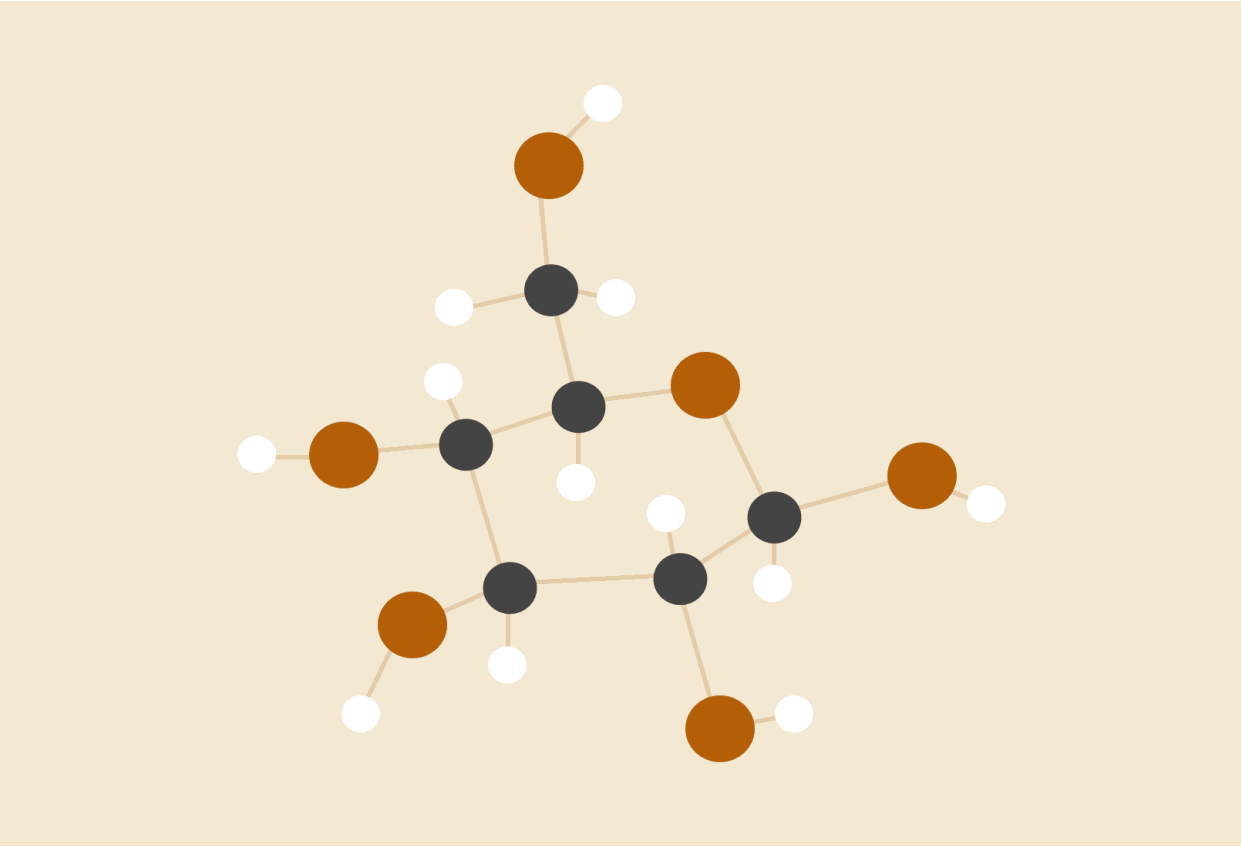
PROGETTO FISICA DEI SISTEMI COMPLESSI

Modello predittivo per pazienti affetti da cancro



Andrea Ercolessi

Alessio Portaro

Francesco Antimi

Laurea Magistrale Informatica (Curricula A-C) A.A. 2019-2020

INDICE

INTRODUZIONE

Il lavoro svolto ricalca quanto descritto nell'articolo Simulation-assisted machine learning. Lo scopo del progetto riguarda la costruzione di un **modello predittivo** per stabilire una classificazione dei pazienti sospetti di essere affetti da cancro colon-rettale. Un modello di questo tipo si può dimostrare utile in ambito medico, come strumento di supporto alle decisioni per il personale sanitario che prende in cura il paziente.

Solitamente gli approcci per risolvere questo problema possono essere suddivisi in **due categorie**.
Il primo modo possibile per approcciare questo problema potrebbe essere quello di applicare tecniche di modellazione classiche dell'Informatica. Infatti, poiché il problema si pone come lo studio di un modello di classificazione, si potrebbe pensare di applicare degli **algoritmi di machine learning** che solitamente vengono utilizzati per l'analisi e la classificazione di campioni di dati.
Uno dei problemi principali di questo approccio riguarda però la scelta delle caratteristiche che permettano di costruire il modello che descrive il dominio nel modo più preciso. Descrivere i pazienti con poche caratteristiche macroscopiche potrebbe infatti rendere il modello risultante inutile ai fine dello studio.
D'altra parte riportare troppe caratteristiche e/o troppo microscopiche potrebbe rendere la creazione del modello impraticabile, per via delle difficoltà nel misurare tali caratteristiche e nella complessità di trattarle algoritmicamente.

Un secondo possibile approccio potrebbe essere quello più tipico nell'ambito della ricerca medica. In questo campo si è soliti infatti approcciare il problema andando a ricercare i meccanismi che regolano i processi studiati ed andando poi ad ideare strumenti precisi e specifici che siano in grado di definire al meglio la malattia. L'idea in questo caso è quindi di creare dei **modelli di simulazione** che riescano a riprodurre in maniera precisa il processo sotto esame.
In generale però per un approccio di questo tipo le problematiche sono varie.
In primo luogo i processi biologici sono tipicamente dei sistemi complessi e in quanto tali risulta difficile descrivere le interazioni tra le parti in gioco.
Una seconda problematica potrebbe riguardare poi le tempistiche necessarie allo studio e l'ideazione di modelli con una tale specificità e precisione come sono quelli descritti sopra.

In ultimo, ma non per importanza, è necessario tenere in considerazione la questione dell'enorme dimensione che è intrinseca ai dati di tipo genetico, necessari a descrivere in maniera meticolosa i processi in questione.
Per via di questa loro enorme dimensione tali dati ricadono infatti spesso sotto la classificazione di ***Big Data***.

Considerato quanto detto finora, l'approccio scelto per questo studio cerca perciò di sfruttare tutte le caratteristiche positive degli approcci di cui sopra e di mitigare le loro problematiche.
Il metodo studiato è incentrato su **metodi di machine learning basati sul kernel**.
Questi sono degli algoritmi per la costruzione di **modelli di classificazione** che permettono di trattare agevolmente vettori con molte caratteristiche. Essi, infatti, non necessitano di eseguire operazioni sui vettori nella loro interezza ma hanno semplicemente bisogno di indici di similarità tra le coppie degli stessi vettori.
In questo modo si cerca di ovviare al **problema della dimensione** dello spazio di ricerca e allo stesso tempo, calcolando tali indici di similarità su vettori molto significativi per ogni paziente, per costruzione, si cerca di minimizzare il problema della significatività dei vettori. Tali vettori sono infatti ottenuti a partire da **simulazioni su dati reali** che cercano di modellare, in maniera precisa ma comunque in tempi computazionalmente ragionevoli, i processi biologici studiati.

STRUMENTI UTILIZZATI

1.  **MaBoSS (Markovian Boolean Stochastic Simulator):**
   Software per la simulazione di reti booleane, utilizzato per identificare gli attrattori di tali reti e la probabilità che il sistema evolva fino ad raggiungere l'attrattore stesso.
2.  **MPI (Message Passing Interface):**
   Standard de facto per la programmazione parallela secondo il paradigma message passing. Utilizzato per l'esecuzione distribuita delle varie simulazioni relative ai pazienti. È stata utilizzata l'implementazione nel linguaggio C++ denominata MpiCH.
3.  **NumPy :**
   Libreria Python per la gestione e l'elaborazione di dati in formato vettoriale e matriciale. In particolare mette a disposizione operazioni ottimizzate per i dati di grandi dimensioni.
4.  **Pandas :**
   Libreria per l'analisi dati. Mette a disposizione strutture dati e metodi per la manipolazione dei dati in serie temporali e dati strutturati in tabelle.
5.  **Scikit-Learn :**
   Libreria che implementa gli algoritmi di machine learning che utilizzano kernel.
6.  **MatPlotLib :**
   Libreria per la presentazione grafica dei risultati, compatibile con NumPy e le relative strutture dati.

CONCETTI PROPEDEUTICI

Gene Regulatory Networks

Le **Gene Regulatory Networks** (GRN) sono degli insiemi di geni, proteine ed altri fattori cellulari che interagiscono tra loro tramite processi chimici e fisici.
Le GRN svolgono un ruolo centrale in fase di morfogenesi dei tessuti all'interno degli organismi animali, e possiedono pertanto un ruolo centrale anche in fase di creazione di cellule malate o tumorali.
Questi oggetti vengono anche chiamati **reti** poiché spesso vengono modellati con delle **strutture a grafo**, dove i geni e le proteine che interagiscono rappresentano i nodi mentre le loro interazioni rappresentano gli archi.

Per poter costruire dei modelli di queste reti è necessario analizzare quelli che vengono detti ***Dati di espressione genetica***, dove questi sono dati che indicano la quantità in cui determinati geni o proteine sono presenti all'interno di una cellula sotto analisi.
Per poter determinare questi dati sono state ideate varie tecnologie.
Tra queste la più utilizzata in Biologia è nota con il nome di **Microarray**.
A partire da questi dati, attraverso diversi algoritmi si cerca di costruire dei **modelli che approssimino le GRN**. I principali paradigmi per la modellazione di tali reti sono tre :

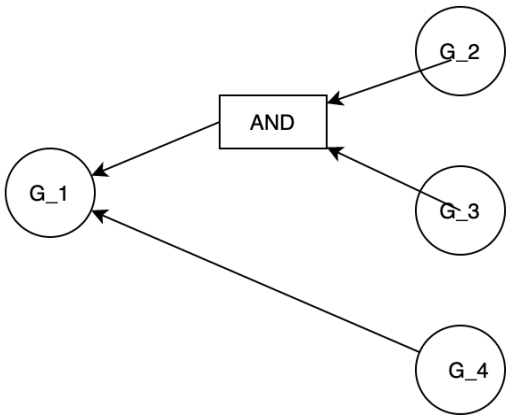
1. Boolean Networks
2. Bayesian Networks
3. Sistemi di equazioni differenziali

Tra questi, il modello più semplice è quello delle **Boolean Networks**.
Tuttavia nonostante la sua semplicità, questo modello è quello che si presta meglio alle necessità dello studio e che si è pertanto deciso di utilizzare.

Boolean Networks

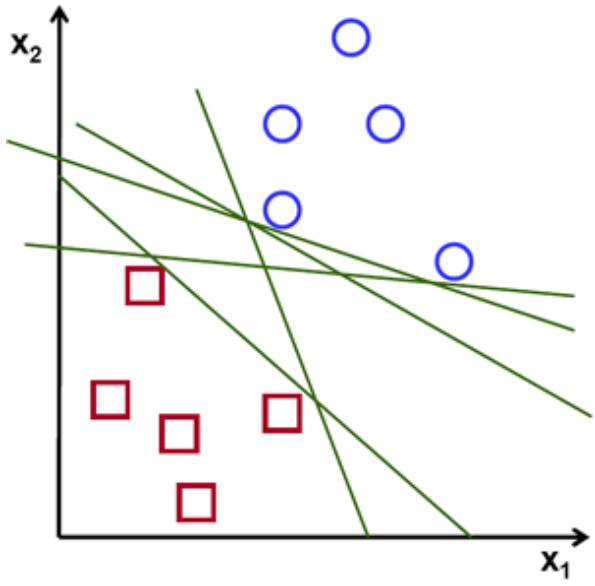
Una Boolean Network è un tipo di rete che ad ogni gene fa corrispondere un nodo e i cui archi che li collegano sono **regole booleane**. Un esempio di regola booleana è visibile nell'immagine sottostante.

$$G_1 \leftarrow (G_2\ AND\ G_3)\ OR\ G_4$$



Una regola booleana risulta sempre composta da un gene che si trova in testa(G_1) e da un corpo ($(G_2\ AND\ G_3)\ OR\ G_4$), ovvero una proposizione booleana che prende come argomenti altri geni posti in correlazione col gene di testa.
Lo scopo di una rete di questo tipo è quello di descrivere l'evoluzione del sistema nel tempo tramite l'utilizzo di una regola specifica per ogni gene che lo mette in testa.
Ognuna di queste regole viene istanziata a partire dallo stato del sistema al tempo *t*, dove per stato si intende l'insieme dei valori nel dominio booleano che ogni gene nel sistema assume.
Dall'applicazione di tutte le regole, i valori dei geni che si vengono a trovare in testa rappresentano così lo stato del sistema in un tempo successivo *t+1* e permettono quindi di vederne l'evoluzione.
Lo stato iniziale del sistema viene stabilito a partire dallo studio dei dati di espressione genetica e nel caso in cui il gene risulti presente all'interno della cellula in quantità maggiori rispetto ad un certo parametro soglia S precedentemente fissato, gli viene assegnato il valore 1, altrimenti 0. Ciò che per noi risulta più interessante studiare all'interno di una rete booleana sono i cosiddetti **attrattori**, vale a dire una traiettoria nell'evoluzione dello stato del sistema che, passando attraverso una serie di transizioni di stato successive, ritorna alla fine ad uno stato già visitato risultando quindi in un ciclo.
Dato che le Boolean Networks sono reti con un numero finito di stati è chiaro quindi come sia sempre possibile trovare degli attrattori, in quanto una traiettoria prima o poi finirà per forza di cose su uno stato già visitato.
È bene però fare una precisazione, ovvero che quanto detto si riferisce unicamente a reti che presentano un'evoluzione deterministica e non influenzata da fattori stocastici.

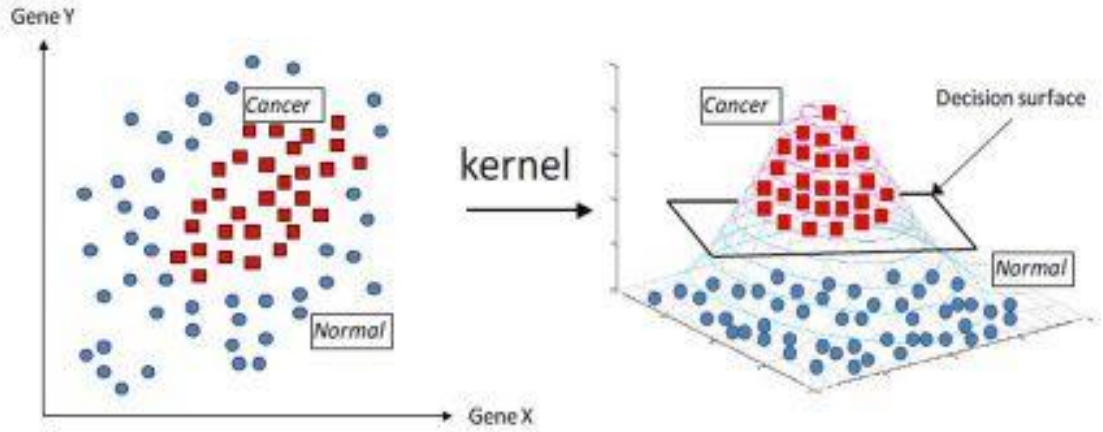
Machine Learning: kernel methods



Come già anticipato nell'introduzione, nel progetto si è scelto di utilizzare un algoritmo di machine learning per la creazione di un modello di classificazione.
Un modello di questo tipo ci permette infatti di rappresentare i dati in input come punti in uno spazio e, una volta trovato un pattern che consenta di suddividerli in classi, di identificare in tempo costante la classe a cui appartiene un nuovo dato qualsiasi.
Nel nostro specifico caso ogni punto rappresenta un paziente mentre le classi il loro stato di salute (ovvero se presentano o meno la patologia in studio).
Ogni paziente, tuttavia, viene identificato da un numero elevato di caratteristiche, il che si traduce in una dimensionalità dello spazio molto grande e in un carico computazionale significativo. Per ovviare al problema si è quindi scelto di ricorrere ai cosiddetti **kernel methods**.
Tali metodi si basano sulla costruzione di una *matrice kernel* (una matrice quadrata simmetrica con diagonale pari a 1) di dimensione *NxN* dove *N* è il numero di pazienti.
Il valore contenuto in ciascuna cella di indice *(i,j)* rappresenta la similarità tra un generico paziente *i* ed un altro paziente *j*.
In questo modo l'algoritmo permette di considerare unicamente i valori scalari di similarità senza dover tenere conto dei vettori delle caratteristiche dei pazienti nella loro interezza, andando così a ridurre notevolmente la dimensione e la complessità del problema.
Affinché questo sia possibile la matrice deve rispettare alcune proprietà matematiche definite dal **Teorema di Mercer:**

$$\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j)\, c_i\, c_j \geq 0$$

dove $k(x_i, x_j)$ rappresenta l'indice di similarità tra il paziente *i* e il paziente *j* mentre c_i $c_j$sono due valori reali qualunque.


Gene Y
Cancer
Normal
Gene X
kernel
Cancer
Decision surface
Normal

IMPLEMENTAZIONE

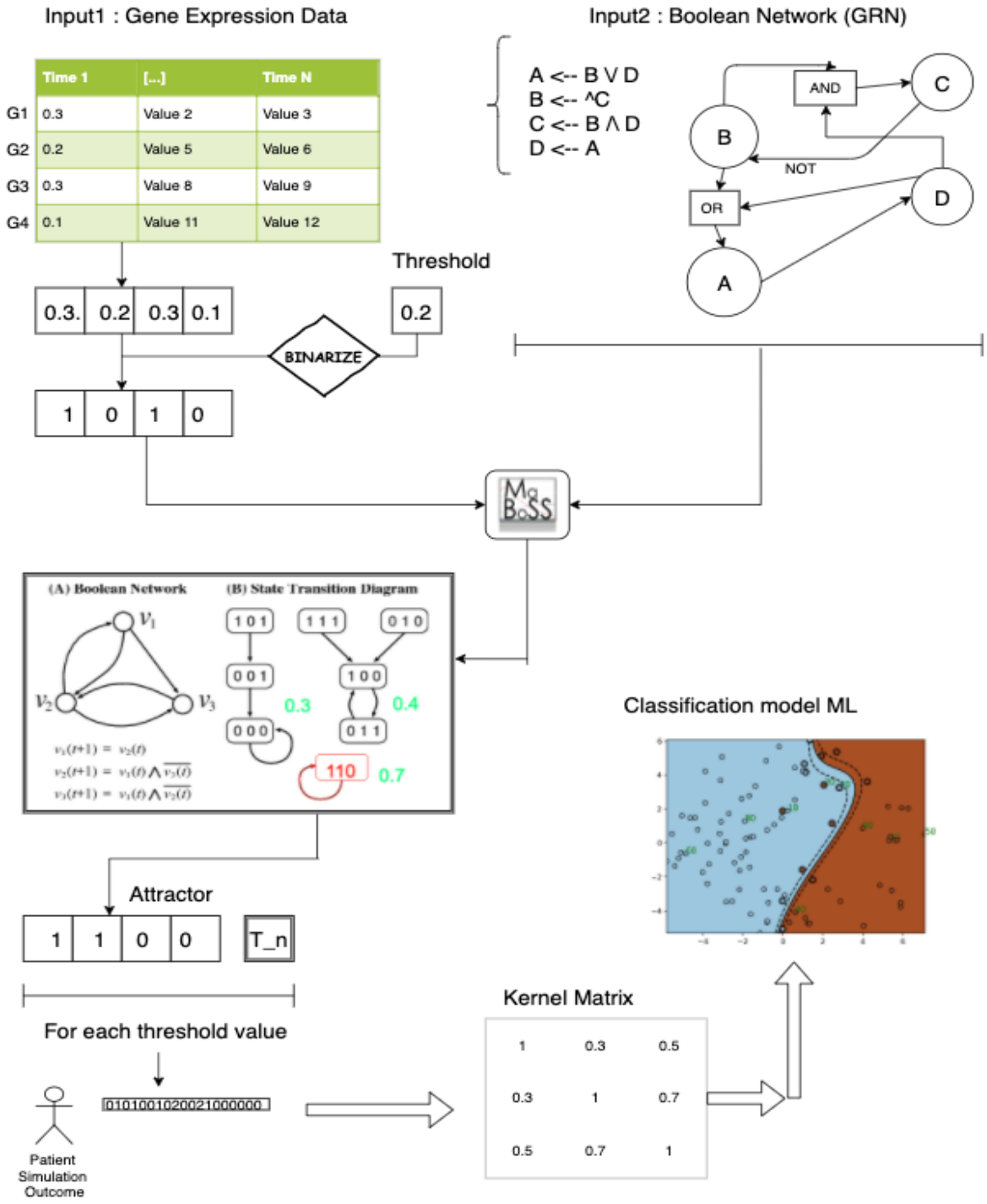


Diagramma di Flusso dell'architettura software.

Prima fase: simulazione rete booleana

Il software implementato si compone di **due macro-componenti**, delle quali la prima si occupa di effettuare delle **simulazioni** che approssimano **l'evoluzione nel tempo del sistema di geni** caratterizzanti la malattia di un paziente.

Per fare questo, il software ha bisogno di **due strutture dati in input**.
La prima di queste è una tabella contenente i **dati di espressione genetica** di ogni paziente. All'interno di questa tabella, ogni riga rappresenta l'evoluzione nel tempo di un singolo gene all'interno del sistema; ogni colonna corrisponde invece allo stato complessivo del sistema ad un dato istante temporale.

La seconda struttura dati è una GRN [vedi **Concetti Propedeutici**] rappresentata come **rete booleana**. L'inferenza di tale rete a partire da dati reali è stata effettuata da alcuni ricercatori, che hanno pubblicato i risultati ottenuti nell'articolo [**2**], ed è proprio a partire dalla stessa rete riportata in questo articolo che è stata strutturata la nostra simulazione.
Si tratta infatti di una che rete modella le **interazioni che avvengono tra i principali geni** coinvolti nel processo di evoluzione del cancro colon-rettale, ovvero il nostro caso di studio.

A partire da questi dati in input, è possibile **studiare l'evoluzione della malattia** di ogni paziente e capirne **le similarità** rispetto ad altri casi affini.
Per fare ciò si è deciso di utilizzare un framework sviluppato da sviluppatori terzi [vedi **Strumenti Utilizzati**] che prende il nome di **MaBoSS** (**M**arkovian **B**oolean **S**tochastic **S**imulator).

Tale simulatore si occupa di **analizzare le traiettorie** della rete booleana in input e **calcolare gli attrattori** in tali traiettorie indicando per ognuno di questi un indice rappresentante la probabilità che il sistema entri nel corrispondente attrattore.
Per determinare queste traiettorie il framework ha bisogno di uno **stato iniziale** che viene calcolato a partire da una colonna della tabella contenente i dati di espressione genetica. E come detto, tale colonna rappresenta lo stato del sistema ad un certo istante, ma i valori contenuti in essa rappresentano la percentuale di espressione di ogni gene. Una rete booleana tuttavia lavora con dati binari, e pertanto si è reso necessario **discretizzare i dati in input**. Per fare questo, è sufficiente, dato un certo valore soglia ***Th*** iterare il vettore colonna e settare al valore 1 ogni elemento che supera il valore ***Th***, mentre ogni altro valore viene posto a 0.

Alla fine di questa fase di simulazione per ogni paziente verrà costruito **un vettore che descrive il comportamento del paziente** relativo alla semplice malattia che segue la sua naturale evoluzione oppure in relazione ad un dato trattamento farmacologico (questo dipende da ciò che i dati di espressione genetica in input rappresentano).
Il vettore ha dimensione pari a:

$$Dim = (\#OutGenes + 1) * \#ThresholdValues$$

Il primo elemento nella formula rappresenta la cardinalità dell'insieme *OutGenes o*vvero i geni di output. Tale insieme è composto da alcuni geni che vengono appositamente presi in esame in quanto indicatori correlati ad eventi macroscopici rilevanti nell'evoluzione della malattia. Per fare un esempio, nel caso in esame del cancro colon-rettale, viene definito gene di output il gene **CASP9** che è conosciuto in letteratura per essere notoriamente un biomarker per il fenomeno dell'***Apoptosi.*** Il fenomeno dell'Apoptosi è particolarmente rilevante nell'evoluzione di patologie legate al cancro in quanto è il fenomeno per il quale una cellula decide di programmare la sua morte in modo da non infettare le cellule adiacenti. Nel modello vengono quindi definiti geni di output i seguenti:

$$OutGenes = \{Metastasis, Migration, Invasion, EMT, Apoptosis, CellCycleArrest\}$$

Dallo studio degli attrattori della rete booleana possiamo poi capire quando il sistema simulato entra in uno stato in cui il valore di attivazione dei geni di output rimarrà per sempre invariato. A questo punto sarà perciò inutile proseguire oltre con la simulazione e si potrà procedere ad analizzare **lo stato finale del sistema** corrispondente ad una condizione in cui il paziente si è stabilizzato. Viene inoltre presa nota dell'istante di tempo in cui questo stato stabile viene raggiunto e per questo motivo nella formula sovrastante compare un ***+1***.
Il secondo elemento che compare nella formula è invece la cardinalità dell'insieme *ThresholdValues*, che comprende tutti i valori assegnati al valore soglia ***Th.***
Per poter studiare meglio, infatti, quanto due pazienti si comportino in modo simile, il nostro studio ne ha analizzato **il comportamento al variare del valore soglia**, e in questo modo è possibile calcolare in modo più preciso il valore di similarità tra due pazienti che comparirà in seguito nella matrice Kernel.
I possibili valori assegnati alla variabile soglia sono:

$$Th = \{0.1;\ 0.2;\ 0.3;\ 0.4;\ 0.5;\ 0.6;\ 0.7;\ 0.8;\ 0.9\}$$

Alla luce di quanto detto, il vettore di output per un singolo paziente risulterà perciò strutturato come segue:

G1	G2	G3	G4	G5	G6	T	...	G1	G2	G3	G4	G5	G6	T
S1								S9						

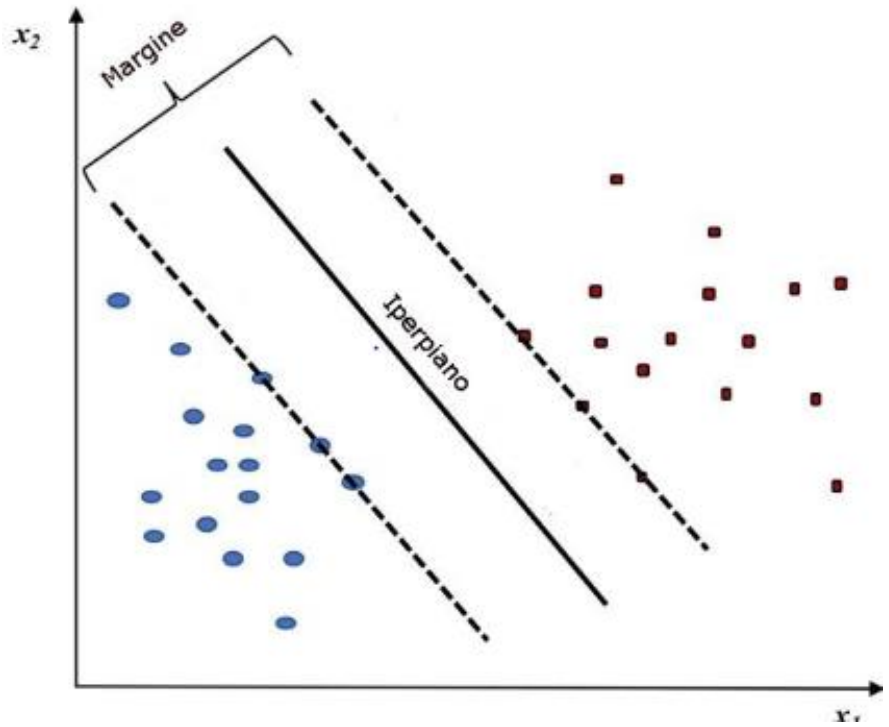
Per ogni valore soglia *S_n* vengono riportati i valori che i geni di output assumono nell'istante di tempo *T* nel quale il sistema entra nell'attrattore seguiti dal valore *T* stesso. In coda a questi valori si trovano poi i seguenti ***(#OutGenes + 1)*** valori relativi al valore soglia successivo.

Dal momento che questa prima componente del software implementato si occupa di simulare un sistema biologico, è facile capire come la complessità temporale sia elevata. Infatti, al crescere della dimensione della rete da simulare e del numero di campioni da analizzare, il tempo necessario rischia di diventare velocemente intrattabile.

Per mitigare questo aspetto si è pensato quindi di utilizzare strumenti di **calcolo parallelo** per diminuire di un fattore costante i tempi di calcolo, spostando la computazione su più macchine a loro volta composte di più unità di calcolo.
La tecnologia che si è scelto di utilizzare per fare ciò è ***MPI***. Questa tecnologia rappresenta lo standard de facto per la programmazione parallela con paradigma message passing e, tra le sue implementazioni, è stata utilizzata quella denominata ***MPICH***.
Nello specifico, per ridurre i tempi di esecuzione, è stato implementato un programma ***MAIN*** sequenziale, scritto in linguaggio ***C++***, che poi si occupa di avviare più istanze eseguite in parallelo del software di simulazione, scritto nel linguaggio ***Python***.

```
35      MPI_Init(&argc, &argv);
36
37      // Get the number of processes
38      int world_size;
39      MPI_Comm_size(MPI_COMM_WORLD, &world_size);
40
41      // Get the rank of the process
42      int world_rank;
43      MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
44
45      //Get patients Expression data
46      patients = (patients/world_size)*world_size;
47
48      //Start parallel simulations
49      string simulate = "python main.py -num=" + to_string(patients/world_size);
50      simulate = simulate + " -start=" + to_string((patients/world_size)*world_rank + first_line);
51      if(world_rank==0)
52          simulate = simulate + "-master";
53      system(simulate.c_str());
54      //Finalize
55      MPI_Finalize();
```

Seconda fase : Creazione del modello di ML



Come illustrato precedentemente, per creare il modello di classificazione dei pazienti si è scelto di utilizzare un **algoritmo di tipo SVM (Support-Vector Machines)**. Questo tipo di algoritmo rappresenta i pazienti come dei punti in uno spazio di ***N*** dimensioni. Per definire le classi di appartenenza dei punti, l'algoritmo cerca di costruire in tale spazio uno o più iperpiani che lo dividono in sottospazi, ognuno dei quali contiene tutti i punti di una stessa classe. Per definire tali iperpiani l'algoritmo utilizza i dati noti in fase di training per i quali è già a conoscenza delle classi di appartenenza.

In seguito, per ogni punto, viene creato il cosiddetto ***Vettore Di Supporto,*** che è un vettore con modulo pari alla distanza del punto rispetto all'iperpiano e con segno concorde alla classe di appartenenza del punto.

La questione a questo punto si riconduce ad un **problema di ottimizzazione** dove l'obiettivo è quello di rendere il maggior numero dei vettori di supporto concordi alla classe di appartenenza del punto che è noto a priori, andando a variare i parametri secondo i quali è definito l'iperpiano.

Come è evidente, quindi, l'algoritmo cerca di costruire una **funzione di regressione lineare** rappresentata dall'iperpiano.
Chiaramente però questa operazione non risulta essere sempre possibile, e per ovviare a questo problema esistono sostanzialmente due soluzioni.
La prima soluzione è quella di rendere ammissibile la classificazione errata di un numero esiguo di punti detti ***Outlier*** in modo tale che l'algoritmo, nel caso in cui non riesca a costruire una funzione di regressione che faccia cadere ogni punto nella classe di appartenenza, tenti comunque di costruire un modello lineare il più coerente possibile.
La seconda soluzione invece è quella di mappare i punti del nostro spazio **N** dimensionale in uno spazio **N+1** dimensionale nel quale i punti potrebbero essere divisi da un iperpiano lineare tramite lo stesso procedimento descritto in precedenza.
L'operazione di mapping dei punti tra i due spazi, tuttavia, non sempre è computazionalmente banale ma potrebbe essere anche molto dispendiosa. A questo proposito ci vengono in aiuto i **metodi kernel**. Questi, infatti, ci permettono di poter evitare il mapping dei punti a patto di conoscere un indice di similarità tra ogni coppia di vettori nello spazio ***N+1*** dimensionale.

Nel nostro caso, la similarità tra due pazienti da classificare viene calcolata sulla base delle righe corrispondenti ai due pazienti nella matrice ***M*** che viene elaborata dalla componente 1 del framework sviluppato, descritta in precedenza. Una volta calcolata la similarità per ogni coppia di pazienti otterremo la nostra **matric**e ***Kernel K.***
Sempre a partire dall'analisi della riga corrispondente ad un paziente nella matrice ***M,*** viene poi calcolato anche il valore di outcome che definisce la **classe di appartenenza del paziente**.
In particolare, in base al significato che hanno i geni di output, è in questo modo possibile individuare lo **stato di salute del paziente**. I valori di outcome per ogni paziente vengono poi raccolti in vettore denominato ***Y*** che verrà utilizzato per costruire il modello di machine learning.

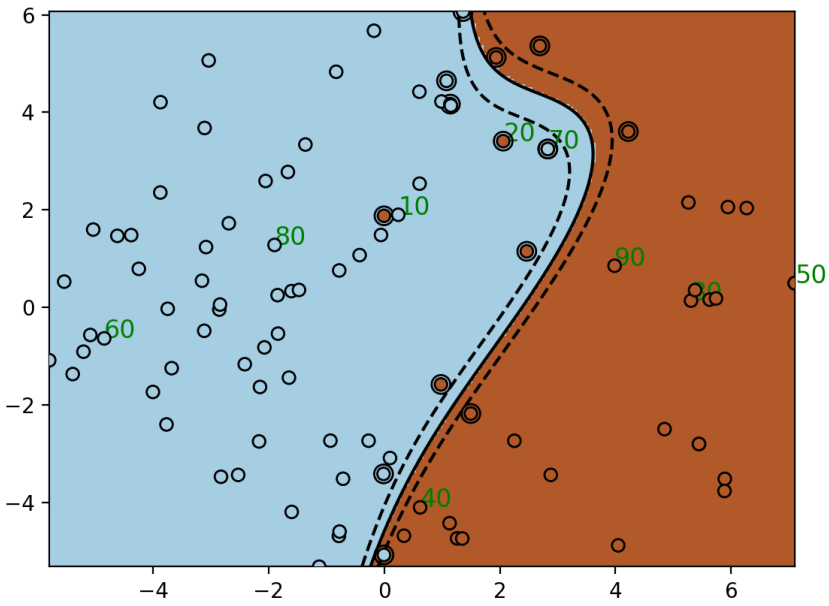
Una volta calcolate tutte le componenti necessarie al training del modello di classificazione, si può poi procedere con la fase di applicazione dell'algoritmo di machine learning vero e proprio.
Per fare questo, nel nostro studio è stato utilizzato il framework per il linguaggio Python denominato ***Scikit-Learn,*** che fornisce all'utilizzatore l'algoritmo SVM descritto in precedenza già implementato.

```
49  FILE = "results.csv"
50
51  res = pd.read_csv(FILE,header=None)
52
53  noY = res.iloc[0].to_numpy()
54
55  for row in range(1,len(res.index)):
56      current = res.iloc[row].to_numpy()
57      noY = np.vstack((noY,current))
58
59  Y = np.zeros(noY.shape[0])
60
61  outcome_calc = ANALIZE_PATIENT_OUTCOME
62
63  for i in range(0,noY.shape[0]):
64      Y[i] = outcome_calc(noY[i])
65
66  xY = np.column_stack((noY,Y))
67
68
69  MODEL = svm.SVC(kernel=simKernel)
70  MODEL.fit(noY,Y)
71
72  ######################__MODEL_CREATION__######################
```

Nel codice riportato sopra gli elementi più importanti e per cui vale la pena spendere due parole sono:

1. ***ANALIZE_PATIENT_OUTCOME***: questa funzione si occupa di calcolare il valore di Outcome che verrà utilizzato nella classificazione, analizzando la riga della matrice ***M*** relativa ad un singolo paziente;
2. ***SIMKERNEL*:** funzione che prende in input due righe della matrice M corrispondenti a due pazienti e ne calcola la similarità;
3. ***SVM.SVC***: si tratta della funzione che costruisce il modello di classificazione che utilizza l'algoritmo SVM sfruttando la funzione ***simKernel*** definita precedentemente. Tale funzione è già definita nel pacchetto ***Scikit-Learn***;
4. ***FIT***: funzione che prende in input la funzione degli Outcome ***Y*** e sfruttando la funzione Kernel assegnata precedentemente si occupa di effettuare l'operazione di training del modello.

Visualizzazione del classificatore ottenuto allenato su 100 pazienti



Terza fase: utilizzo

```
 98   NEW_PATIENT = user_input()
 99   OUTCOME = MODEL.predict(NEW_PATIENT)
100   PLOT.scatter(NEW_PATIENT,OUTCOME)
101   PLOT.show()
```

Come già detto in precedenza, lo strumento software implementato è da intendersi come un **tool di supporto alle decisioni del personale medico**.

Per questo motivo è stata implementata la possibilità da parte di un utente di inserire i dati relativi ad un nuovo paziente, in modo da ottenere dal modello una **previsione della futura evoluzione dello stato di salute del paziente**.

Tramite la funzione ***user_input*** l'utente ha quindi la possibilità quindi di inserire i dati relativi ad un nuovo paziente. Tale metodo si occuperà quindi di elaborare e di riscrivere i dati in un formato adatto al classificatore.

Una volta poi trasformato il vettore nel formato corretto, il metodo ***predict***, già implementato all'interno delle librerie ***Scikit-Learn***, permetterà di calcolare un valore di outcome per quel paziente.

Infine, il tool restituirà una **rappresentazione grafica del classificatore** ed il **posizionamento del paziente** nello spazio del classificatore stesso.

CONCLUSIONI

È evidente come uno strumento software di questo tipo potrebbe risultare molto d'aiuto al personale medico sanitario per velocizzare e rendere più efficiente il trattamento di alcune particolari patologie. Tuttavia, affinché il tool sia realmente utile, è necessario che questo sia **preciso** e fornisca **informazioni attendibili** e in un formato che risulti **comprensibile** all'utilizzatore finale, anche nel caso in cui esso non sia un esperto in ambito informatico.
Per fare un esempio noto riguardante questa problematica, basti pensare a quanto l'utilizzo di reti neurali artificiali faccia fatica ad essere ampiamente adottato in campo medico, poiché la semantica dietro al ragionamento di una ***ANN*** non è sempre chiara, e pertanto il personale medico è solitamente più restio ad affidarsi ad uno strumento di questo tipo.

Un altro aspetto rilevante nell'implementazione del tool riguarda poi i **tempi di risposta** necessari per elaborare una richiesta effettuata dall'utilizzatore finale.
A tal proposito la difficoltà maggiore che si incontra nello sviluppo di tali strumenti è senza dubbio la nota complessità spaziale e temporale nel simulare sistemi biologici.
Per mitigare questa problematica, nel nostro caso, si è pensato di utilizzare metodologie orientate al calcolo parallelo e distribuito e potrebbe perciò risultare utile in futuro esplorare maggiormente tutte le varie possibilità che si aprono in questo senso.

Infine, durante il nostro lavoro, è emersa una grossa problematica relativa alla **reperibilità ed interpretazione dei dati relativi alle patologie**.
È apparso evidente, infatti, come sia difficile trovare delle basi di dati ben strutturate mantenute da enti pubblici ufficiali dalle quali reperire dati per i propri studi.
Oltre a questo, inoltre, i pochi dati che si riescono a reperire risultano essere difficilmente interpretabili, in quanto mancano degli standard ufficialmente riconosciuti ed approvati dalla comunità scientifica per la rappresentazione degli stessi.
Ciò significa pertanto che, finché l'interpretazione dei dati rappresenterà una sfida così ardua, sarà sempre più necessario che i team di ricerca siano composti da persone specializzate in diversi campi. Sebbene un esperto in Computer Science sia capace di sviluppare strumenti software efficienti, dovrà infatti sempre essere affiancato da degli esperti che abbiano maggior conoscenza del dominio e che lo indirizzino nella comprensione e modellazione del problema da risolvere.

Alla luce di quanto detto, quindi, l'assenza di esperti in campo genetico all'interno del nostro team potrebbe aver reso il lavoro svolto poco significativo dal punto di vista dell'affidabilità dello strumento implementato. Tuttavia, l'intento del progetto era quello di mostrare come l'implementazione di un tool di questo genere sia effettivamente realizzabile.

RIFERIMENTI

1. Simulation-assisted machine learning
2. Mathematical Modelling of Molecular Pathways Enabling Tumour Cell Invasion and Migration
3. A survey of models for inference of gene regulatory networks