificano i nodi ed i *values* i link presenti (1) o assenti(0) tra essi.

```
def Connected_Components(Adj_matrix):
    Componenti = list(nx.connected_components(nx.Graph(Adj_matrix))
    Comp = {}
    for i in range(len(Componenti)):
        for j in range(len(Componenti[i])):
            Comp[Componenti[i][j]] = i
    Comp = np.asarray(Comp.values())
    from collections import Counter
    Size = Counter()
    for val in Comp:
        Size[val] += 1
    Size = np.asarray(Size.values())
    print 'Network: %d nodes %d components'%(len(Adj_matrix), max(Comp)+1)
    return Comp, Size
```

Attraverso la precedente funzione è ottenibile un vettore i cui indici identificano le componenti connesse e gli elementi i nodi ad esse appartenenti. Inoltre per ognuna viene calcolata anche la dimensione riportata nel vettore Size. In questo modo l'output della funzione è confrontabile con quello ottenuto dalla funzione presente nella Boost Graph Library e a cui fanno riferimento linguaggi come C++ e Matlab.
Passando invece alle misure di centralità, mentre il calcolo della Degree risulta banale sommando righe (entrante) o colonne (uscente) della matrice di adiacenza[14], l'implementazione della BC e di C sono già implementate in NetworkX. Per praticità sono state sviluppate le seguenti funzioni:

```
def Betweenness_Centrality(Adj):
    import networkx as nx
    G = nx.Graph(Adj)
    BC = nx.betweenness_centrality(G)
    BC = np.asarray(BC.values())
    return BC
```

```
def Clustering_Coefficient(Adj, node = None):
    import networkx as nx
    G = nx.Graph(Adj)
    C = nx.clustering(G, node)
    C = np.asarray(C.values())
    return C
```

Entrambe le funzioni restituiscono due array indicizzati come i nodi del grafo e i cui elementi corrispondono ai valori delle misure di centralità ricercate.

[14] In Python la sommatoria di righe o colonne è ottenibile scrivendo semplicemente K = [sum(Adj_matrix[:,i] for i in range(len(Adj_matrix))] (per le colonne).

Per la rimozione dei nodi pendenti del grafo è stata scritta la funzione Pendrem. Secondo la definizione di nodo pendente, l'algoritmo procede con un ciclo while che termina nel momento in cui il minor valore di connettività presente nel grafo è diverso da 1. In modo ricorsivo, quindi, vengono rimossi tutti gli strati di nodi pendenti. Per tenere in considerazione anche il caso di grafo-a-catena, i.e. grafo privo di connessioni trasversali tra nodi, è introdotta una variabile di break (brk) che arresta il ciclo. Il codice proposto è quindi il seguente

```
def Pendrem(Adj):
    AdjRem = Adj
    RemInd = np.arange(0, len(AdjRem))
    print '%d nodes ...' %(len(RemInd))
    AdjK = np.asarray([sum(AdjRem[:, i]) for i in range(len(AdjRem))])
    brk = 0
    while min(AdjK) == 1:
        PenInd = np.where(AdjK == 1)
        PenInd = PenInd[0]
        adjrem = np.delete(AdjRem, PenInd, 0)
        AdjRem = np.delete(adjrem, PenInd, 1)
        RemInd = np.delete(RemInd, PenInd)
        AdjK = np.asarray([sum(AdjRem[:, i]) for i in range(len(AdjRem))])
        if len(AdjK) == 0:
            print 'Il network era una catena'
            brk = 1
            return AdjRem, RemInd, brk
    print '%d non-pendant nodes'%(len(RemInd))
    return AdjRem, RemInd, brk
```

C++ - Boost Graph Library

La Boost Graph Library (BGL), formalmente *Generic Graph Component Library*, è una libreria ad alte performances per la creazione e manipolazione dei grafi che fa parte della collezione delle librerie Boost. Queste librerie sono una collezione di codici open-source sviluppati in linguaggio C++ che hanno guidato l'evoluzione dello sviluppo delle librerie nella C++ community e nell'ANSI/ISO C++ standard.
Implementate anche per il linguaggio Matlab, costituiscono lo stato dell'arte per gli studi numerici sui networks. Attraverso la gestione di classi template, inoltre, permettono una totale generalità di applicazione e velocità di esecuzione.
L'utilizzo di questa libreria ha permesso la traduzione del codice-prototipo scritto in Python (vedi sopra) in linguaggio C++. In questo modo è stato possibile inserire nell'algoritmo QDANet PRO la componente di analisi a network. Per simmetria a quanto esposto precedentemente per il linguaggio Python, sono state sviluppate apposite funzioni per la determinazione delle componenti connesse, la stima delle misure di centralità

e la rimozione dei nodi pendenti. Come in precedenza, mentre le prime due classi sono implementabili mediante funzioni già presenti all'interno della libreria, la Pendrem è stato necessario scriverla totalmente.
In Appendice C sono riportati i codici sviluppati per le varie funzioni.

Capitolo 3

Algoritmo QDANet PRO

L'algoritmo QDANet PRO (*Quadratic Discriminant Anlisys with Network PROcessing*) convoglia insieme gli obiettivi di classificazione, dimensionality reducion e rappresentazione grafica in un unico metodo. Ideato dal gruppo di ricerca in biofisica dell'Università di Fisica e Astronomia di Bologna, guidato dal Prof. Remondini, il metodo è già stato testato su alcuni databases di piccole dimensioni con ottimi risultati. Il presente lavoro di tesi si è dedicato al completamento della sua implementazione e alla verifica delle sue performances su grandi datasets biologici.

Con il progredire degli studi sulla *gene expression* è risultato evidente che i livelli di espressione non sono affatto omogenei e deterministici per tutti i geni. Ogni gene tende ad esprimersi in quantità diversa e con valori distribuiti (rumore biologico). Spesso, inoltre, accade che siano proprio i geni con il più basso livello di espressione ed afflitti dalle maggiori fluttuazioni ad essere quelli di interesse per gli studi o comunque quelli di cui tutt'ora si ha scarsa conoscenza.

Uno degli obiettivi di analisi per la Biological Big Data Analytics è la classificazione dei dati: spesso gli studi vengono fatti contrapponendo i livelli di espressione di due o più classi di individui, in modo da poterne studiare le differenze (es. malati/sani, vecchi/giovani, prima/dopo il trattamento). Scopo di questi studi è appunto la ricerca di una correlazione tra l'appartenza ad una classe ed un incremento di espressione di uno o più geni.

Grazie alle tecniche NGS, però, tale confronto arriva a trattare numeri molto elevati di variabili, di cui solo una limitata porzione risulteranno possedere le informazioni rilevanti. L'analisi dei dati deve, quindi, non solo fornire il sottoinsieme di variabili a minor dimensione possibile, ma anche selezionare campioni che possano essere giustificati sul piano biologico: quello che si viene a creare è, quindi, la contrapposizione tra risultato biologico e risultato statistico.

Pur essendoci numerosi metodi per la riduzione della dimensionalità dei campioni, non sempre è facile affiancare ai risultati ottenuti una facile interpretazione biologica. Considerando un classificatore non lineare e, quindi, una superficie n-dimensionale di separazione per i dati, è possibile ottenere ottime performances di discriminazione, pagando

il prezzo di una difficile interpretazione dei risultati ed una poco robusta correlazione con la biologia.

Un'altra problematica da affrontare è legata alle possibili interazioni tra le variabili in gioco. In molte patologie, ad esempio, non vi è un solo gene responsabile o un sottogruppo disgiunto, ma bensì un sottogruppo a mutua interazione. Vista l'estrema complessità dei processi biologici una situazione di questo tipo è più che prevedibile.

Proprio rispettando il livello di complessità del campo applicativo, gli studi mediante complex networks trovano ampio utilizzo nella biologia. La generazione di un network di interazione tra geni, tuttavia, comporta la valutazione di tutte le possibili coppie di variabili ($O(N^2)$).

Il metodo QDANet PRO è stato ideato proprio per colmare queste difficoltà. Selezionando le migliori coppie (*best couples*) di probes in relazione al loro potere discriminante (primo step di riduzione), connette tra loro le coppie con membri comuni, fino a creare il network di interazione (*Discriminant Network* o D-Net nel seguito). La classificazione delle variabili viene eseguita attraverso un classificatore con la forma il più possibile semplice (lineare o al più quadratico), in modo da consentire una facile interpretazione biologica dei risultati. Una superficie di separazione lineare conduce a due casi possibili: uguale livello di espressione di entrambi i geni oppure la preponderanza di uno dei due sull'altro (cfr. Figura 3.3).

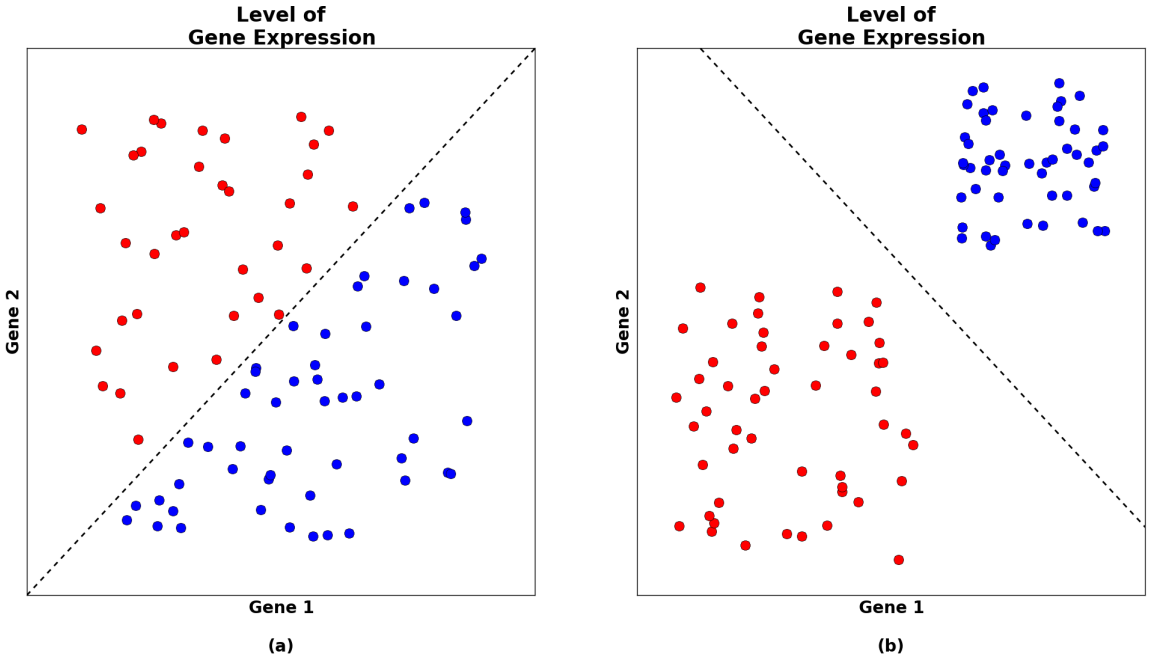


Figura 3.1: Modello ideale per l'espressione di una coppia di geni. In (a) è presentato il caso di uguale livello di espressione per entrambi i geni, da cui deriva una separabilità lineare mediante la bisettrice degli assi. In (b) è mostrato il caso di preponderanza nei livelli di espressione di un gene sull'altro, separabile da una retta a coefficiente angolare negativo. Per entrambi i casi (ideali) è evidente la semplicità di interpretazione biologica dei risultati.

Una volta ottenuto il D-Net delle best couples, questo viene suddiviso in sottografi connessi (secondo step di riduzione) ed ognuno di essi è studiato individualmente come potenziale Signature. L'ipotesi alla base di questo passaggio è che il massimo di informazione risieda in sottografi con alto tasso di mutua interazione e che, quindi, il loro studio individuale possa isolare i nodi a maggiore performance. La successiva analisi delle Signatures, mediante misure di centralità e riduzione del grafo (terzo step di riduzione) porta all'estrazione della *best Signature*.

Il metodo QDANet PRO definisce un semplice algoritmo, facilmente implementabile ed ad alta capacità di parallelizzazione. Ogni coppia viene processata separatamente e il codice può essere distribuito su più processori. Avendo in esame grandi volumi di dati questo fattore è al contempo indispensabile per la riuscita dell'algoritmo. L'output consisterà in un sottogruppo di probes ad alto potere discriminante, ridotta dimensionalità e facile interpretazione biologica. Come metodo offre anche numerose possibili estensioni: con un'adeguata potenza di calcolo potrebbero venir valutati anche gruppi più ampi delle singole coppie, a prezzo di una più difficile rappresentazione ed interpretazione della rete.

Nel complesso l'algoritmo può essere suddiviso in quattro macro-step: i primi due dati dalla valutazione delle coppie e dalla creazione del D-Net, ed i successivi inerenti l'analisi di questo e l'estrazione della best Signature. Nelle sezioni seguenti sono presentati nel dettaglio ognuna delle fasi di analisi.

3.1 Fase 1 : Valutazione delle coppie

Per prima cosa è necessaria la lettura completa del dataset. Non essendoci uno standard da rispettare per la costruzione dei files di dati inerenti la gene expression, è stato scelto un particolare formato per l'input da dare all'algoritmo. Ogni dataset in analisi è necessario uniformarlo a questo stile prima di ogni sottomissione al QDANet PRO. Nel dettaglio, è necessario che il file di testo sia in formato *.txt* e strutturato ponendo come prima riga le labels dei samples separate da $\backslash t$ e, successivamente, nella prima colonna tutte le sigle identificative delle probes seguite dai relativi campionamenti.

Successivamente vengono calcolate tutte le possibili combinazioni di coppie di geni. All'interno di un ciclo vengono valutati tutti i samples delle coppie attraverso una Leave-One-Out Cross Validation ed una classificazione diagQuadratic.

Questa parte dell'algoritmo è stata sviluppata sia in linguaggio Python (e Matlab) per la prototipizzazione che in linguaggio C++. Per una maggiore comprensione dell'algoritmo viene proposta la versione in Python del codice.

```
import itertools
Comb = list(itertools.combinations(np.arange(0, Ngeni), 2)
for i in range(Ngeni):
   Comb.append((i,i))
Comb = np.asarray(Comb)
Ncomb = len(Comb)
Good = np.zeros((Ncomb, 1))
GoodU = np.zeros((Ncomb, Nlabel_type))
for i_c in range(Ncomb):
   EstLabel = np.zeros(Nsample)
   for i_p in range(Nsample):
      Test = np.matrix(Data[Comb[i_c,:], i_p])
      Lbl_test = Label[i_p]
      Training = np.delete(Data[Comb[i_c],:], i_p, 1).T
      Lbl_training = np.delete(Label, i_p)
      EstLabel[i_p] = Classify(Training, Test, Lbl_training, 2)
   Good[i_c] = sum(Label==EstLabel)
   for ii in range(0, Nlabel_type):
      jj = np.where(Label==Label_type[ii])
      GoodU[i_c, ii] = sum(Label[jj]==EstLabel[jj])
```

Come si può notare la quantità di calcoli per ogni coppia di geni è alquanto ridotta e semplice. La reale difficoltà dell'elaborazione risiede nell'elevato numero di combinazioni che è necessario valutare. L'utilizzo di un classificatore *diagQuadratic* produce un minor costo a livello temporale evitando il calcolo dell'intera matrice inversa di covarianza, i cui elementi potrebbero dar luogo a singolarità.

Le versioni Python e Matlab dell'algoritmo possono essere utilizzate solamente su piccole quantità di dati, mentre per l'applicazione a databases reali è necessario sviluppare al meglio la componente di parallelizzazione del codice e l'ottimizzazione dell'algoritmo, possibile con il linguaggio C++. Al proposito si noti che l'intero algoritmo fa capo ad un unico ciclo *for* in cui vengono prese in esame separatamente le coppie di geni. L'indipendenza delle coppie tra loro consente un'elevata parallelizzazione dell'algoritmo. In questo lavoro si è scelto di utilizzare una parallelizzazione su più processori attraverso la libreria OpenMP. I programmi che utilizzano OpenMP sono compilati come programmi a *multithread*, in cui i *threads* condividono lo stesso spazio di memoria e, quindi, le comunicazioni tra *threads* possono essere molto efficienti. L'applicazione OpenMP è molto facile da utilizzare perché è il compilatore ad occuparsi della trasformazione da codice sequenziale a parallelo secondo le direttive. In questo modo possono essere scritti programmi *multithread* senza una totale comprensione del meccaniscmo *multithreading*. L'approccio utilizzato per la parallelizzazione è una suddivisione in blocchi per il passaggio tra sezioni sequenziali e parallele. All'ingresso del blocco parallelo, un singolo

thread di controllo viene suddiviso in più threads, per poi essere ricongiunti insieme alla fine del calcolo.

Ulteriori accortezze per il miglioramento dei tempi di calcolo possono essere prese per il calcolo dei coefficienti della matrice di covarianza. Per quanto visto nella funzione *Classify* (C++), gli elementi vengono stimati come $< x^2 > - < x >^2$, piuttosto che come scarto quadratico dalla media, ottimizzando il numero di calcoli, in quanto gli unici elementi necessari alla matrice sono quelli diagonali.

Al termine di questa sezione dell'algoritmo si otterrà una matrice di risultati di performances di classificazione di ogni coppia di geni. La performance viene valutata sia sulla singola classe che sulla totalità. Passaggio successivo sarà, quindi, il sorting delle performances per la creazione del ranking delle coppie e la creazione del grafo.

3.2 Fase 2 : Creazione del D-Net

Vista l'elevata quantità di dati sotto processo, l'ordinamento dei vettori delle performances risulta un passaggio costoso a livello temporale. L'utilizzo dell'algoritmo standard di sorting (funzione `sort()` del C++) è ristretto al solo vettore delle performances totali (corrispettivo di `Good` del codice precedente) mentre per le performances di singola classe è stato utilizzato un apposito algoritmo parallelizzato in OpenMP.

Per quanto riguarda l'algoritmo di sorting parallelo, il vettore di risultati viene suddiviso in n sottoparti uguali, con n pari al numero massimo di threads a disposizione. All'interno di ogni sottogruppo viene applicato il sorting standard, con un evidente miglioramento dell'efficienza ed una minor complessità. La parallelizzazione del sorting mediante multithreads costituisce un fattore critico per il miglioramento dei tempi di calcolo dell'algoritmo, grazie al quale si ottengono riduzioni di circa il 70% dei tempi di esecuzione.

Oltre alla totalità delle possibili coppie di interazioni (i, j), l'algoritmo prevede il calcolo delle performances anche delle singole probes, identificate come (i, i). Su di esse viene eseguito un sorting separato dalle restanti. In questo modo sarà possibile controllare la loro efficienza in classificazione in relazione alle coppie, stimando la locazione della loro miglior efficienza in relazione alla totalità.

L'intera Fase 1 e la parte iniziale di questo secondo step costituiscono le componenti computazionalmente più onerose del metodo QDANet e sono racchiuse in un unico codice eseguibile. Al termine di questi steps viene fornito in output un file di testo con le performances delle coppie ordinate: in incipit sono posizionate le coppie ordinate di singola probe seguite dalle coppie miste. La scrittura è fatta in linguaggio binario per l'ottimizzazione dello storage dei dati in un file denominato `binary_data`.

Nonostante l'estrema ottimizzazione dell'algoritmo finora presentato, per l'utilizzo è necessario disporre di componenti hardware sufficientemente potenti ed avanzate. Per questo motivo questa prima parte dell'algoritmo è stata eseguita sulle macchine HPC

disponibili al gruppo di ricerca in biofisica del Dipartimento di Fisica ed Astronomia dell'Università di Bologna.

Il presente lavoro di tesi si è rivolto principalmente all'implementazione ed ottimizzazione della parte successiva, avendo come punto di partenza il dataset ed il file contenente le performances ordinate. A questo punto, infatti, è necessario creare il network delle best-performances, applicando il primo step di dimensionality reduction. Poiché la quantità di dati da elaborare risulta nettamente inferiore alla precedente[1], viene meno la necessità di utilizzare algoritmi parallelizzati e, in alcuni casi, persino la versione del codice in C++.

La creazione del D-Net comporta il setting di alcuni parametri, i quali vengono decisi al momento dell'esecuzione del programma dall'utente. Una volta determinati gli istogrammi delle performances di classificazione (un istogramma per ogni classe in aggiunta a quello totale), all'utente è richiesto su quale ha intenzione di operare mediante la scelta di un valore numerico: per il caso di sole due classi le possibili opzioni sono 0=prima classe, 1=seconda classe e 2=totale. Una volta scelto il gruppo di performances è richiesto il valore di sogliatura da applicarvi, al fine di selezionare solamente le coppie di probes con efficienza superiore (o uguale). Fissati questi parametri il programma restituisce il numero di probes selezionate ed appartenenti alle varie coppie[2].

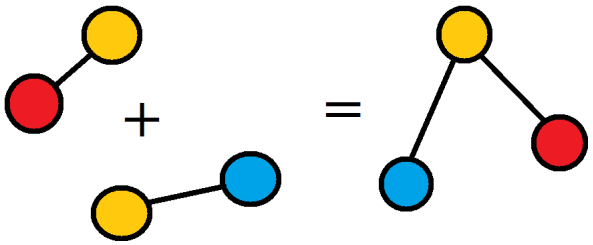


Figura 3.2: Unione delle best couples selezionate dal QDANet PRO. Avendo indicizzato le coppie sarà possibile unire tra loro quelle che presentano un gene in comune ed il cui valore di classificazione sia superiore alla soglia imposta. L'unione delle coppie andrà a costituire la Signature su cui poter proseguire l'analisi. Questo passaggio rappresenta il punto di forza del metodo QDANet PRO per la riduzione della dimensionalità.

Le probes così selezionate costituiranno i nodi del D-Net, i cui link vengono decisi dalle composizioni delle singole coppie selezionate. Il numero di variabili su cui verte ora l'analisi dovrebbe essere altamente ridotto rispetto al valore iniziale. La sogliatura dell'istogramma è fatta, in genere, considerando valori prossimi alla massima performance in modo da selezionare le effettive *best-probes*.

[1] Si noti che la quantità di calcoli da effettuare solamente per la valutazione delle medie dei samples corrisponde a (numero di colonne)×(numero di righe)^2/2 di tutto l'intero datasets, il quale può arrivare a contenere intorno ai 10^4 probes.

[2] Questa operazione è espressa dalla funzione unique dei linguaggi ad alto livello, mentre per la versione C++ può essere calcolata sfruttando i set-array delle STL.

3.3 Fase 3 : Riduzione del D-net

Da questo sottogruppo di probes interagenti secondo la topologia del D-Net è necessario un ulteriore step di riduzione. Questo secondo step di dimensionality reduction è ottenuto sfruttando metodi di Teoria dei Grafi.

Come visto in 2.4, a partire da un grafo ad elevato numero di nodi è possibile chiedersi se al suo interno vi siano componenti connesse. A livello computazionale la determinazione dei sottografi permette l'applicazione dei successivi algoritmi su matrici ridotte, con, quindi, un aumento dell'efficienza di calcolo. A livello interpretativo, invece, l'analisi delle componenti porterà all'identificazione della Signature cercata.

Estratte tutte le componenti connesse del D-Net è eseguita su ciascuna di esse una valutazione in performances di classificazione mediante classificatore *diagQuadratic*. Anche in questo caso, come per il primo step dell'algoritmo, i dati da valutare vengono suddivisi mediante una Leave-One-Out Cross Validation. Al termine sono riportate in output le performances di classificazione di ciascuna componente, relative a ciascuna classe ed alla totalità, con numero di probes associate.

In base ai risultati ottenuti, l'utente potrà selezionare la miglior componente connessa, scegliendo il compromesso ottimale tra dimensione ed efficienza di classificazione. Tali valori di performances rimangono comunque validi solamente in relazione ai dati di training. La possibilità di scelta della componente connessa su cui continuare ad operare e la precedente selezione della sogliatura vengono lasciati all'utente per un'applicazione del metodo QDANet PRO a qualsiasi tipologia di dato e necessità.

Qualora il D-Net fornisca componenti connesse ad elevato numero di nodi, il programma propone il salvataggio a parte (su file *.txt*) della matrice sparsa. L'implementazione di questa opzione è stata applicata al seguito di alcuni test su datasets analizzati in fase di sviluppo, i quali, disponendo di un numero ridotto di samples, producevano un elevato numero di coppie ad alta performance. A partire dalla matrice sparsa del D-Net è possibile proseguire l'analisi sfruttando misure di centralità[3].

[3]Il caso in questione fa riferimento ad uno studio effettuato in fase di sviluppo del software su un dataset di *gene expression* relativo a bovini e fornito da un'ente esterno. L'obiettivo verteva sull'estrazione di informazioni utili per la discriminazione tra campioni sani (NN), malati (PP) e con patologia in incubazione (NP). Di ciascuna classe ci sono stati forniti solamente 5 samples, rendendo impossibile uno studio statistico accurato. Si è deciso per questo di applicare il metodo QDANet PRO alla dicotomia tra NN e NP, ipotizzando la classe PP come caso estremo di quest'ultima. Il ridotto numero di samples, tuttavia, ha portato alla selezione della quasi totalità delle coppie di probes, rendendo impossibile l'analisi successiva della Signature e vanificando l'obiettivo della dimensionality reduction. Per questo è stata sviluppata un'apposita variante del metodo che utilizzasse l'intersezione di più misure di centralità. In particolare si è visto che selezionando i nodi con elevato coefficiente di clustering e connettività era possibile estrarre un sottografo di dimensione molto ridotta e a performance ottimale di classificazione. Applicando la validazione della Signature sui dati della classe PP le performances continuavano a rimanere molto alte ed ulteriori test hanno confermato la scelta del sottogruppo di probes. La Signature ha poi trovato interpretazione anche sul piano biologico, confermando le potenzialità dell'algoritmo QDANet PRO nell'analisi di dati bio-molecolari.

3.4 Fase 4 : Estrazione della best Signature

In quest'ultima fase occorre ridurre la componente connessa selezionata, fino all'ottenimento della Signature finale. Qualora la componente sia di dimensioni ridotte non sarà necessario ridurla ulteriormente, ma sarà sufficiente verificare a livello statistico la sua efficienza di classificazione. Per fare ciò è prevista la valutazione della Signature, ad ogni step di riduzione, mediante una K-Fold Cross Validation iterata un numero N di volte. Come per i casi precedenti sia il valore di K che quello di N sono scelti al momento dell'esecuzione dall'utente.

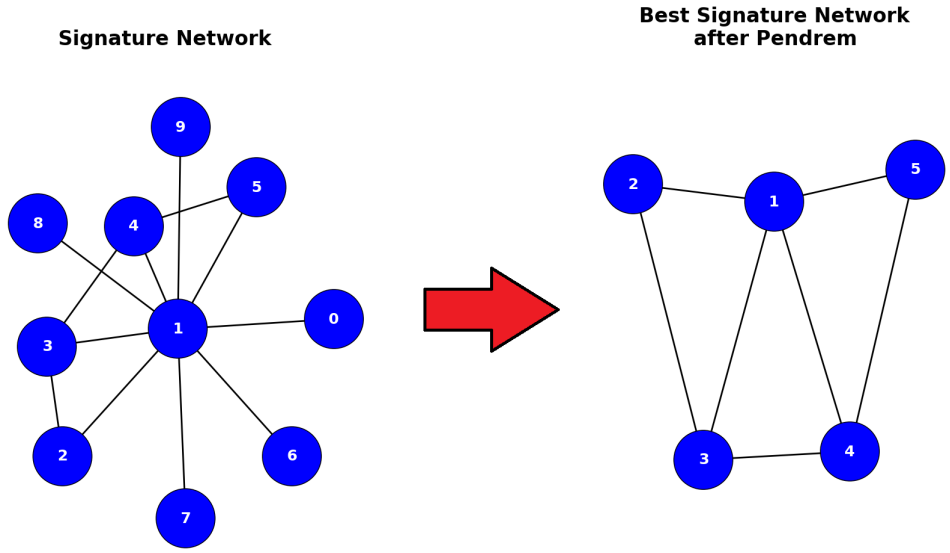


Figura 3.3: Estrazione della best Signature mediante rimozione dei nodi pendenti. Il passaggio è reso possibile dall'applicazione della funzione Pendrem (cfr. Sez. 2.4.2) alla matrice di adiacenza della Signature.

Per la riduzione della dimensionalità è possibile operare secondo tre modalità distinte: la rimozione di tutti i nodi a connettività pari ad 1, dei nodi pendenti (ottenibile con l'applicazione della funzione pendrem, cfr. 2.4.2) oppure dei nodi provenienti da una lista selezionata manualmente. Al seguito della riduzione la Signature estratta viene data in input al classificatore *diagQuadratic*.
Nella maggior parte dei casi la rimozione dei nodi pendenti è sufficiente per una buona riduzione del problema. Associata all'idea di nodo pendente, infatti, vi è l'interpretazione di una probe a minima interazione con le altre, fattore non ricercato all'interno di una buona Signature.
Come ulteriore tool di visualizzazione dei risultati ottenuti, il codice mette a disposizione il calcolo della PCA sulle probes selezionate, considerando come assi di proiezione le sole due componenti principali.
Al termine dell'esecuzione è possibile salvare la Signature ricavata su un file formato *.txt*. Nel file vengono salvate in un'unica colonna i nomi delle probes, in modo da essere utilizzati nella fase di validazione.

La validazione della Signature deve essere eseguita su dati non utilizzati durante l'addestramento e, quindi, lasciati da parte durante i calcoli finora riportati. Mediante i nomi delle probes selezionate verranno estratti dall'intero datasets le sole righe appartenenti a questi dati e, riaddestrando il classificatore sui samples utilizzati in precedenza, testata la classificazione dei nuovi dati con lo stesso classificatore precedente (*diagQuadratic Classifier*) o eventualmente con la versione più accurata (*Quadratic Classifier*).
Ad ulteriore conferma dei risultati ottenuti è estratto per M ($M \sim 10^4 - 10^5$) volte in modo random un numero di probes equivalente alla dimensione della Signature e valutata la classificazione. Questo consente di verificare o meno l'ipotesi sulla bontà della Signature e del metodo QDANet PRO, confrontando i risultati in relazione alla distribuzione di modello nullo. Dalla distribuzione delle performances è individuata la probabilità che un sottogruppo altrettanto numeroso di probes random abbia la medesima efficienza di quello selezionato, permettendo di dare consistenza ai risultati ottenuti.
Qualora si verifichi il caso di classi a numerosità particolarmente dis-omogenea è necessario apportare modifiche al metodo finora esposto. La ricerca della best Signature è stata effettuata finora valutando la sola performance totale di classificazione, la quale risulterebbe falsata nel caso di forte aderenza del classificatore ad una particolare classe. In questo caso, allora, è sufficiente effettuare la sogliatura in relazione alla performance totale pesata secondo la numerosità delle singole classi. Questa modifica si ripercuoterà anche nella verifica del modello nullo. Per questo motivo è utilizzato il *Coefficiente di Correlazione di Matthews* (MCC), una misura metrica quantitativa utilizzata nella classificazione binaria. Per definizione MCC $\in [-1, 1]$, dove con $+1$ si identifica una predizione perfetta, -1 una predizione totalmente errata e 0 indica una predizione analoga a quella ottenuta da un classificatore random. L'efficienza del Coefficiente di Matthews risiede nella sua formulazione

$$MCC = \frac{TP \times TN - TP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

dove identifichiamo con TP i *veri positivi*, TN i *veri negativi*, FP i *falsi positivi* e FN i *falsi negativi*. Ognuno di questi valori è ottenibile dalla matrice di confusione del classificatore, in cui vengono messe in relazione le classi predette con quelle osservate. É immediato notare, allora, come il valore di MCC sia più adeguato della semplice performance totale nella valutazione di una Signature proveniente da classi a numerosità non omogenea.

Capitolo 4

Applicazioni a Databases reali

Come datasets di verifica per il metodo QDANet PRO, sono stati utilizzati i dati provenienti dal TCGA (syn1710282,doi:10.7303/syn1710282), database già analizzato e pre-processato da Yuan et al. nel loro lavoro [1]. Il loro studio ha riguardato l'analisi di quattro tipologie di tumore mediante la classificazione di dati di varia tipologia. La loro ricerca ha dimostrato come l'integrazione di dati molecolari con dati clinici sia in grado di migliorare le performances di classificazione in maniera sensibile. A questo scopo nel lavoro sono stati utilizzati 8 differenti classificatori con kernel più o meno complessi, analizzando la totalità delle features a disposizione.

Dall'intero database sono state selezionate quattro tipologie di tumore: *kidney renal clear cell carcinoma* (KIRC), *glioblastoma multiforme* (GBM), *ovarian serous cystadenocarcinoa* (OV) e *lung squamous cell carcinoma* (LUSC). Queste tipologie sono state selezionate poiché all'interno del database sono presenti un sufficiente numero di campioni per considerazioni statistiche. Per ogni tumore sono stati utilizzati dati di 4-5 differenti tipologie, ognuna ricollegabile al livello di espressione genica ((*i*) SCNA (o CNV[1]): *Affimetrix Human SNP Array 6.0, ∼100 arm or focal alteration*; (*ii*) DNA methylation[2]: *Illumina DNA Methylation microarray, ∼20,000 genes*; (*iii*) mRNA expression: *Agilent 244 K microarray or Illumina mRNA-seq., ∼20,000 genes*; (*iv*) microRNA (miRNA) expression: *Agilent Human miRNA-specific microarray or Illumina miRNA-seq, >500 microRNAs*; (*v*) protein expression (o RPPA[3]): *reverse-phase pro-*

[1] Copy number variation (CNVs): è uno dei relativamente nuovi campi della genomica. Dagli studi genomici è emerso che sezioni del genoma tendono a ripetersi ed il numero di ripetizioni varia tra individui della stessa specie. Mentre fino a qualche tempo fa si riteneva che la variabilità del genoma fosse legata a singole basi (*Single Nucleotide Polymorphism*, SNP), si è recentemente dimostrato che la maggior fonte di variabilità risiede nella variazione del numero di copie (CNVs). Uno dei metodi di maggior successo per l'individuazione delle CNVs consiste nel conteggio del numero di sequenze (*Read Count*, RC) che vengono allineate a porzioni del genoma.

[2] Con metilazione del DNA identifichiamo una modifica epigenetica del DNA. Questo processo risulta strettamente correlato al silenziamento genico anomalo, tipico delle cellule tumorali. Anche per questa tipologia di dati è possibile ricondursi ad una tecnologia microarray.

[3] *Reverse phase protein array* (RPPA): è una matrice microarray progettata per la misurazione dei livelli di espressione di una proteina. La tecnologia di funzionamento è del tutto analoga a quella dei microarray per l'espressione genica. Non potendo sfruttare l'ibridizzazione, in questo caso il substrato

tein array, ∼170 proteins). I dati sono stati pre-processati, andando ad estrarre da ogni dataset un *core*. Per quanto riguarda i dati di miRNA e mRNA, il pre-procesing porta a considerare solamente geni di cui sono note le funzionalità biologiche (cfr. Tabella 4.1).

Cancer	Overall survival	SCNA	Methy	mRNA	miRNA	Protein	Core set
GBM		SNP_6	27k	AgilentG4502A	H-miRNA_8x15k	RPPA[a]	
	565	563	287	492	491	214	210
KIRC		SNP_6	450k	HiseV2	GA+Hiseq	RPPA	
	500	493	283	469	454	480	243
OV		SNP_6	27k	AgilentG4502A	H-miRNA_8x15k	RPPA	
	563	559	600	558	586	412	379
LUSC		SNP_6	450k[a]	HiseqV2	GA+Hiseq	RPPA	
	305	343	225	220	351	195	121

Tabella 4.1: Per ogni tipologia di tumore, la prima riga mostra la piattaforma e la seconda il numero di samples. SNP_6: Affymetrix Genome-Wide Human SNP Array 6.0; 27k: Illumina Infinium Human DNA Methylation 27K, 450k: Illumina Infinium Human DNA Methylation 450K; AgilentG4502A: Agilent 244K Custom Gene Expression G4502A; HiseqV2: Illumina HiSeq 2000 RNA Sequencing V2; H-miRNA_8x15K: Agilent 8x15K Human miRNA-specific microarray platform; GA+Hiseq: Illumina Genome Analyzer/HiSeq 2000 miRNA sequencing platform; RPPA: MD Anderson reverse phase protein array.
[a] Questo tipo di dato non è stato incluso per questo tumore.

Ogni dataset è stato suddiviso per 100 volte in training set e test set per avere sufficiente statistica. La dicotomia dei samples è stata effettuata considerando i seguenti cutoff:

- KIRC: 4 anni o 365 * 4 = 1460 giorni, numero di samples: 150;
- GBM: 1 anno o 365 giorni, numero di samples: 155,
- LUSC: 2 anni o 365 * 2 = 730 giorni, numero di samples: 77;
- OV: 3 anni o 365 * 3 = 1095 giorni, numero di samples: 252;

I samples con sopravvivenza censurata prima del cutoff sono stati esclusi. La dicotomia è ottenuta ponendo l'appartenenza alla classe 1 per i samples sopra la soglia (strettamente maggiore di) e con 0 per quelli al di sotto ($\leq$ cutoff). I risultati ottenuti dallo studio di Yuan et al. utilizzando l'intero ammontare di probes per la classificazione sono presentati in Figura 4.1, dove vengono indicati i valori di mediana delle distribuzioni delle performances di classificazione relative ai 100-fold.

è formato da anticorpi monoclonali o da peptidi di piccole, medie e grandi dimensioni.

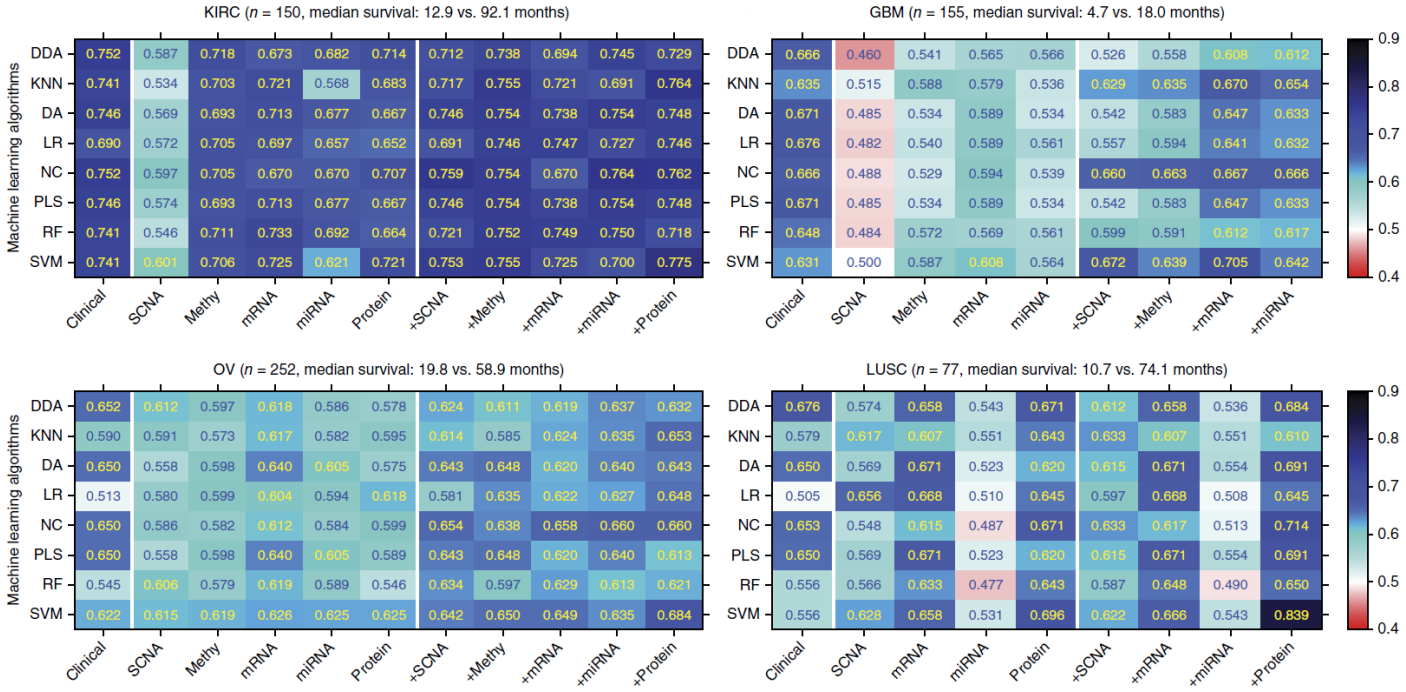


Figura 4.1: Performances di classificazione di 8 differenti classificatori utilizzati nello studio di Yuan et al. sull'intero set di probes a disposizione. L'analisi ha riguardato sia il potere predittivo delle singole tipologie di dati molecolari che la loro combinazione con dati clinici.

4.1 Applicazione del QDANet PRO

Alla luce di quanto emerso dallo studio di Yuan et al., l'algoritmo QDANet PRO si propone come metodo alternativo per l'analisi di dati molecolari. A partire dai *core-datasets* utilizzati per lo studio precedente, sono state utilizzate le medesime condizioni sperimentali per l'elaborazione dei dati, sia per quanto riguarda la dicotomizzazione dei samples, sia per la suddivisione dei dati in training set e test set. In questo modo sarà possibile un confronto diretto dei risultati ottenuti con le altre tipologie di classificatore. Il confronto dovrà essere eseguito non solo in termini di performances di classificazione, ma terrà conto della potenzialità del QDANet nel ridurre la dimensionalità del problema mediante l'estrazione di un sotto gruppo di probes.

L'algoritmo è stato ideato per l'analisi di livelli di espressione e per questa ragione l'elaborazione ha riguardato prevalentemente i dati di miRNA, mRNA e RPPA, tralasciando quelli di metilazione e verificando solamente l'applicabilità anche a quelli di CNV. Prima di procedere all'applicazione è necessario uniformare i datasets allo standard d'input del QDANet (cfr. Sez. 3.1). Avendo a disposizione 100 possibili sequenze di addestramento e relativi test, si è utilizzata la prima suddivisione proposta da Yuan et al. per l'estrazione della best Signature, riservando il test set corrispondente per la prima validazione. Una volta estratta, la best Signature è ri-addestrata anche sugli altri training set e validata sui corrispondenti test: in questo modo si otterrà una distribuzione di

performances dalla quale poter estrarre il valore di mediana per il confronto dei risultati. Per l'estrazione della best Signature è utilizzato un classificatore *diagQuadratic*, mentre per la validazione un classificatore *Quadratic*.
Per ogni tipologia di training set (cfr. Tabella 4.1) è applicata la prima parte dell'algoritmo per la valutazione delle coppie. Con l'intento di lavorare successivamente con le coppie a maggiore performance si è scelto di salvare il 10% delle coppie *top scorer* per i dati di mRNA, mentre per le restanti tipologie di dati (meno numerosi) l'intero insieme di possibili coppie. In questo modo si rende più agevole la manipolazione successiva dei dati.

		Probes	Sample	Lbl 0	Lbl 1	Couples
	miRNA	1045				546535
	CNV	69				2415
	RPPA	166				13861
KIRC	mRNA	20530	192	121	71	6075072
	miRNA+mRNA	21575				232751100
	miRNA+RPPA	1211				73388
	miRNA	798				318801
	CNV	109				5995
	RPPA	165				13695
OV	mRNA	17813	300	139	161	5866040
	miRNA+mRNA	18611				173193966
	miRNA+RPPA	963				46418
	miRNA	533				142311
	CNV	106				5671
GBM	RPPA	[a]	168	50	118	[a]
	mRNA	17813				5866040
	miRNA+mRNA	18346				168297031
	miRNA	1045				546535
	CNV	114				6555
	RPPA					15225
LUSC	mRNA	20530	96	60	36	6075072
	miRNA+mRNA	21575				232751100
	miRNA+RPPA	1219				74360

Tabella 4.2: Tabella riassuntiva dei dati analizzati: per ogni tumore sono presenti il numero di probes, il numero di samples, il numero di campioni appartenenti alla classe 0, il numero di campioni appartenenti alla classe 1 e il numero di coppie analizzate dall'algoritmo, per ciascuna tipologia di dato. Oltre alle tipologie singole è presente anche la combinazione di dati di (miRNA, mRNA) e (miRNA, RPPA).
[a] Tipologia di dato assente per questo tumore.

Per l'elaborazione delle coppie di miRNA, CNV e RPPA, visto il basso numero di probes,

è stato sufficiente l'utilizzo di un computer Laptop (processore Intel(R) Core(TM) i7-6500U CPU @ 2.60 GHz, RAM 8,00 GB) a 4 core: per ogni *run* del codice i tempi di calcolo non hanno mai superato il minuto ($\sim 10^5$ coppie). Per quanto riguarda i dati di mRNA, invece, è stato utilizzato l'HPC del gruppo di biofisica dell'Università di Bologna (Intel(R) Canoe Pass Tower, 2x750W, S2600CP4, 8 hot swap 3.5" 2; Intel Xeon E5 2650 eight core 2.00 GHz 20 MB 8.0GT/S; Kingston DDR3 ECC REG 16 GB 1600MHz, 192 Gb, Intel Sata SSD 250 GB 7200 rpm 64 MB; WD Sata 6 GB/S 2TB 7200 rpm 64 MB): in questo caso i tempi di calcolo sono rimasti comunque confinati intorno ai 5-6 minuti di elaborazione ($\sim 10^8$ coppie). In Tabella 4.13 sono riportate le informazioni inerenti la prima fase di elaborazione del QDANet PRO.

Nello studio di Yaun et al. si è considerata la combinazione di dati molecolari e clinici, escludendo una possibile combinazione dei primi. Prevedendo una migliore risposta dell'algoritmo ai dati di mRNA, seguiti dai miRNA ed RPPA, si è testata l'efficacia del QDANet PRO anche sulle combinazioni (miRNA, RPPA) e (miRNA, mRNA)[4]. L'obiettivo è quello di ottenere Signatures con combinazioni di queste tipologie di dati, valutando un'eventuale maggior risposta in classificazione ottenuta dall'unione di informazioni differenti.

La scelta della best Signature in fase di training è stata svolta selezionando ogni volta il sottogruppo di probes a maggior performance totale di classificazione, con al contempo un adeguato bilanciamento delle performances di singola classe. Ad ogni step di classificazione l'algoritmo fornisce i valori di mediana delle distribuzioni di performances ed è quindi preferibile avere tutti i valori superiori, o al più prossimi, ad almeno il 50%. Nel caso delle Signatures eterogenee, la selezione non ha riguardato solamente la miglior performance di classificazione, ma si è scelto di operare la sogliatura massimale per la comparsa di entrambe le tipologie di dato all'interno del network. I risultati ottenuti durante l'addestramento, per ogni dataset, sono riportati in Tabella 4.3.

I risultati dimostrano che l'informazione per una buona classificazione è ottenibile utilizzando solamente un numero ridotto di probes rispetto a quelle a disposizione. Questo comportamento è del tutto generale nell'analisi di Big Data, nei quali vi è inevitabilmente grande ridondanza delle informazioni ed un basso rapporto segnale rumore. Come atteso, la tipologia di dato che meglio si presta all'utilizzo del QDANet PRO risulta essere l'mRNA, seguita dai dati di miRNA e RPPA. Per ogni tumore le Signatures di mRNA consentono le migliori performances di classificazione, connesse alla maggiore riduzione della dimesionalità: questo risultato è strettamente correlato alla maggiore quantità di probes di mRNA rispetto alle altre tipologie. Le Signatures eterogenee, invece, sembrano non portare miglioramenti nelle performances, evidenziando invece una

[4] In fase di sviluppo è stata testata l'efficienza anche della combinazione (mRNA, RPPA). Questo caso ha messo ben in evidenza come la presenza dei dati di mRNA, o più in generale di una tipologia predominante sull'altra per la classificazione, comporti problematiche nell'estrazione di una Signature eterogenea: anche alzando i valori di sogliatura delle coppie le uniche combinazioni che si presentano sono composte da probes di mRNA. Per questa ragione questa tipologia di dato è stata rimossa dall'analisi.

		Signature Dimension	True Negative Class 0	True Positive Class 1	Total Correct Classification
KIRC	miRNA	140	85.9%	42.3%	69.8%
	CNV	48	65.3%	32.4%	53.1%
	RPPA	16	85.1%	59.2%	75.5%
	mRNA	113	88.4%	70.4%	81.78%
	miRNA+mRNA	7	53.7%	59.1%	55.7%
	miRNA+RPPA	23	88.4%	36.6%	69.3%
OV	miRNA	59	58.9%	67.7%	63.7%
	CNV	8	49.6%	69.6%	60.3%
	RPPA	2	51.8%	68.3%	60.7%
	mRNA	42	71.2%	69.5%	70.3%
	miRNA+mRNA	854	50.4%	49.7%	50.0%
	miRNA+RPPA	64	59.7%	69.6%	65.0%
GBM	miRNA	165	48.0%	71.2%	64.3%
	CNV	17	62%	52.5%	55.4%
	RPPA	[a]			
	mRNA	63	58%	88.9%	79.8%
	miRNA+mRNA	14	48.0%	68.6%	62.5%
LUSC	miRNA	365	73.3%	75.0%	73.9%
	CNV	4	55.0%	66.7%	59.4%
	RPPA	72	81.7%	66.7%	76.0%
	mRNA	8	76.7%	86.1%	80.2%
	miRNA+mRNA	21	43.3%	63.8%	51.0%
	miRNA+RPPA	84	93.3%	41.7%	73.9%

Tabella 4.3: Estrazione della best Signature: per ogni dataset è estratto il numero minimale di probes per la miglior classificazione attraverso un diagQuadratic Classifier. Per ogni Signature è riportato il numero di probes che la compone ed i valori percentuali di classificazioni corrette (*True Positive* e *True Negative*) rispetto alla numerosità della classe. La scelta della Signature è effettuata considerando un buon bilanciamento tra le performances di singola classe, anche a discapito della performance totale. A questo fine, in alcuni casi è utilizzata una sogliatura delle coppie in relazione alla performance totale pesata rispetto al numero delle componenti delle singole classi.

[a] Tipologia di dato assente per questo tumore.

perdita di informazione rispetto ai dati singoli. Unico caso di aumento di performances lo si ha per OV mediante la combinazione (miRNA, RPPA): entrambe le Signatures singole possiedono già una buona performance totale, la quale tende ad aumentare con la combinazione senza un aumento significativo delle dimensioni del network.

Passando alla validazione delle best Signatures così isolate è possibile stilare una distribuzione di performances totale di classificazione. Nella Figure 4.2 sono presentati i

box plot delle distribuzioni, andando a contraddistinguere il valore delle mediane (riga nera e valore numerico riportato in alto), le medie (stelle bianche) e i valori di mediana delle distribuzioni di classificazione di singola classe: in questo modo si può verificare che la Signature consenta una classificazione bilanciata. Per ogni tipologia di tumore, oltre ai risultati sui singoli dati molecolari sono presenti anche le distribuzioni delle combinazioni (miRNA, RPPA) e (miRNA, mRNA).

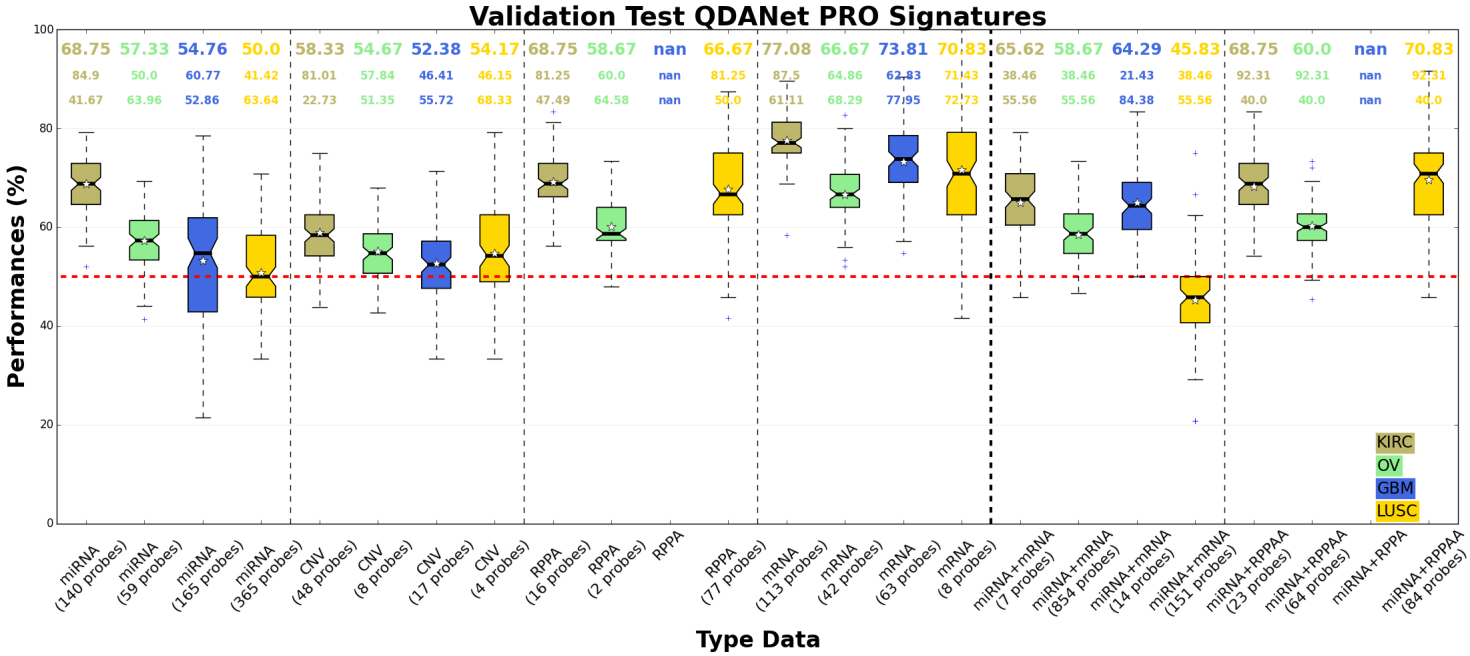


Figura 4.2: Validazione delle best Signatures: mediante le 100 combinazioni di training-test set è stata addestrata a validata, rispettivamente, la best Signature estratta dal QDANet. Il classificatore utilizzato è di tipo Quadratic. Per ogni Signature è riportato il numero di probes che la compone, il valore di mediana della distribuzione (linea nera) e media (stellina). Sopra sono riportati, dall'alto verso il basso, i valori numerici delle mediane delle distribuzioni di performances totale, della classe 0 e della classe 1, rispettivamente. Si noti come per ogni tumore la tipologia di dato che conduce ai migliori risultati di classificazione è l'mRNA. Al seguito della linea tratteggiata in nero (più spessa) sono riportati i risultati delle Signatures eterogenee, date dalle combinazioni di dati (miRNA, mRNA) e (miRNA, RPPA).

La validazione delle Signatures mette in evidenza l'alta efficacia delle Signatures di mRNA rispetto alle altre, le quali conducono, per ogni tumore, ai migliori risultati, sia in performances totali che di singola classe. Una buona efficienza la si ha anche per i dati di RPPA. Il tumore KIRC presenta le migliori risposte al metodo con distribuzioni confinate sempre al di sopra della soglia del 50% (tranne che per il caso di CNV), seguito da OV. Più problematici risultano i dati di miRNA di GBM, per i quali la distribuzione di valori, pur avendo mediana ben superiori al 50% presenta valori appartenenti ad un ampio intervallo ($\in [22, 78]\%$). Analogo comportamento lo si riscontra anche per LUSC, nel quale la Signature di miRNA ha la minor efficienza (50%) e quella di CNV

un'ampia varianza. Per le Signatures eterogenee, invece, è riscontrato un miglioramento in performances nella combinazione (miRNA, RPPA): ognuno dei 4 tumori presenta una migliore performance di classificazione sia sulla totalità che sulle singole classi. Ciò non vale per la combinazione (miRNA, mRNA). La causa è da ricercarsi nell'evidente migliore risposta dei mRNA rispetto ai miRNA: l'aggiunta di ulteriori informazioni ai dati di mRNA comporta solamente rumore nella classificazione, fattore che non si verifica per i miRNA e RPPA, le cui efficienze sono più simili.
Un confronto con i risultati di Yuan necessiterebbe dei dati delle distribuzioni di performances di ciascun classificatore utilizzato (Figura 4.1). Avendo a disposizione solamente i valori di mediana (senza informazioni né sull'efficienza di singola classificazione, né sullo spread delle distribuzioni), la verifica dei risultati del QDANet verterà solamente sul confronto tra i valori di mediana della distribuzione di performances totale di classificazione. Per una visualizzazione pratica dei risultati in Figura 4.3 sono riportati i grafici ottenuti plottando sulle ascisse i valori di mediana degli 8 classificatori utilizzati da Yuan messi a confronto con l'unico valore ottenuto dal QDANet: in questa visualizzazione grafica l'altezza delle colonne è priva di significato. Nel confronto è necessario tener presente che i risultati di Yuan fanno uso, per ogni caso analizzato, dell'intero set di probes a disposizione.

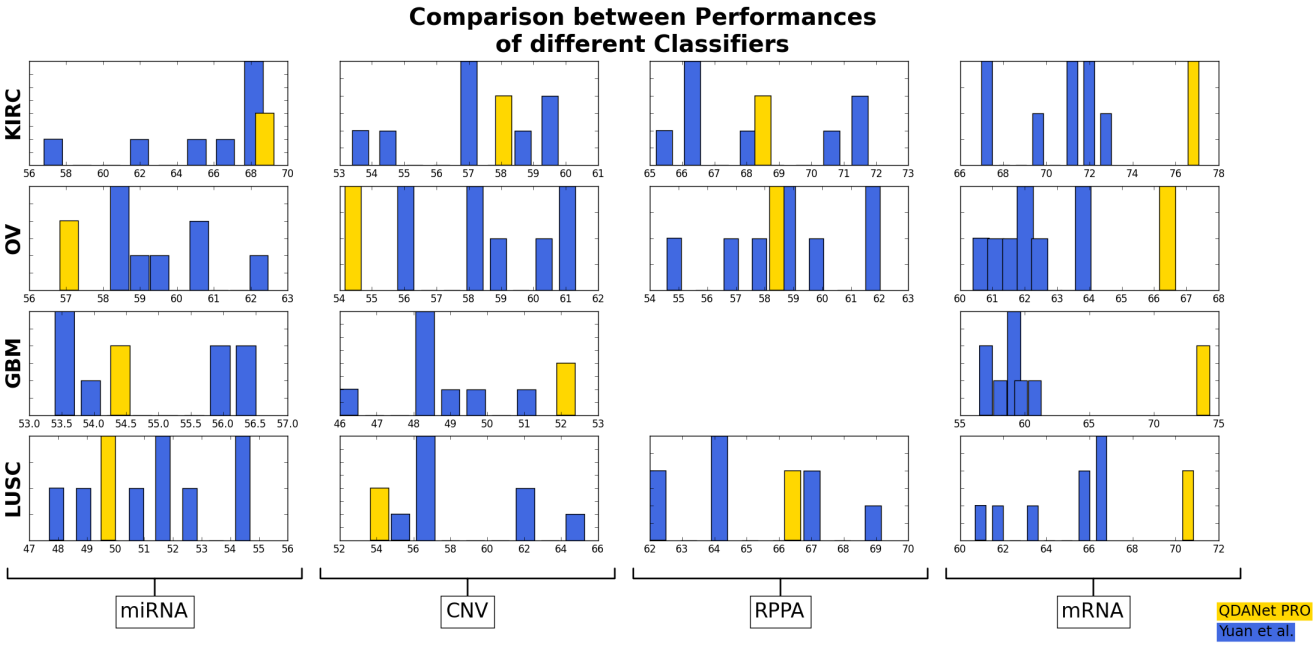


Figura 4.3: **(a)-(d)** Confronto dei risultati ottenuti dallo studio di Yuan et al. (blu) e l'algoritmo QDANet PRO (giallo) per i quattro tumori in analisi (**(a)** KIRC, **(b)** OV, **(c)** GBM e **(d)** LUSC). Le distribuzioni di Yuan sono ottenute dai valori di mediana degli 8 classificatori utilizzati nel suo studio (Figura 4.1).

Dal grafico è evidente come i risultati di mRNA siano superiori a quelli ottenuti da Yuan per ogni tipologia di tumore, ai quali è da aggiungere l'estrema riduzione della

dimensionalità messa in atto dal QDANet per questa tipologia di dato. Per i valori di miRNA le performances del QDANet risultano comparabili con quelle ottenute dagli 8 classificatori nei casi di GBM e LUSC, migliori per KIRC e lievemente inferiori per OV. Analoghi risultati li si riscontra anche per i dati di RPPA, per i quali l'efficienza del QDANet è sempre contenuta nell'intervallo di valori di Yuan. Per i dati di CNV i casi di OV e LUSC presentano risultati inferiori, da contrapporsi al caso di GBM per cui la best Signature presenta una migliore classificazione.

4.2 Test di robustezza delle best Signatures

La struttura dell'algoritmo QDANet PRO consente l'estrazione di un sotto-gruppo di probes connesse secondo una struttura a network. Tuttavia non viene contemplata la possibilità che un egual numero di probes, sconnesse e non, sia comunque in grado di dare risultati di classificazione parimente efficaci, se non anche migliori.
Dallo studio dei grafi e delle loro proprietà è possibile determinare un cosiddetto *modello nullo*, ovvero un grafo che possiede uguali caratteristiche strutturali, ma che viene originato in maniera random. In questo modo, il modello nullo può essere utilizzato come termine di confronto per il grafo in analisi, al fine di determinarne la presenza o meno di caratteristiche peculiari.
Nel caso dell'algoritmo QDANet PRO, la best Signature estratta costituisce il grafo sotto analisi, mentre dalla totalità delle probes a disposizione è possibile estrarre un numero di grafi equivalenti in modo random (con ripetizioni, ad indicare la presenza anche di *loop*) pari al coefficiente binomiale $\binom{n+k-1}{k}$, con n=numero di probes e k=dimensione della Signature. In questo modo viene testata l'ipotesi nulla di una corretta selezione di probes e valutata la scelta di una struttura a network per la costruzione della best Signature. Andando ad estrarre un numero n (nel nostro caso $n = 2 \times 10^4$) di probes in modo random si verifica la loro efficienza di classificazione. Ogni Signature Random è addestrata e validata sfruttando tutte le 100 combinazioni di training-test utilizzate in precedenza. Dalle 100 validazioni è ottenuta una distribuzione il cui valore di mediana è confrontato con quello ottenuto dal QDANet.
Al fine di considerare in un unico coefficiente numerico sia il valore di classificazione totale che quello di singola classe è utilizzato il coefficiente di correlazione di Matthews (MCC, cfr. Sez. 3.4). Poiché le tipologie di dato che meglio hanno risposto al QDANet sono i miRNA, RPPA e mRNA, l'analisi è stata svolta sulle relative Signatures. Nelle Figure 4.4, 4.5 e 4.6 sono presentate le distribuzioni di MCC stimati su 2×10^4 possibili Signatures Random, messe in relazione con il valore (mediano) relativo alla best Signature del QDANet (linea tratteggiata in rosso), per i dati di miRNA, RPPA e mRNA, rispettivamente.
Per la verifica della significatività del valore di MCC del QDANet è stato calcolato il percentile entro cui cade in relazione alla distribuzione di Signatures Random: data l'ipotesi nulla di una possibile compatibilità della best Signature con una scelta random,

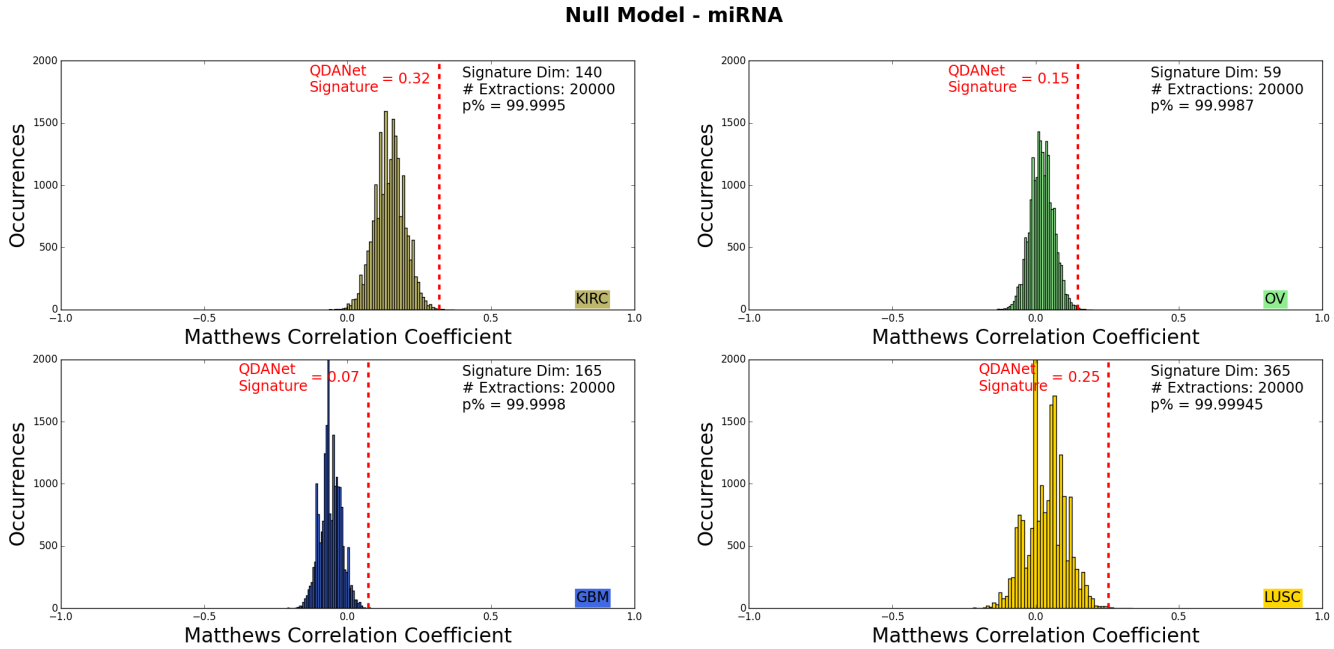


Figura 4.4: Verifica della robustezza della best Signature di miRNA estratta dal QDANet PRO mediante confronto con un modello nullo. Per 2×10^4 volte sono estratti in modo random un numero di probes pari alla dimensione della best Signature e calcolata l'efficienza di classificazione sui 100 training-test a disposizione. Per ogni coppia training-test è calcolato il coefficiente di correlazione di Matthwes ed al termine stimato il valore mediano. Al termine delle 2×10^4 simulazioni viene calcolato il percentile ($p\%$) entro cui cade il valore ottenuto dalla best Signature del QDANet PRO. Il test vuole verificare l'ipotesi di appartenza della best Signature alla distribuzione di Signatures Random.

l'obiettivo è quello di falsificarla con un elevato percentile. I valori riscontrati sono riportati in ognuno dei grafici.

É evidente l'alta efficacia della best Signature di mRNA per la quale il valore di MCC risulta nettamente maggiore rispetto alla distribuzione del modello nullo[5]: nessuna delle Signatures Random presenta un eguale o maggiore valore di MCC rispetto alla Signature estratta con il QDANet PRO e per ogni tumore è possibile falsificare in ottima confidenza l'ipotesi nulla. Anche nella valutazione delle Signatures di miRNA i risultati mostrano l'alta efficacia del QDANet nella selezione delle probes. Anche in questo caso l'appartenza della best Signature alla distribuzione random è falsificata secondo l'elevato valore del percentile riscontrato in ognuno dei tumori[6]. Per i dati di RPPA è ancora confermata l'efficienza su LUSC ($p\% = 99.99965$) e OV ($p\% = 99.99235$),

[5]Si noti che il valore di MCC della best Signature rimane comunque distante dal valore unitario massimo, in virtù della non perfetta classificazione ottenuta dalla validazione. Solamente nel caso di una massima (100%) performance su entrambe le classi il coefficiente di Matthews raggiunge valore unitario (positivo).

[6]Su una popolazione di 2×10^4 Signatures Random solamente 10 nel caso di KIRC, 26 per OV, 4 per GBM e 11 per LUSC presentano un valore di MCC superiore alla best Signature corrispondente.

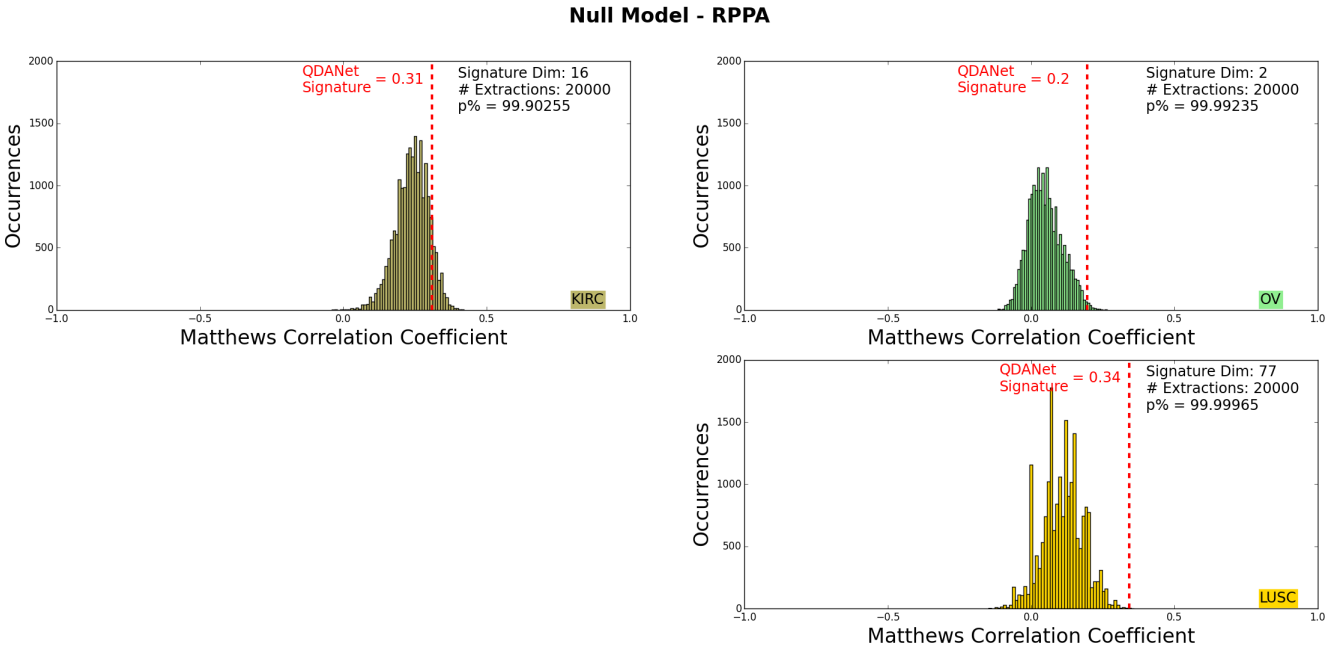


Figura 4.5: Verifica della robustezza della best Signature di RPPA estratta dal QDANet PRO mediante confronto con un modello nullo.

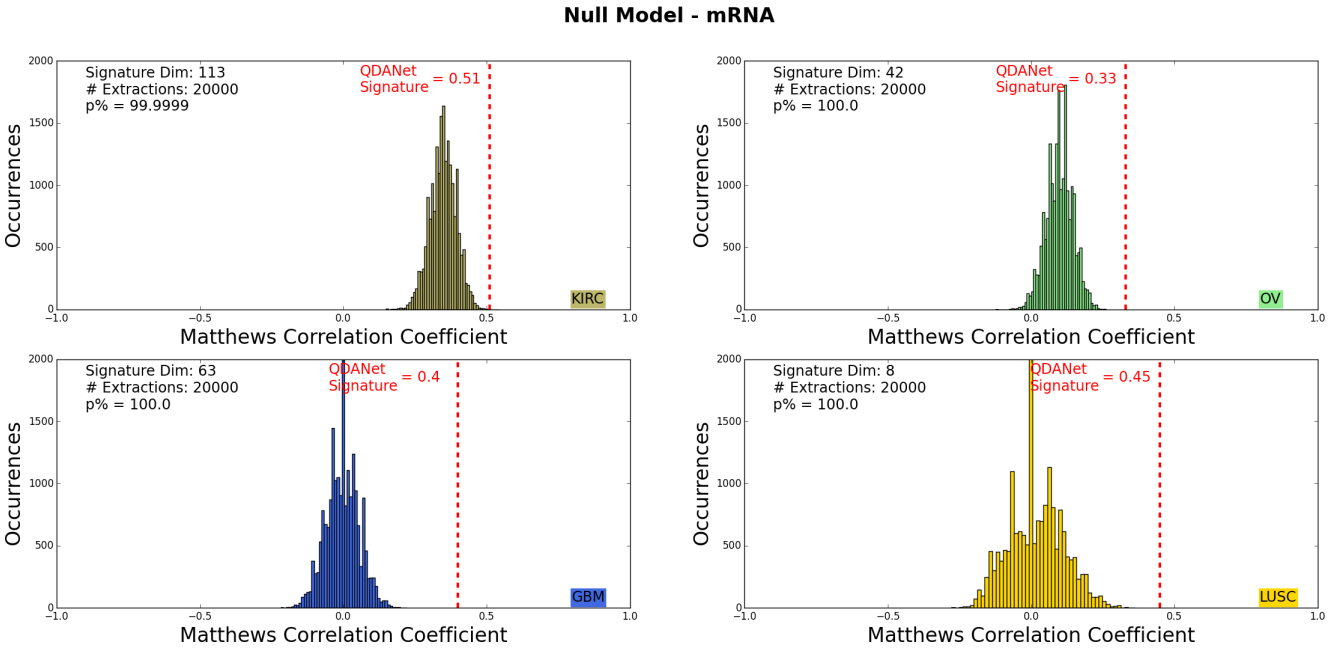


Figura 4.6: Verifica della robustezza della best Signature di mRNA estratta dal QDANet PRO mediante confronto con un modello nullo.

mentre la Signature di KIRC presenta 1949 casi su 2×10^4 in cui il valore di MCC viene superato (o eguagliato). Quest'ultimo caso, più problematico rispetto ai precedenti, risulta accettabile con un α-level del 10%: visto l'ammontare delle Signatures Random analizzate il risultato rimane accettabile, mettendo in luce, però, la fragilità del metodo QDANet nell'elaborazione di alcuni dati di RPPA.

Si conclude quindi che per ognuno dei casi analizzati il test dei percentili ha permesso di falsificare l'ipotesi nulla, mettendo in evidenza l'efficacia del metodo QDANet nella selezione delle Signatures.

4.3 Test di robustezza della struttura a network

Presupposto di partenza del metodo QDANet è l'efficacia di una struttura a network per la classificazione, a simboleggiare l'interazione più o meno complessa di un sottogruppo di probes. La complessità della classificazione dei samples tumorali, tuttavia, potrebbe necessitare di più Signatures non interagenti, fino al caso limite di probes totalmente disgiunte (nel nostro caso coppie). Un ulteriore test di robustezza per le scelte effettuate dal QDANet è, quindi, la valutazione dell'efficienza di probes sconnesse per la classificazione.
La prima parte dell'algoritmo ci fornisce l'elenco ordinato delle best couples dal quale poter attingere per questo tipo di valutazione. Come per il test precedente, la stima delle performances dell'unione di coppie disgiunte è effettuata considerando il valore di MCC, piuttosto che la semplice performance, calcolato come valore mediano della distribuzione ottenuta sui 100 training-test. Selezionando le prime 100 best couples di mRNA, miRNA e CNV, per ognuno dei tumori, sono state aggiunte sequenzialmente[7] le probes secondo l'ordinamento fornito dal QDANet e valutata la loro efficacia di classificazione. Nelle Figure 4.7, 4.8 e 4.9 sono mostrati gli andamenti dei coefficienti di Matthews all'aumentare della dimensione della Signature disgiunta nei quattro casi in analisi.
Gli andamenti ottenuti dimostrano l'efficacia del QDANet nella selezione della Signature per i dati di miRNA e RPPA nei casi di OV, GBM e LUSC, mentre per KIRC solamente la Signature di miRNA risulta migliore. La best Signature di RPPA di KIRC, infatti, produce un valore di MCC comparabile, se non inferiore nel primo tratto, al valore ottenibile da Signatures disgiunte. L'alta efficienza dei casi di miRNA è associabile alle grandi dimensioni delle best Signatures estratte dal QDANet: tranne che per il caso di OV, infatti, le best Signatures possiedono dimensioni maggiori rispetto all'intervallo di valutazione, mettendo in evidenza la necessità di un ingente numero di probes per una buona classificazione. Medesima situazione la si riscontra per le Signatures di mRNA, per le quali i valori di MCC risultano comparabili o talvolta superiori a quelli della best Signature di QDANet.
Come per il confronto con i risultati di Yuan, è necessario tener conto dell'efficienza di classificazione unita alla dimensione della Signature considerata. Nel caso dei miRNA, escluso OV, ognuna delle best Signatures estratte possiede dimensioni superiori all'intervallo considerato. La maggior efficacia della best Signature rispetto alle coppie sconnesse dimostra che i casi in analisi necessitano di un numero superiore di probes per

[7] Per non avere ripetizioni è stata utilizzata la funzione unique sull'insieme di probes selezionate prima della classificazione.

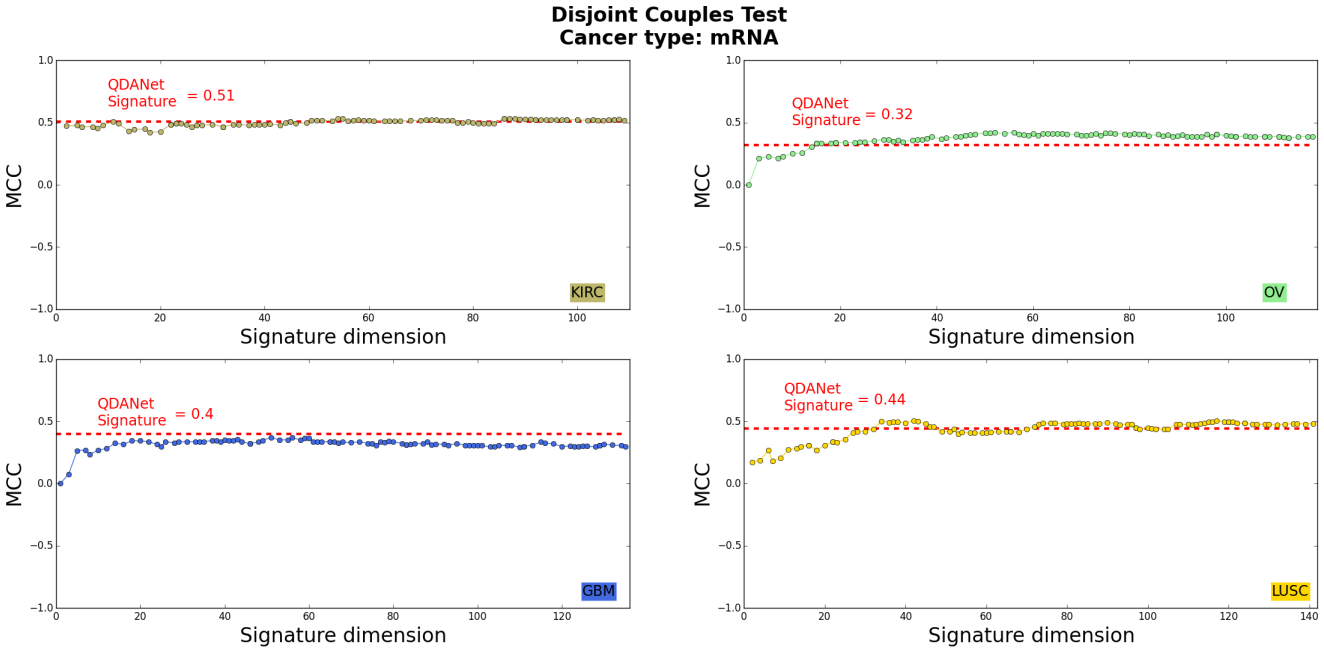


Figura 4.7: Selezione di best couples di mRNA disgiunte dall'ordinamento fornito dalla prima parte del QDANet: si intende verificare l'efficienza di classificazione di sottogruppi non necessariamente interagenti e topologicamente assimilabili ad un network. L'ordinamento delle coppie è dettato dal valore di performance totale pesata in numerosità delle singole classi. Per ognuno dei quattro tumori è stato calcolato il coefficiente di Matthews mediano della distribuzione ottenuta sui 100 training-test a disposizione. Ad ogni step è stata aggiunta la best couple successiva nell'ordinamento.

la classificazione e l'importanza nell'utilizzare una struttura a network per l'estrazione. Per i mRNA, invece, la dimensione delle best Signatures è sempre contenuta nell'intervallo considerato per Signatures Sconnesse: nei casi di KIRC, GBM e LUSC nonostante l'andamento possa risultare confrontabile (o superiore) con quello del QDANet, a parità di dimensione le best Signatures risultano maggiormente performanti. Eccezione la presenta OV, per il quale con un medesimo o leggermente inferiore numero di probes è possibile ottenere un valore di MCC più elevato. Considerazioni analoghe valgono per i dati di RPPA, in cui l'eccezione in questo caso è data da KIRC.

Per completezza sono state estratte le Signatures Sconnesse ottenute dall'ascissa relativa al punto di massimo dell'andamento di MCC precedente: l'analisi si è rivolta ai dati di mRNA di tutti e 4 i tumori ed ai dati di RPPA relativi a KIRC. In Figura 4.10 sono riportate le distribuzioni box plot delle validazioni delle Signatures Sconnesse (cfr. risultati di mRNA e RPPA di KIRC in Figura 4.2).

I risultati di classificazione ottenuti risultano migliori sia in termini di performances totale di classificazione che in singola classe per KIRC e LUSC mentre le best Signatures di GBM e OV rimangono migliori. Lo stesso si verifica per i dati di RPPA di KIRC. Al contempo le Signatures Sconnesse di KIRC (RPPA) e LUSC (mRNA) possiedono una

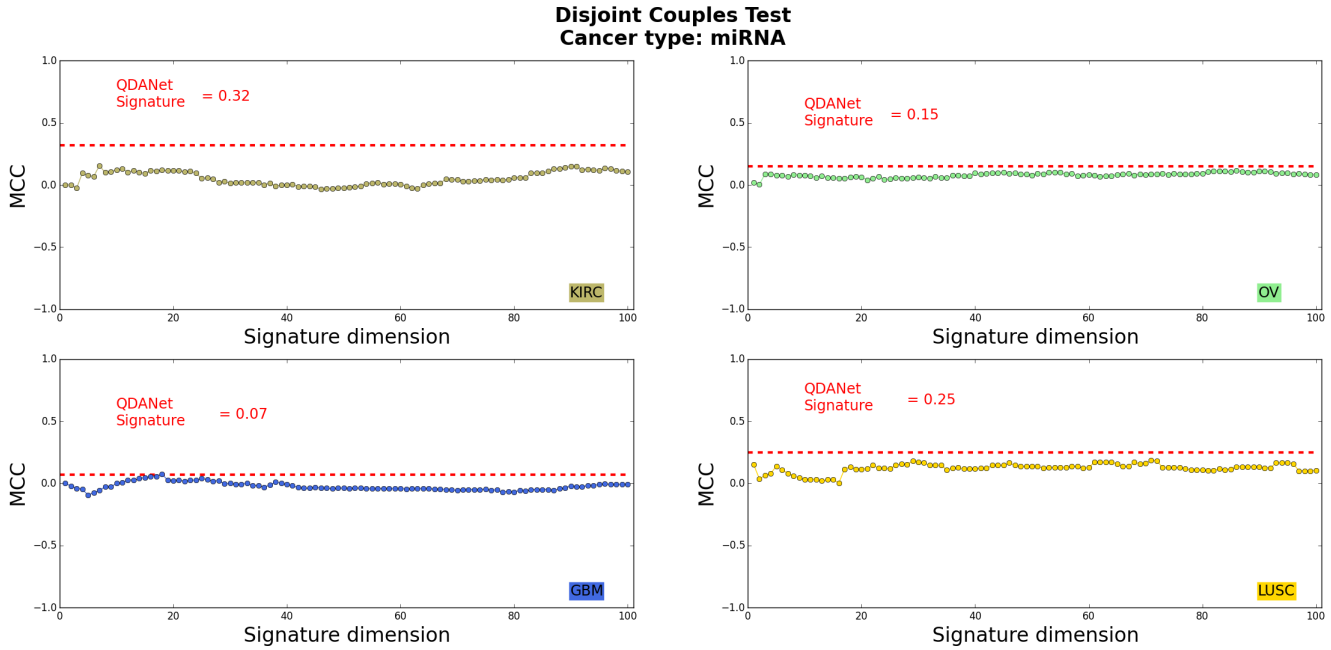


Figura 4.8: Selezione di best couples di miRNA disgiunte dall'ordinamento fornito dalla prima parte del QDANet. L'ordinamento delle coppie è dettato dal valore di performance totale pesata in numerosità delle singole classi.

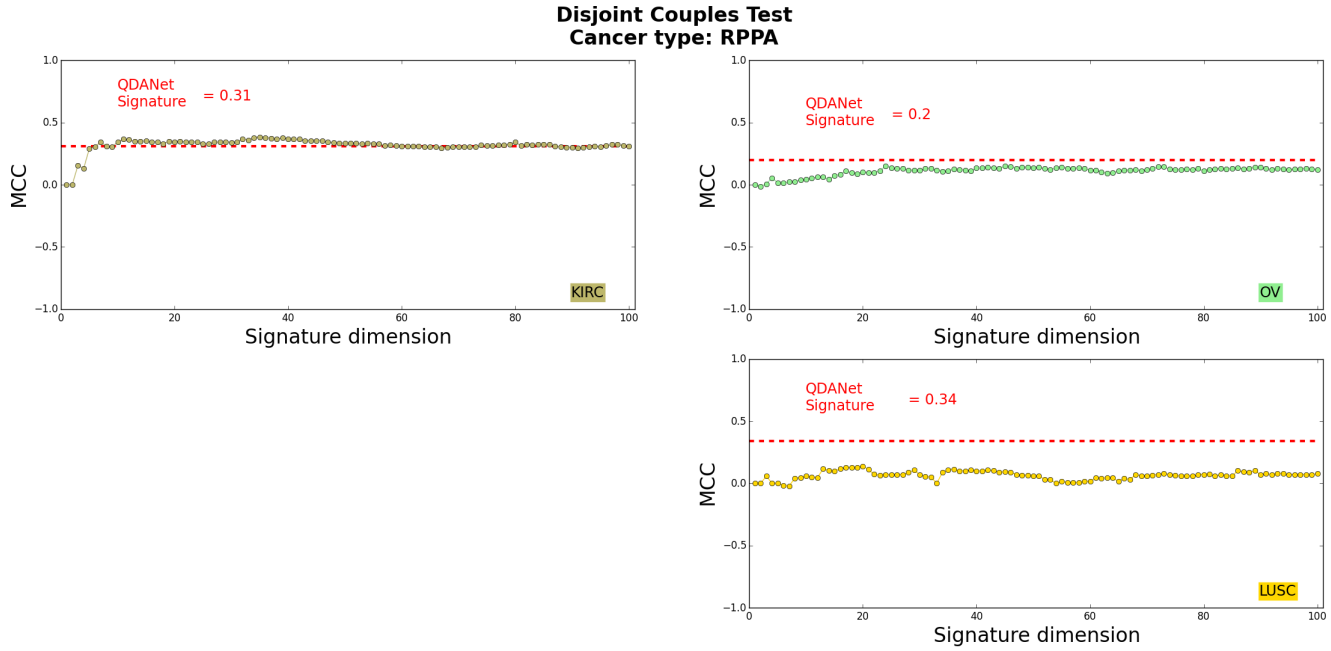


Figura 4.9: Selezione di best couples di RPPA disgiunte dall'ordinamento fornito dalla prima parte del QDANet. L'ordinamento delle coppie è dettato dal valore di performance totale pesata in numerosità delle singole classi.

dimensione superiore alle best Signatures del QDANet. Eccezione la presenta KIRC nella Signature Sconnessa di mRNA, con una maggiore performance ed una dimensione

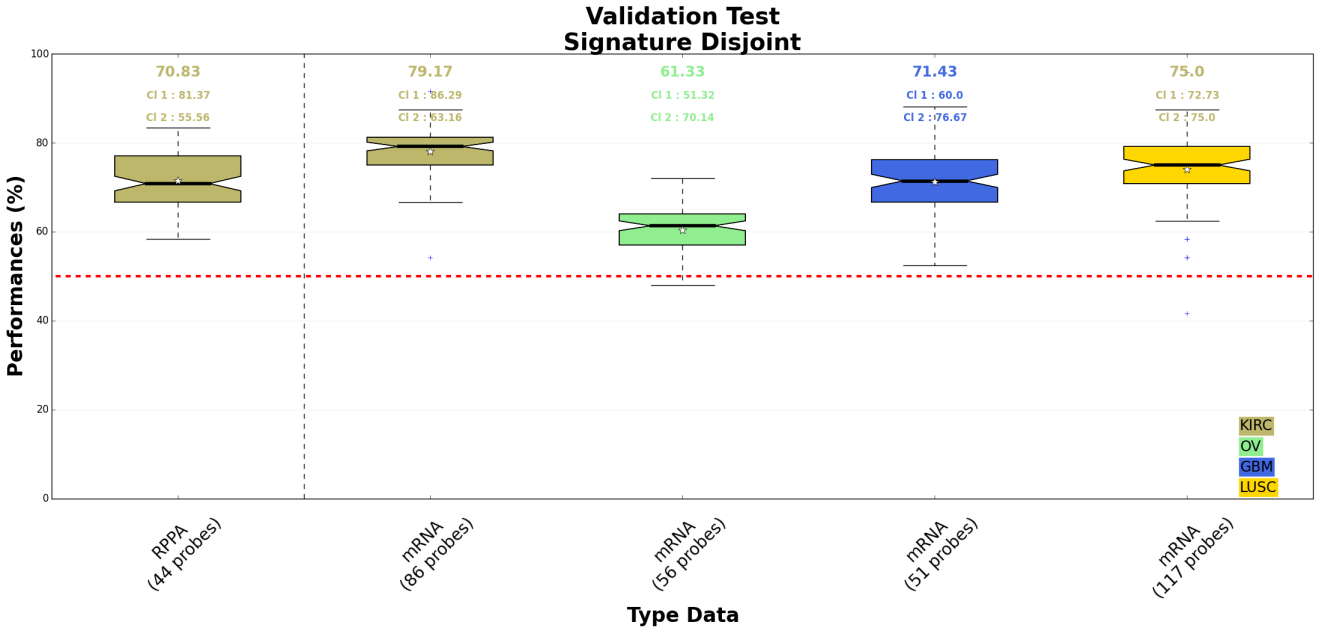


Figura 4.10: Test di validazione della Signature disgiunta di mRNA relativa al punto di massimo dell'andamento di MCC riportato in Figura 4.7. Dall'andamento del coefficiente di Matthews studiato per la selezione di coppie disgiunte di mRNA emerge una maggiore risposta in classificazione nello scegliere le prime 72 coppie di KIRC (86 probes), le prime 41 coppie di OV (56 probes), le prime 35 coppie di GBM (51 probes) e le prime 83 coppie di LUSC (117 probes). Per il caso di RPPA la miglior risposta di KIRC la si ha con le prime 49 coppie (35 probes). L'ordinamento delle coppie è dettato dal valore di performance totale pesata in numerosità delle singole classi.

più ridotta. Analizzando le probes che compongono le due Signatures si riscontra la presenza di 52 probes comuni ($> 60\%$), dimostrando una forte correlazione tra le coppie sconnesse e quelle selezionate dal QDANet.

Volendo mantenere un'unica struttura a network e continuando a sfruttare l'ordinamento delle coppie elaborate dal QDANet (ordinamento pesato) è possibile effettuare anche una valutazione inversa rispetto alla normale funzionalità dell'algoritmo, cioé andando ad accrescere la Signature. Partendo dalla coppia maggiormente performante si è determinato l'andamento di MCC, andando ad "attaccare" in modo ricorsivo le coppie con una probe in comune secondo l'ordinamento delle performances. In Figura 4.11 è riportato l'andamento di MCC in funzione della dimensione della Signature ottenuta attraverso questa modalità di test, per i dati di mRNA.

Anche in questo caso la potenzialità del QDANet viene messa in discussione dalla presenza di Signatures Connesse in grado di raggiungere un maggiore valore di MCC. Solo nel caso di GBM la best Signature di QDANet continua ad essere la migliore scelta, mentre quelle di KIRC e LUSC circa equivalenti. Per OV, invece, l'andamento di MCC è superiore a quello del QDANet per quasi l'intero range di Signatures Connesse valutate. Il confronto tra le dimensioni delle Signatures dimostra ancora una volta la validità

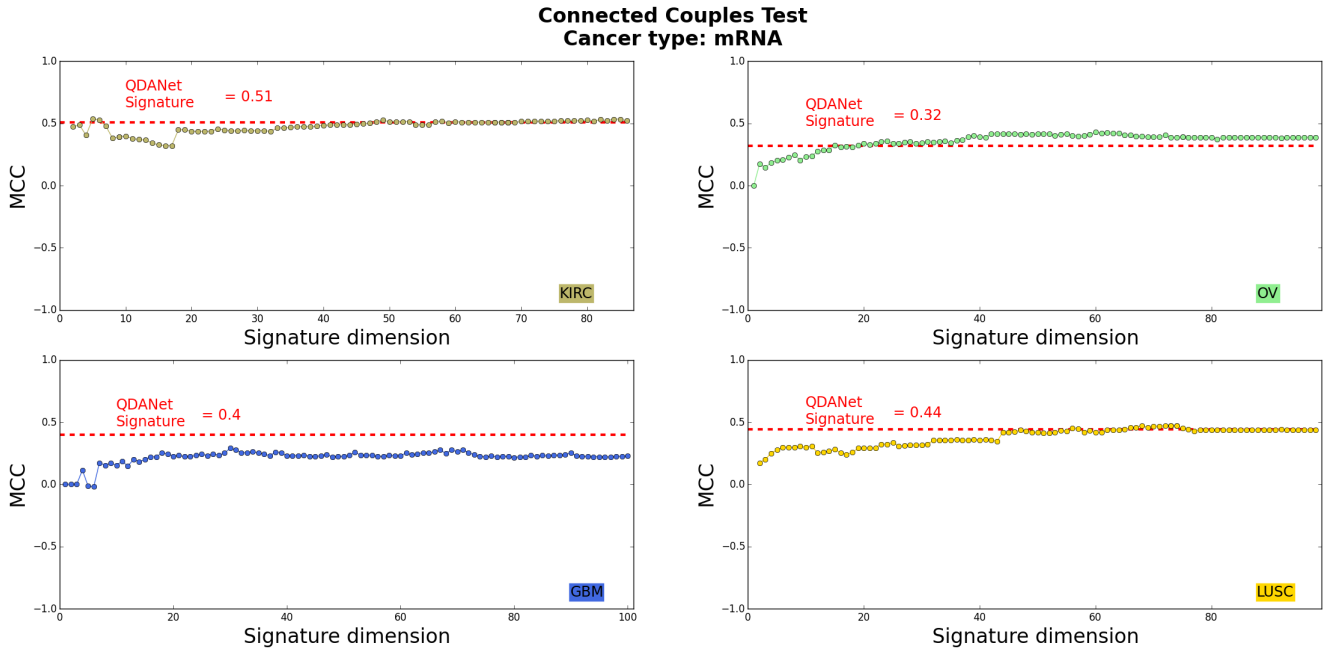


Figura 4.11: Selezione di best couples di mNRA connesse dall'ordinamento fornito dalla prima parte del QDANet: si intende verificare l'efficienza di classificazione di sottogruppi interagenti e topologicamente assimilabili ad un network ottenuti a partire dalla coppia a maggiore efficienza. L'ordinamento delle coppie è dettato dal valore di performance totale pesata in numerosità delle singole classi. Per ognuno dei quattro tumori è stato calcolato il coefficiente di Matthews mediano della distribuzione ottenuta sui 100 training-test a disposizione. Ad ogni step è stata aggiunta la best couple successiva nell'ordinamento e presentante una probe in comune alle precedenti.

del QDANet: escluso il caso di KIRC per il quale l'intervallo considerato è inferiore alla dimensione della best Signature[8], sia GBM che LUSC a parità di dimensione possiedono una miglior risposta con le best Signatures piuttosto che con le Signatures Connesse. Come per il caso precedente sono state estratte le Signatures relative ai punti di massimo dell'andamento del MCC e validate sui test set a disposizione (Figura 4.12).

Dalla validazione non si riscontra miglioramento in termini di performances per nessuna delle 4 tipologie di tumore. Sia in termini di performance totali che di singola classe, i valori ottenuti dalle best Signatures del QDANet risultano maggiori di quelli ricavati da questo secondo test.

Questo studio ha messo in evidenza alcune sensibilità dell'algoritmo nella scelta del miglior sottogruppo di dati. Il maggiore grado di libertà lasciato all'algoritmo risiede, infatti, nella scelta del valore di sogliatura delle coppie da considerare. Questo, pur consentendo da un lato la possibilità di avere Signatures di dimensioni comparabili alle necessità degli studi successivi, rende il metodo non univoco nell'estrazione della reale

[8]Si noti che l'andamento sembra aver raggiunto una sorta di saturazione, ragion per cui l'aggiunta di ulteriori probes non comporterebbe grandi variazioni.

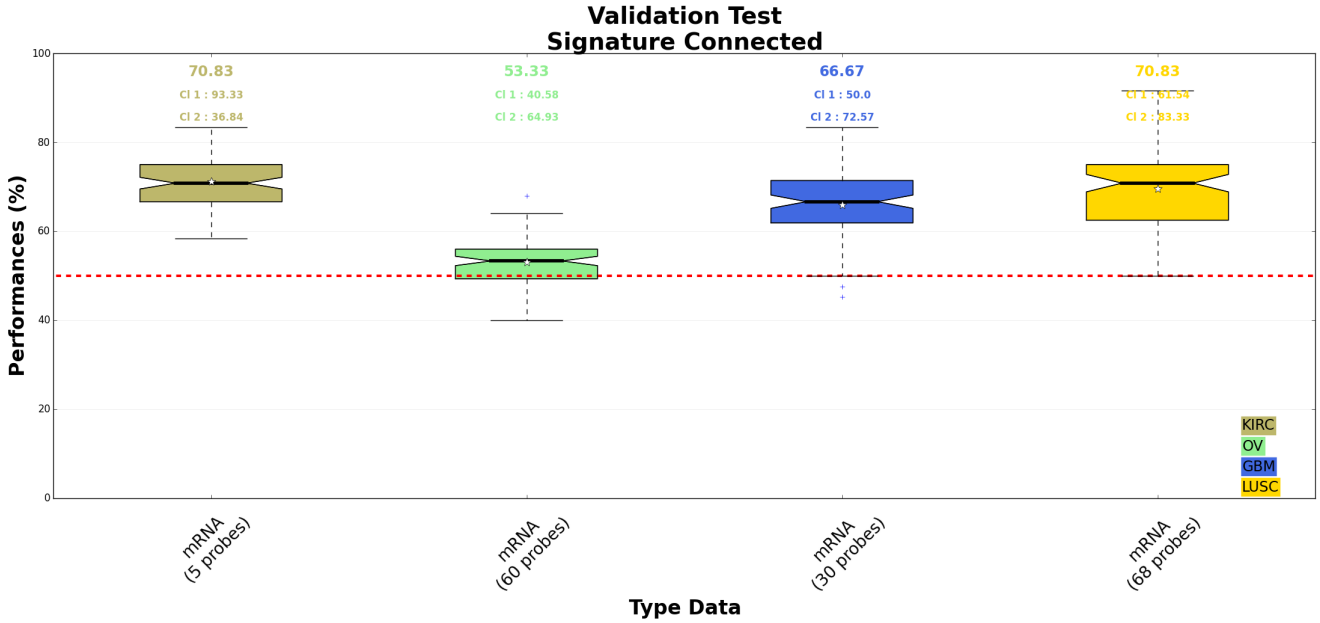


Figura 4.12: Test di validazione della Signature connessa di mRNA relativa al punto di massimo dell'andamento di MCC riportato in Figura 4.11. Partendo dalla coppia top scorer di classificazione viene creata la Signature andando a connettere le componenti a maggior performance. La dimensione della Signature è data dall'ascissa relativa al punto di massimo di MCC ricavato per le coppie connesse. L'ordinamento delle coppie è dettato dal valore di performance totale pesata in numerosità delle singole classi.

best Signature. A livello algoritmico non è possibile, con la struttura attuale, ottenere il sottogruppo con effettiva miglior efficienza di classificazione, ma selezionare un sotto gruppo altamente performante con le dimensioni (più o meno) desiderate.

4.4 Test di robustezza del metodo QDANet PRO

I risultati sulle best Signatures ottenuti fino qui fondano la loro validità sull'ipotesi che il metodo QDANet PRO non risenta in maniera significativa del training set utilizzato per l'estrazione. In altre parole, abbiamo assunto finora che la scelta di considerare la prima suddivisione dei dati proposta da Yuan et al. per la determinazione della Signature conduca alle stesse probes delle restanti 99 possibilità.
A conferma della scelta effettuata e a verifica della robustezza del metodo QDANet PRO al partizionamento dei dati è sufficiente considerare le *top scorer probes* presenti per ciascuna delle 100 possibili simulazioni della prima parte (cfr. Sez. 3.1) dell'algoritmo. Da ogni simulazione è estratto il 10% o 1% della totalità delle possibili coppie (ordinate) e sono calcolati il numero di occorrenze delle coppie delle 100 simulazioni possibili (cfr. Tabella 4.13 per il numero di coppie totali). Per verificare la presenza di overlap nelle best couples sono stati considerati i dati di miRNA, CNV e RPPA relativi a KIRC. In Figura 4.13 sono presentati gli istogrammi di occorrenze, in cui viene riportato il

numero di coppie che compaiono in oltre il 75% dei training set (con relativo valore percentuale rispetto al numero di coppie selezionate).

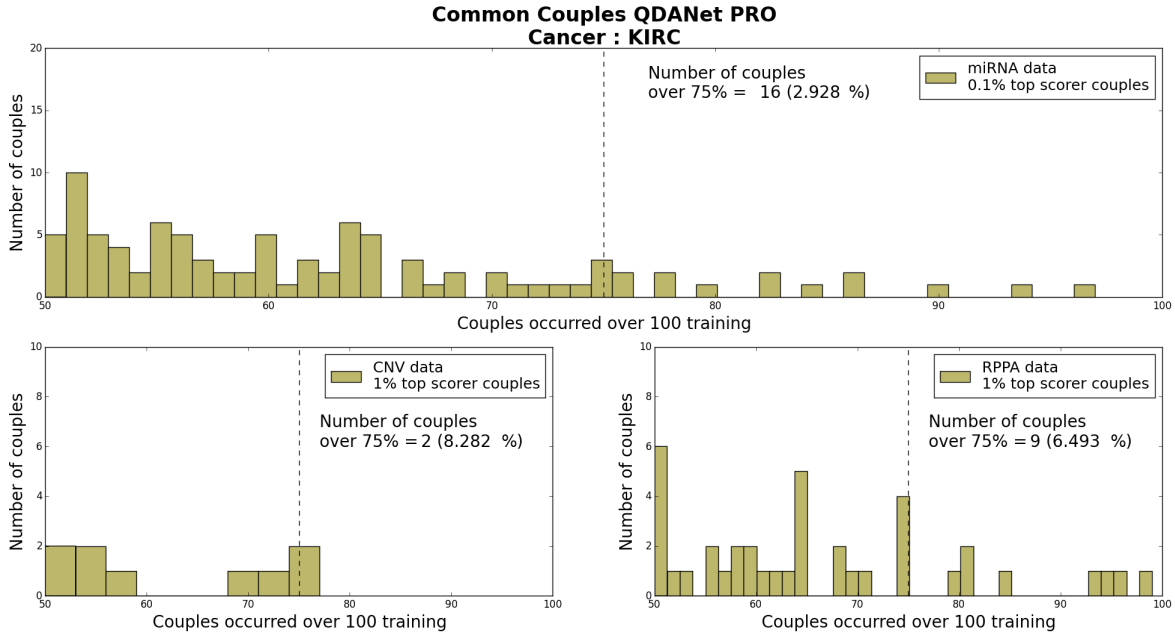


Figura 4.13: Istogrammi di occorrenze delle best couples individuate nei 100 training set a disposizione. Sono selezionati il 10% della totalità delle coppie per i dati di miRNA e l'1% per quelli di CNV e RPPA (meno numerosi) relativi a KIRC. In ognuno dei grafici sono delineate (linea tratteggiata nera) il numero di coppie che si presentano in oltre il 75% dei training, con relativa percentuale rispetto al numero di coppie selezionate in ognuna delle simulazioni.

Dai grafici risulta che 94 coppie di miRNA risultano comuni nell'oltre il 50% dei casi, di cui 16 oltre il 75%, corrispondenti a $\sim 3\%$ della percentuale di coppie valutate. Per i dati di CNV e RPPA la percentuale risulta relativamente maggiore andando però a considerare l'1% delle top scorer, con 9 coppie sopra il 50% e 2 nell'oltre il 75% per i dati di CNV e 39 coppie sopra il 50% e 9 nell'oltre il 75% per i dati di RPPA. Confrontando le probes appartenenti alle coppie ad occorrenza superiore al 75% con quelle presenti nelle best Signatures del QDANet si riscontra un overlap di 12/20 (60%) per i miRNA, 4/4 (100%) per i CNV e 7/13 (53%) per gli RPPA.
Considerando l'enorme numero di coppie valutate dall'algoritmo QDANet, i risultati di overlap mettono in evidenza una sensibilità del metodo al training set a disposizione. Tuttavia, dovendo estrarre Signatures di dimensione il più possibile ridotta, nelle percentuali di dati considerate il numero di coppie di overlap risulta sufficiente per dimostrare la presenza di un sotto gruppo di probes maggiormente indicative per la classificazione. Inoltre la presenza delle probes appartenenti alle coppie a maggior frequenza all'interno delle best Signatures avvalora la sufficiente robustezza dell'algoritmo al training set.
Utilizzando sempre le coppie *top scorer* è calcolata l'occorrenza delle singole probes all'interno delle coppie, a verificare la presenza di particolari probes singole altamenti

efficienti nella classificazione. Prendendo in analisi i dati precedenti sono selezionati lo 0.001% delle best couples di miRNA, lo 0.01% di quelle di RPPA e lo 0.1% di quelle di CNV. Da queste sono stati ottenuti gli istogrammi riportati in Figura 4.14. Si noti che considerando 100 training, le possibili occorrenze delle singole probes non sono necessariamente limitate a 100 poiché più coppie possono presentare un elemento comune. Nondimeno risulta utile fissare come sogliatura un valore pari alle 75 occorrenze.

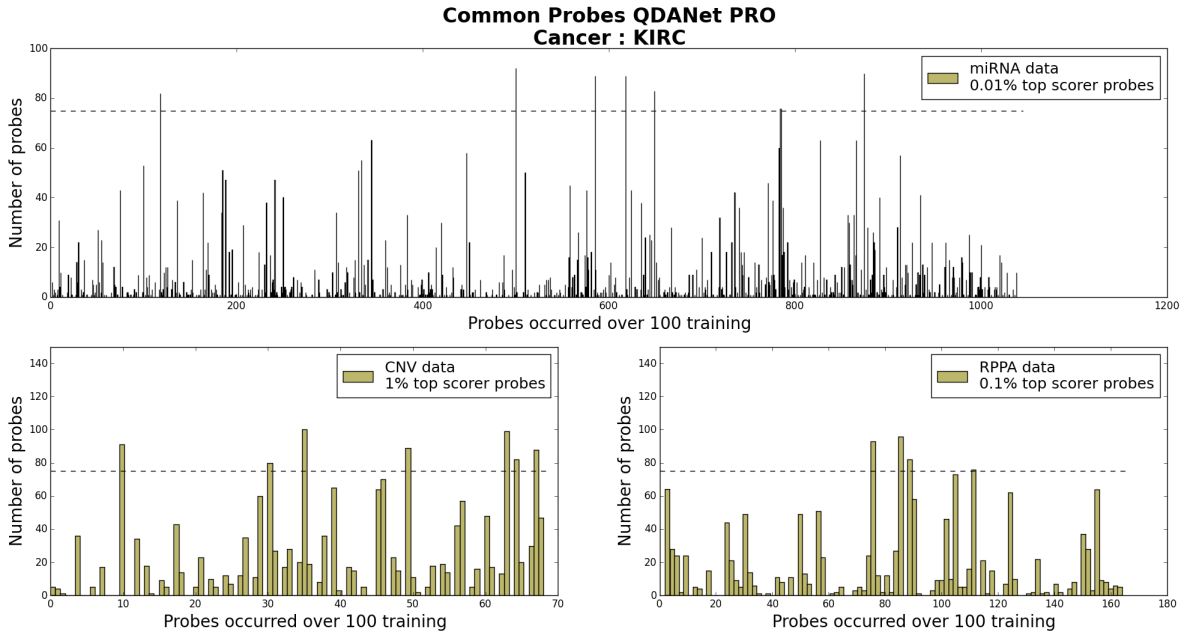


Figura 4.14: Istogrammi di occorrenze delle probes presenti nelle best couples individuate nei 100 training set a disposizione. Sono selezionati lo 0.001% della totalità delle coppie per i dati di miRNA, lo 0.01% per quelle di RPPA e lo 0.1% per quelle di CNV relativi a KIRC. In ognuno dei grafici è evidenziata la soglia delle 75 occorrenze.

Dai grafici risulta che solo un limitato numero di probes presenta un'elevata frequenza all'interno delle poche best couples selezionate. Ciò nonostante in tutti e tre i casi tendono a comparire quasi tutte le probes: 581 probes per i miRNA, 62 probes per i CNV e 101 probes per gli RPPA.
Estraendo le sole probes con una frequenza ≥ 75 si arriva a selezionare solamente 8 probes per i miRNA, 7 probes per i CNV e 4 probes per gli RPPA. É importante notare come di queste probes una buona percentuale sia contenuta all'interno delle best Signatures selezionate dal QDANet: in particolare 4/8 (50%) per i miRNA, 7/7 (100%) per i CNV e 3/4(75%) per gli RPPA. Questo risultato continua ad avvalorare l'efficacia del QDANet nella selezione delle probes e dimostra l'influenza di un loro sottoinsieme, rispetto alla totalità, per la classificazione.

Capitolo 5

Conclusioni

In questo lavoro è stata testata per la prima volta su dati disponibili in letteratura l'efficacia dell'algoritmo QDANet PRO, ideato dal gruppo di Biofisica dell'Università di Bologna. Il database da cui ho attinto i dati è il *The Cancer Genome Atlas* (TGCA), un riferimento nel settore della (Biological) Big Data Analytics. Per valutare l'efficienza del metodo, i risultati sono stati confrontati con l'attuale stato dell'arte, fornito dagli studi pubblicati su *Nature* di Yuan et al., seguendo la medesima pipeline di pre-processing. L'analisi ha riguardato 4 differenti linee tumorali, descritte da 4-5 tipologie di dato molecolare ciascuna. Per l'analisi di tali volumi di dati è stato indispensabile l'utilizzo dei Servers a disposizione al gruppo di ricerca ed un elevato livello di parallelizzazione del codice.
A conclusione del lavoro è emersa l'efficacia del metodo QDANet PRO nella classificazione di dati molecolari, con particolare efficienza per i dati di mRNA. Le performances ottenute risultano comparabili, se non superiori, a quelle ottenibili dai classificatori maggiormente efficienti nella Big Data Analytics testati da Yaun et al., i quali sfruttano algoritmi di computazione non lineare e l'intero set di probes a disposizione. I risultati ottenuti sulle altre tipologie di dati molecolari sono stati soddisfacenti e comparabili con quelli dello stato dell'arte.
La miglior risposta da parte dei dati di mRNA è imputabile alla stessa biologia dei geni: questi, infatti, spesso tendono a variare i livelli di espressione in relazione a regolatori up/down. Attualmente non è dimostrato un analogo comportamento per i miRNA, giustificando la minor efficienza di quest'ultimi. Per quanto riguarda CNV e RPPA, invece, il calo delle performances è riconducibile all'intrinseca natura di questi dati, prodotti da misurazioni per le quali non è semplice fornire un'univoca spiegazione.
Seguendo l'idea proposta da Yuan et al. nel combinare differenti sorgenti di dati per l'incremento delle performances di classificazione, si è testata l'efficienza di Signatures eterogenee, i.e combinazioni di diversi dati molecolari. Dallo studio è emersa una maggiore efficienza mediante la combinazione dei dati di miRNA e RPPA. Altrettanto non si è potuto dire delle combinazioni contenenti i dati di mRNA: quest'ultimi, in virtù dell'ottima efficacia posseduta singolarmente, non consentono un aumento di performances con l'aggiunta di ulteriori tipologie di campione.

Per quanto riguarda i dati relativi a GBM, una delle tipologie di tumore maggiormente studiate in virtù dell'estrema complessità ed aggressività, le best Signatures si sono dimostrate particolarmente significative. L'algoritmo ha permesso di selezionare un sottoinsieme di probes a performances di classificazione nettamente superiori a quelle riportate in letterature da altri metodi.

Quanto ottenuto permette di avvalorare l'ipotesi proposta sul comportamento dei livelli di espressione genica: la sua variazione è modellizzabile attraverso un comportamento a soglia piuttosto che ad uno bistabile, consentendo una classificazione mediante curve a basso livello di complessità. Secondo queste ipotesi l'algoritmo QDANet PRO costituisce un buon metodo per selezionare le probes e per preservare una facile interpretazione del loro comportamento sul piano biologico.

Lo studio condotto ha permesso anche di testare le varie componenti dell'algoritmo. Il confronto tra la best Signature del QDANet PRO e la distribuzione di modelli nulli ha dimostrato l'efficacia del metodo nel selezionare le probes: le performances di classificazione del QDANet PRO sono risultate sempre nettamente superiori rispetto a quelle ottenibili con una scelta random di un egual numero di probes. Anche in questo caso la massima efficacia la si è ottenuta dalle best Signatures di mRNA, convalidando la significatività di quella di GBM.

Per quantificare l'importanza della struttura a network delle best Signatures sono stati verificati diversi metodi di features extraction di tipo ricorsivo, considerando Signatures sia disgiunte che connesse. Anche in questo caso, a parità di dimensione, le best Signatures del QDANet PRO sono risultate migliori in performance di classificazione, le quali sono state stimate mediante MCC. Questo risultato convalida l'ipotesi iniziale sull'importanza di considerare fenomeni di interazione (almeno a livello biologico) tra le probes.

L'attuale implementazione dell'algoritmo lascia diversi gradi di libertà all'utente, il quale è libero di scegliere il miglior compromesso tra dimensionalità della Signature e performances di classificazione. Dall'analisi delle strutture a network si è confermata la criticità di questi gradi di libertà: il metodo QDANet PRO non è stato ideato per la selezione dell'effettivo miglior sottogruppo di probes. A tal proposito, però, occorre sottolineare come la stessa costruzione della best Signature del QDANet PRO comporti un'interpretazione biologica spesso consistente. I risultati ottenibili dal QDANet PRO, infatti, appartengono solamente al piano statistico dell'analisi ed ogni best Signature selezionata necessita di una verifica anche sul piano biologico. Qualora questo secondo step non si verifichi, sarà possibile ottenere una best Signature differente sia in dimensionalità che in performances. Per questo motivo, i risultati ottenuti in questo lavoro di tesi verranno presto sottomessi anche a studi di carattere biologico, per l'identificazione di possibili pathways d'interazione in cui sono coinvolte le probes selezionate.

É stato effettuato un ulteriore studio sulla selezione delle probes da parte dell'algoritmo, considerando l'overlap sia di coppie che di singole probes top scorers per 100 differenti simulazioni. In questo modo si è potuto verificare che in ottime percentuali

($\geq 50\%$) ognuna delle best Signatures risulta composta da coppie e probes top ranking a prescindere dalla suddivisione del campione a disposizione.
L'analisi finora condotta attraverso i numerosi test sulle componenti critiche dell'algoritmo ha confermato la sua robustezza. Volendo effettuare ulteriori verifiche sarà necessario applicare l'algoritmo ad ulteriori databases, anche più voluminosi del TCGA. Per farlo, tuttavia, sarà indispensabile l'utilizzo di tecnologie hardware più complesse (HPC e Cloud) ed un livello di parallelizzazione ancora superiore per l'abbattimento dei tempi di calcolo. In questo modo sarà possibile svolgere considerazioni statistiche ancor più solide e perfezionare la regolazione dei parametri dell'algoritmo.
La ridondanza delle informazioni e la necessità di selezionare sottogruppi di features costituiscono rispettivamente caratteristiche e obiettivi primari per la Big Data Analytics. Lo studio ha messo in evidenza le potenzialità dell'algoritmo QDANet PRO nell'estrazione delle informazioni dal rumore, identificandolo come nuova metodologia di analisi per problemi di dimensionality reduction e feature extraction. L'algoritmo arriva a costituire, quindi, un potente metodo di analisi per numerose applicazioni in campo biologico e non solo. Le sue caratteristiche lo rendono, infatti, applicabile anche a problemi di imaging biomedico, ponendosi come metodo di features extraction, o per problemi di estrazione di segnali dal background, problematica fondamentale di numerosi settori di ricerca tra cui la fisica delle alte energie. Con le dovute modifiche al pre-processing dei dati la metodologia sfruttata dal QDANet PRO può porsi come valida ed efficiente alternativa per lo studio di numerosi problemi di classificazione, spaziando la sua applicabilità a diversi livelli di scala per le dimensioni dei dati ed alle diverse tecnologie hardware a disposizione.

Appendice A - Bayesian Classifier (Python Code)

```
def Classify(Training, Test, Label_training, Method):
    LblU = np.unique(Label_training)
    if len(Test) == 1:
       Test = Test[0]
    if Method == 'quadratic':
       from sklearn.qda import QDA
       cls = QDA()
       cls.fit(Training, Label_training)
       classifier = cls.predict(Test)
       return classifier
    if Method == 'linear':
       from sklearn.lda import LDA
       cls = LDA()
       cls.fit(Training, Label_training)
       classifier = cls.predict(Test)
       return classifier
    if Method == 'diaglinear':
       classifier = np.zeros(len(LblU))
       covariance = np.zeros((len(Training.T), len(Training.T)))
       for i in range(len(LblU)):
          cls = Training[np.where(Label_training==LblU[i])[0],:]
          scat = (np.cov(cls.T)/len(cls))*(len(cls)-1)
          scat = np.identity(len(scat))*np.diag(scat)
          covariance += scat
          covariance = covariance / len(LblU)
       for classe in range(0, len(LblU)):
          cls = Training[np.where(Label_training==LblU[classe])[0],:]
          mu = np.mean(cls, 0)
          exponential = -0.5*np.dot(np.dot((Test-mu),np.linalg.inv(covariance)).T,
       (Test-mu))
```

```
            classifier[classe] = exponential -0.5*np.log(np.linalg.det(covariance))
        return LblU[classifier==max(classifier)][0]
    if Method == 'diagquadratic':
        classifier = np.zeros(len(LblU))
        for classe in range(0, len(LblU)):
            cls = Training[np.where(Label_training==LblU[classe])[0],:]
            mu = np.mean(cls, 0)
            covariance = (np.cov(cls.T)/len(cls))*(len(cls)-1)
            covariance = np.identity(len(covariance))*np.diag(covariance)
            prior = float(len(cls))/len(Training)
            exponential = -0.5*np.dot(np.dot((Test-mu),np.linalg.inv(covariance)).T,
        (Test-mu))
            classifier[classe] = exponential+np.log(prior) -
        0.5*np.log(np.linalg.det(covariance))
        return LblU[classifier==max(classifier)][0]
    if Method == 'mahalanobis':
        classifier = np.zeros(len(LblU))
        for classe in range(0, len(LblU)):
            cls = Training[np.where(Label_training==LblU[classe])[0],:]
            mu = np.mean(cls, 0)
            covariance = (np.cov(cls.T)/len(cls))*(len(cls)-1)
            distance = -0.5*np.dot(np.dot((Test-mu),np.linalg.inv(covariance)).T, (Test-mu))
            classifier[classe] = distance
        return LblU[classifier==max(classifier)][0]
```

Appendice B - Bayesian Classifier (C++ Code)

```
string QDAloocv(int Nsamp, double* Test, double** Training, int Ntest,
    string* Lbl_train, std::vector<string> LblU, int MetInd)
{
    double start_time=omp_get_wtime();
    omp_set_num_threads(omp_get_num_procs());
    omp_set_nested(1);
    long double* Discriminant = new long double[LblU.size()];
    long double* Determinant = new long double[LblU.size()];
    if ( MetInd == 0 || MetInd == 1 || MetInd == 2)
    {
        for(unsigned short int i=0; i<LblU.size(); i++)
        {
            Discriminant[i] = 0;
            std::vector<int> Nlabel;
            for(int j=0; j<Nsamp; j++)
            {
                if(Lbl_train[j] == LblU[i]) {Nlabel.push_back(j);}
             }
            long double* Mean = new long double[Ntest];
            if( MetInd == 0 || MetInd == 2)
            {
                for(unsigned short int j=0; j<Ntest; j++)
                {
                    Mean[j] = 0;
                    for(unsigned short int k=0; k<Nlabel.size(); k++)
                    {
                        Mean[j] += Training[Nlabel[k]][j] / Nlabel.size();
                    }
                }
                MatrixXd Covariance = MatrixXd::Zero(Ntest, Ntest);
```

```
            for(unsigned short int j=0; j<Nlabel.size(); j++)
            {
                for(unsigned short int k=0; k<Ntest; k++)
                {
                    for(unsigned short int v=0; v<Ntest; v++)
                    {
                        Covariance(k,v) += ((Training[Nlabel[j]][k] - Mean[k])*
                            (Training[Nlabel[j]][v] - Mean[v]) /(Nlabel.size()));
                    }
                }
            }
            MatrixXd Covariance_inv = Covariance.inverse();
            Determinant[i] = Covariance.determinant();
            for(unsigned short int j=0; j<Ntest; j++)
            {
                for(unsigned short int k=0; k<Ntest; k++)
                {
                    Discriminant[i] += ((Test[k] - Mean[k])*(Covariance_inv(k,j))*
                        (Test[j] - Mean[j]));
                }
            }
            Covariance.resize(0, 0);
            Covariance.resize(0, 0);
            delete Mean;
            if(MetInd == 0)
            {
                long double normalizer = 1. / (2*3.141592653589793*
                    sqrt(Determinant[i]));
                long double prior = float(Nlabel.size()) / (Nsamp);
                Discriminant[i] = normalizer * exp( - 0.5 * Discriminant[i] );
            }
            if(MetInd == 2)
            {
                Discriminant[i] = - 0.5 * Discriminant[i];
            }
        }
        if (MetInd == 1) // DiagQuadratic Classifier
        {
            long double* Mean_sq = new long double[Ntest];
            for(unsigned short int j=0; j<Ntest; j++)
            {
```

```
                Mean[j] = 0;
                Mean_sq[j] = 0;
                for(unsigned short int k=0; k<Nlabel.size(); k++)
                {
                    Mean[j] += Training[Nlabel[k]][j] / Nlabel.size();
                    Mean_sq[j] += (Training[Nlabel[k]][j]*Training[Nlabel[k]][j]
                        ) / Nlabel.size();
                }
            }
            for(unsigned short int j=0; j<Ntest; j++)
            {
                Discriminant[i] += ((Test[j]-Mean[j])*(1/(Mean_sq[j] -
                    Mean[j]*Mean[j]))*(Test[j]-Mean[j]));
                Determinant[i] *= (Mean_sq[j] - Mean[j]*Mean[j]);
            }
            delete Mean;
            delete Mean_sq;
            long double normalizer = 1. / (2*3.141592653589793*
                sqrt(Determinant[i]));
            long double prior = float(Nlabel.size()) / (Nsamp);
            Discriminant[i] = normalizer * exp(- 0.5 * Discriminant[i]) * prior;
        }
    }
    long double max_discriminant = - 100000000;
    string result;
    for(unsigned short int i=0; i<LblU.size(); i++)
    {
        if (Discriminant[i] > max_discriminant) { max_discriminant =
            Discriminant[i]; result = LblU[i];}
    }
    delete Discriminant;
    delete Determinant;
    return result;
}
if (MetInd == 3)
{
    long double** Mean = new long double*[Ntest];
    long double* Mean_sq = new long double[Ntest];
    MatrixXd Cov = MatrixXd::Zero(Ntest, Ntest);
    int* Nlabel = new int[LblU.size()];
    std::vector<int> lbl;
```

```
for(unsigned short int i=0; i<Ntest; i++)
{
    Mean[i] = new long double[LblU.size()];
    Mean_sq[i] = 0;
    for(unsigned short int j=0; j<Ntest; j++)
    {
        Mean[i][j] = 0;
    }
}
for(unsigned short int i=0; i<LblU.size(); i++)
{
    Nlabel[i] = 0;
    Discriminant[i] = 0;
    lbl.clear();
    for(int j=0; j<Nsamp; j++)
    {
        if(Lbl_train[j] == LblU[i]) {Nlabel[i] += 1; lbl.push_back(j);}
    }
    for(unsigned short int j=0; j<Ntest; j++)
    {
        for(unsigned short int k=0; k<lbl.size(); k++)
        {
            Mean[j][i] += Training[lbl[k]][j] / lbl.size();
            Mean_sq[j] += ( Training[lbl[k]][j]*
                Training[lbl[k]][j] ) / lbl.size();
        }
    }
    for(unsigned short int j=0; j<lbl.size(); j++)
    {
        for(unsigned short int k=0; k<Ntest; k++)
        {
            for(unsigned short int v=0; v<Ntest; v++)
            {
                Cov(k,v) += ((Training[lbl[j]][k] - Mean[k][i])*
                    (Training[lbl[j]][v]-Mean[v][i]) /(lbl.size()));
            }
        }
    }
}
MatrixXd Cov_inv = Cov.inverse();
for(unsigned short int i=0; i<LblU.size(); i++)
```

```
    {
        for(unsigned short int j=0; j<Ntest; j++)
        {
            for(unsigned short int k=0; k<Ntest; k++)
            {
                Discriminant[i] += ((Test[k] - Mean[k][i])*(Cov_inv(k,j))*
                    (Test[j] - Mean[j][i]));
            }
        }
        Determinant[i] = Cov.determinant();
        long double normalizer = 1. / (2*3.141592653589793 * sqrt(Determinant[i]));
        long double prior = float(Nlabel[i]) / (Nsamp);
        Discriminant[i] = normalizer * exp(- 0.5 * Discriminant[i]) * prior;
    }
    for(unsigned short int i=0; i<Ntest; i++)
    {
        delete Mean[i];
    }
    Cov.resize(0, 0);
    Cov_inv.resize(0, 0);
    delete Mean;
    delete Mean_sq;
    delete Nlabel;
    long double max_discriminant = - 100000000;
    string result;
    for(unsigned short int i=0; i<LblU.size(); i++)
    {
        if (Discriminant[i] > max_discriminant) { max_discriminant =
            Discriminant[i]; result = LblU[i];}
    }
    return result;
}
if (MetInd == 4)
{
    long double** Mean = new long double*[Ntest];
    long double* Mean_sq = new long double[Ntest];
    long double* cov_temp = new long double[Ntest];
    int* Nlabel = new int[LblU.size()];
    std::vector<int> lbl;
    for(unsigned short int i=0; i<Ntest; i++)
    {
```

```
    Mean[i] = new long double[LblU.size()];
    Mean_sq[i] = 0;
    cov_temp[i] = 0;
}
for(unsigned short int i=0; i<LblU.size(); i++)
{
    Nlabel[i] = 0;
    Discriminant[i] = 0;
    Determinant[i] = 1;
    lbl.clear();
    for(int j=0; j<Nsamp; j++)
    {
        if(Lbl_train[j] == LblU[i]) {Nlabel[i] += 1; lbl.push_back(j);}
    }
    for(unsigned short int j=0; j<Ntest; j++)
    {
        Mean[j][i] = 0;
        for(unsigned short int k=0; k<lbl.size(); k++)
        {
            Mean[j][i] += Training[lbl[k]][j] / lbl.size();
            Mean_sq[j] += (Training[lbl[k]][j]*Training[lbl[k]][j])/
                lbl.size();
        }
        cov_temp[j] += (Mean_sq[j] - Mean[j][i]*Mean[j][i])/LblU.size();
    }
}
for(unsigned short int i=0; i<LblU.size(); i++)
{
    for(unsigned short int j=0; j<Ntest; j++)
    {
        Discriminant[i] += ((Test[j]-Mean[j][i])*(1/cov_temp[j])*
            (Test[j]-Mean[j][i]));
        Determinant[i] *= cov_temp[j];
    }
    long double normalizer = 1. / (2*3.141592653589793 * sqrt(Determinant[i]));
    long double prior = float(Nlabel[i]) / (Nsamp);
    Discriminant[i] = normalizer * exp(- 0.5 * Discriminant[i]) * prior;
}
for(unsigned short int i=0; i<Ntest; i++)
{
    delete Mean[i];
```

```
        }
        delete Mean;
        delete Mean_sq;
        delete cov_temp;
        delete Nlabel;
        long double max_discriminant = - 100000000;
        string result;
        for(unsigned short int i=0; i<LblU.size(); i++)
        {
            if (Discriminant[i] > max_discriminant) { max_discriminant =
                Discriminant[i]; result = LblU[i];}
        }
        return result;
    }
}
```

Appendice C - Graph Analyses (C++ Code)

```
#include <boost/graph/adjacency_list.hpp>
#include <boost/graph/connected_components.hpp>
#include <boost/graph/betweenness_centrality.hpp>
#include <boost/graph/exterior_property.hpp>
#include <boost/graph/closeness_centrality.hpp>
#include <boost/graph/property_maps/constant_property_map.hpp>
#include <boost/graph/floyd_warshall_shortest.hpp>
#include <boost/graph/undirected_graph.hpp>
#include <boost/graph/clustering_coefficient.hpp>
/************************************
/* The Eigen include files         */
#include <Eigen/Core>
#include <Eigen/Eigenvalues>
#include <Eigen/QR>
//#include <unsupported/Eigen/MatrixFunctions>
/************************************/

using namespace std;
using namespace boost;
using namespace Eigen;

typedef property < edge_weight_t, double >Weight;
typedef adjacency_list < vecS, vecS, undirectedS, no_property, Weight > Graph;
typedef graph_traits<Graph>::vertex_descriptor Vertex;
typedef graph_traits<Graph>::edge_descriptor Edge;
typedef boost::exterior_vertex_property<Graph, float> ClosenessProperty;
typedef constant_property_map<Edge, int> WeightMap;
typedef ClosenessProperty::container_type ClosenessContainer;
typedef ClosenessProperty::map_type ClosenessMap;
typedef exterior_vertex_property<Graph, int> DistanceProperty;
```

```
typedef DistanceProperty::matrix_type DistanceMatrix;
typedef DistanceProperty::matrix_map_type DistanceMatrixMap;
typedef exterior_vertex_property<Graph, float> ClusteringProperty;
typedef ClusteringProperty::container_type ClusteringContainer;
typedef ClusteringProperty::map_type ClusteringMap;

VectorXd Connected_Components(MatrixXd A, int size)
{
Graph G(size);
for(int i=0; i<size; i++)
{
for(int j=0; j<size; j++)
{
if(A(i,j) == 1) {add_edge(i,j, G);}
}
}
std::vector<int> component(num_vertices(G));
num = connected_components(G, &component[0]); // numero totale di componenti
std::vector<int>::size_type ind_net;
set<int> Ncomp;
for(ind_net=0; ind_net!=component.size(); ind_net++)
{
Ncomp.insert(component[ind_net]);
Comp.push_back(component[ind_net]);
}
for(set<int>::iterator i = Ncomp.begin(); i!=Ncomp.end(); i++)
{
int s=0;
for(ind_net=0; ind_net!=component.size(); ind_net++)
{
if(*i==component[ind_net]) {s += 1;}
}
Size.push_back(s);
}
cout << "Network: "<< num_vertices(G) << " nodes " << num << " components"<<endl;
return Comp;
}

VectorXd Connectivity_IN(MatrixXd A, int size)
{
VectorXd k(size);
```

```
for(int i=0; i<size; i++)
{
k[i] = A.row(i).sum();
}
return k;
}

VectorXd Betweenness_centrality(MatrixXd A, int size)
{
VectorXd BC(size);
Graph G;
G.clear();
for(int i=0; i<size; i++)
{
for(int j=0; j<size; j++)
{
if(A(i,j) != 0) {add_edge(i,j,Weight(A(i,j)),G);}
}
}
boost::shared_array_property_map<double, boost::property_map<Graph,
    vertex_index_t>::const_type> centrality_map(num_vertices(G),
    get(boost::vertex_index, G));
brandes_betweenness_centrality(G, centrality_map);
double BC_max = ((size-1)*(size-2)) / 2; // Fattore di normalizzazione!!
for(int i=0; i<num_vertices(G); i++)
{
BC(i) = centrality_map[i] / BC_max;
}
G.clear();
return BC;
}

VectorXd Clustering_coefficient(MatrixXd A, int size)
{
VectorXd C(size);
Graph G;
G.clear();
for(int i=0; i<size; i++)
{
for(int j=0; j<size; j++)
{
```

```
if(A(i,j) != 0) {add_edge(i,j,Weight(A(i,j)),G);}
}
}
ClusteringContainer coefs(num_vertices(G));
ClusteringMap cm(coefs, G);
float cc = all_clustering_coefficients(G, cm);
for(int i=0; i<num_vertices(G); i++)
{
C(i) = cm[i];
}
G.clear();
return C;
}

MatrixXd Pendrem_Eigen(MatrixXd Adj, int size_adj)
{
MatrixXd AdjRem = Adj;
std::vector<int> RemInd;
VectorXd AdjK(size_adj);
for(int i=0; i<size_adj; i++)
{
RemInd.push_back(i);
AdjK[i] = AdjRem.col(i).sum();
}
cout << RemInd.size() << " nodes..."<<endl;
std::vector<int> PenInd;
int min_adjK = AdjK.minCoeff();
while(min_adjK == 1)
{
PenInd.clear();
for(int i=0; i<AdjK.size(); i++)
{
if(AdjK(i) == 1) {PenInd.push_back(i); }
}
cout << "found "<<PenInd.size()<< " pendants"<<endl;
int shift=0;
for(int i=0; i<PenInd.size(); i++)
{
RemInd.erase(RemInd.begin()+PenInd[i]-shift); shift++;
}
for(int i=0; i<PenInd.size(); i++)
```

```
{
removeColumn(AdjRem, PenInd[i]-i);
removeRow(AdjRem, PenInd[i]-i);
}
int size = AdjK.size()-PenInd.size();
AdjK.resize(size);
for(int i=0; i<size; i++)
{
AdjK[i] = AdjRem.col(i).sum();
}
if(AdjK.size() == 0)
{
cout << "Il network era una catena"<<endl;
return AdjRem;
}
min_adjK = AdjK.minCoeff();
}
cout << RemInd.size() << " non-pendant nodes\n";
return AdjRem;
}
```

Bibliografia

[1] Yuan Yuan, Eliezer M Van Allen, Larsson Omberg, Nikhil Wagle, Ali Amin-Mansour, Artem Sokolov, Lauren A Byers, Yanxun Xu, Kenneth R Hess, Lixia Diao, Leng Han, Xuelin Huang, Michael S Lawrence, John N Weinstein, Josh M Stuart, Gordon B Mills, Levi A Garraway, Adam A Margolin, Gad Getz & Han Liang, *Assessing the clinical utility of cancer genomic and proteomic data across tumor types*, Nature Reviews, Biotechology (2014).

[2] Eric E. Schadt, Michael D. Linderman, Jon Sorenson, Lawrence Lee and Garry P. Nolan, *Computational solutions to large-scale data management and analysis*, Nature Reviews, Genetics (2010).

[3] Casey S. Greene, Jie Tan, Matthew Ung, Jason H. Moore and Chao Cheng, *Big Data Bioinformatics*, J. Cell. Physiol. 229: 1896–1900, 2014.

[4] Oswaldo Trelles, Pjotr Prins, Marc Snir and Ritsert C. Jansen, *Big data, but are we ready?*, Nature Reviews, Genetics (2011).

[5] Vivien Marx, *The big challenges of Big Data*, Nature Reviews (2013).

[6] Tanya Barrett, Dennis B. Troup, Stephen E. Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F. Kim, Maxim Tomashevsky, Kimberly A. Marshall, Katherine H. Phillippy, Patti M. Sherman, Rolf N. Muertter, Michelle Holko, Oluwabukunmi Ayanbule, Andrey Yefanov and Alexandra Soboleva, *NCBI GEO: archive for functional genomics data sets-10 years on*, Nucleic Acids Research, Database issue (2011).

[7] C. L. Philip Chen, Chun-Yang Zhang, *Data-intensive applications, challenges, techniques and technologies: A survey on Big Data*, Information Sciences 275 (2014).

[8] Marco Vassura, Luciano Margara, Pietro di Lena, Filippo Medri, Piero Fariselli and Rita Casadio, *FT-COMAR: fault tolerant three-dimensional structure reconstruction from protein contact maps*, Bioinformatics Applications Note (2008).

[9] E Martínez, K Yoshihara, H Kim, GM Mills, V Treviño and RGW Verhaak, *Comparison of gene expression patterns across 12 tumor types identifies a cancer supercluster characterized by TP53 mutations and cell cycle defects*, Oncogene (2015).

[10] *Bioinformatics clouds for big data manipulation*, Biology Direct (2012).

[11] Hirak Kashyap, Hasin Afzal Ahmed, Nazrul Hoque, Swarup Roy, and Dhruba Kumar Bhattacharyya, *Big Data Analytics in Bioinformatics: A Machine Learning Perspective*, arXiv:1506.05101v1 (2014).

[12] Mohsen Rezaei, *A diverse Clustering Method on Biological Big Data*, International Journal of Automation, Control and Intelligent Systems (2015).

[13] Divya Kumari, Ravi Kumar, *Impact of Biological Big Data in Bioinformatics*, International Journal of Computer Applications (0975-8887) (2014).

[14] Alanin Barrat, Marc Bartélemy, Alessandro Vespignani, *Dynamical Processes on Complex Networks*, Cambridge University Press (2008)

[15] Guido Caldarelli, Alessandro Vespignani, *Large Scale Structure and Dynamics of Complex Networks, from information technology to finance and natural science*, World Scientific Publishing Co. Pte. Ltd. (2007)

[16] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer Science+Business Media, LLC (2006)

[17] Nils J. Nilsson, *Introduction to Machine Learning*, Department of Computer Science, Stanford University (1998)

[18] Jussi Tohka, *Introduction to Pattern Recognition*, Department of Signal Processing, Tampere University of Technology (2013)

[19] Alex Smola, S. V. N. Vishwanathan, *Introduction to Machine Learning*, Cambridge University Press (2008)

[20] Ricardo Gutierrez-Osuna, *Pattern Analysis*

[21] Anil K. Jail, Robert P. W. Duin, Jianchang Mao, *Statistical Pattern Recognition: A Review*, IEEE Transactions on pattern analysis and machine intelligence (2000)

[22] Dott. Davide Maltoni, *Classificazione Supervisionata*, Scienze dell'informazione, Università di Bologna

[23] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerial Recepies, The Art of Scientific Computing (Third Edition)*, Cambridge University Press (2007)

[24] Aric Hagberg, Dan Schult, Pieter Swart, *NetworkX Reference - Release 1.9.1*, September 20, 2014

[25] Scikit-learn developers, *Scikit-learn user guide - Release 0.12-git*, June 04, 2012

[26] Office of the Privacy Commissioner of Canada, *Fiche d'information*, Commissariat à la protection de la vie privée du Canada

[27] F. Emmert-Strib and M. Dehmer, *Analysis of Microarray Data: A Network-Based Approach*, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim (2008)

[28] Amos J. Storkey, *Machine Learning and Pattern Recognition - Principal Component Analysis*, Institute for Adaptive and Neural Computation, School of Informatics, University of Edinburgh (2004).

[29] Hiroshi Shimodaira, *Discriminant Functions*, Learning and Data Note 10 (2015).

[30] Bojun Tu, Zhihua Zhang, Shusen Wang, Hui Qian, *Making Fisher Discriminant Analysis Scalable*, Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, (2014)