Analisi di dati biomedici e generazione automatica di regole

Marco Muselli

Consiglio Nazionale delle Ricerche (CNR)

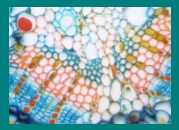
Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT)

e

IMPARA Srl (spin-off del CNR)

Analizzare per comprendere e prevedere

Cellula

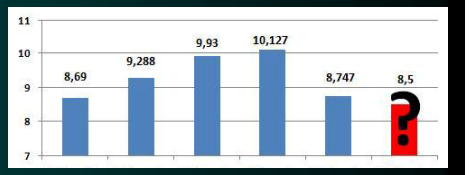


Tessuto

Organismo

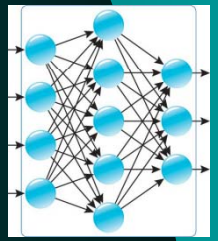
Dati

Analisi



Previsioni

Conoscenza



Modelli

Il metodo scientifico

Cellula, tessuto, organismo sono tre esempi di **sistema fisico**, il cui studio è basato sul metodo scientifico introdotto da Galileo.

Si distinguono essenzialmente tre fasi:

1. **Osservazione**: vengono effettuati esperimenti per raccogliere dati sul comportamento del sistema in oggetto.
2. **Induzione**: viene generato un modello matematico capace di descrivere con sufficiente accuratezza i dati rilevati.
3. **Applicazione**: il modello prodotto viene impiegato per prevedere il comportamento del sistema in tempi successivi.

Previsione = Conoscenza?

In alcuni casi non è di interesse prevedere il comportamento del sistema, ma soltanto comprendere le relazioni tra le sue variabili.

Ad es. comprendere i geni coinvolti nell'insorgere di una determinata patologia.

D'altra parte esistono tecniche predittive che non producono alcuna conoscenza sul sistema in esame.

Ad es. il metodo del nearest-neighbor assegna la classe ad un determinato soggetto sulla base della sua distanza dai dati noti.

Si può quindi avere conoscenza senza previsione e previsione senza conoscenza!

Modello = Conoscenza?

Nel metodo scientifico lo scopo di un modello è fare previsioni sull'andamento del sistema.

Se il modello è generato manualmente, la sua costruzione nasce dalla conoscenza dell'ideatore e ne esprime il contenuto.

Se il modello è generato automaticamente, può non apportare alcuna conoscenza.

Ad es. i modelli black-box (reti neurali, support vector machine, …) sono retti da equazioni non lineari difficilmente analizzabili.

In entrambi i casi il modello non produce conoscenza aggiuntiva!

Modelli che generano conoscenza

Questo tutorial sarà incentrato sui modelli predittivi capaci di generare nuova conoscenza su un sistema fisico a partire dai dati.

In genere tre tipi di informazione sono di grande interesse:

1. Misure di rilevanza delle variabili in gioco (e dei valori da esse assunti),
2. Indicazione delle relazioni che descrivono il modello (facilità di comprensione e di applicazione),
3. Valori di accuratezza e confidenza della previsione (sia a livello globale che di singolo dato).

Rumore, incertezza e probabilità

I modelli considerati saranno deterministici, in quanto la loro esecuzione non comporta l'impiego di variabili aleatorie.

Al contrario, i sistemi fisici considerati sono intrinsecamente affetti da rumore inteso come il risultato di variabilità intrinseche, errori di acquisizione, ridotta precisione numerica, …

È quindi necessario un approccio di tipo stocastico, capace di trattare appropriatamente l'incertezza insita nei dati.

Il modo maggiormente utilizzato per questo scopo fa uso delle probabilità e suppone l'esistenza di una specifica densità $p(\boldsymbol{x})$, caratteristica del sistema, che ha generato i dati a disposizione.

Metodi statistici e di machine learning

Esistono due approcci distinti alla costruzione di modelli da dati:

1. Metodo statistico: la densità $p(\boldsymbol{x})$ viene ritenuta nota (a meno di parametri) o viene stimata dai dati; successivamente il modello viene costruito a partire dalla $p(\boldsymbol{x})$. La bontà del modello viene valutata attraverso misure statistiche di confidenza (*p*-value)
2. Metodo di machine learning: il modello viene stimato direttamente dai dati senza ipotesi a priori sulla densità $p(\boldsymbol{x})$. L'accuratezza del modello viene misurata attraverso l'applicazione diretta su un insieme di dati indipendenti (cross-validation).

Due classi di problemi

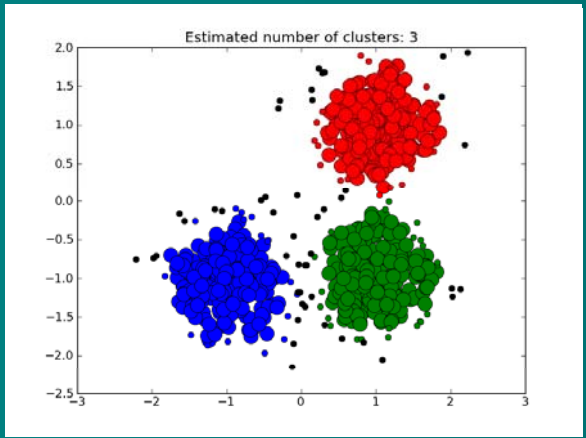
Si possono individuare due diverse classi di problemi:

1. Problemi di apprendimento non supervisionato (o di clustering): avente l'obiettivo di individuare gruppi di campioni vicini tra loro secondo una qualche misura di similarità;
2. Problemi di apprendimento supervisionato: esiste un'uscita y e il modello deve fornire per ogni $\boldsymbol{x}$ il valore di y che massimizza la densità condizionata $p(y \mid \boldsymbol{x})$.

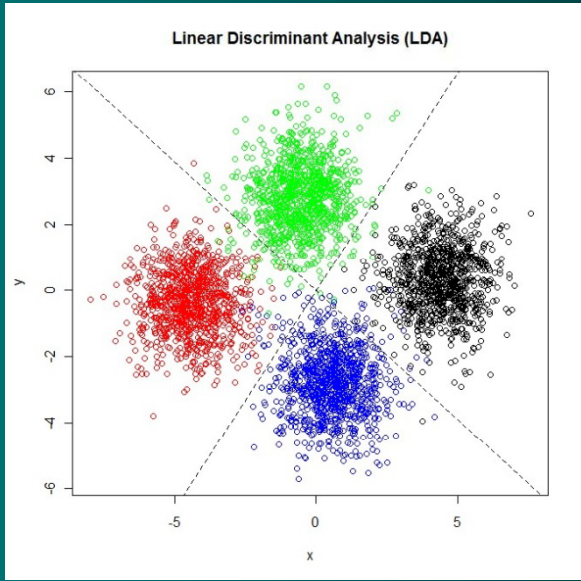
   Se la y è di tipo nominale si parla di problemi di classificazione; nel caso opposto (y di tipo ordinato) i problemi vengono detti di regressione.

Clustering, classificazione e regressione

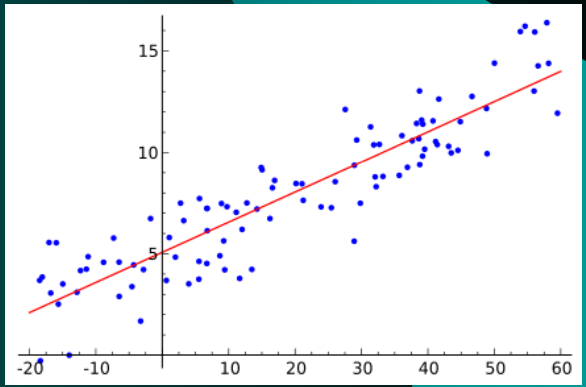
Clustering



Classificazione



Regressione



Stesso dataset → Problemi diversi

L'indagine su un sistema fisico può riguardare aspetti diversi del suo comportamento. Di conseguenza, l'analisi di un dataset biomedico può avvenire secondo obiettivi e modalità differenti, conducendo così a problemi di carattere diverso:

- Può cambiare la variabile d'uscita (dipendente) e l'insieme di variabili d'ingresso (indipendenti), anche attraverso trasformazioni che ne alterano il dominio o la dinamica.
- L'analisi (e conseguente costruzione del modello) può essere effettuata in modo supervisionato o non supervisionato.
- La tabella dei dati può essere esaminata per righe o per colonne, per finalità diverse.

Esempio: analisi di dati da microarray (1)

Da alcuni anni è possibile fotografare i geni attivi di una cellula (il loro livello di espressione) attraverso una macchina detta *microarray*.

Analizzando l'*mRNA* presente nelle cellule di un dato tessuto, il microarray produce uno spot luminoso per ognuno dei geni che si desidera esaminare.

L'intensità della luce emessa dallo spot fornisce una misura del livello di espressione del gene associato.

Esempio: analisi di dati da microarray (2)

Vengono eseguiti esperimenti diversi per confrontare:

- Cellule dello stesso tipo sottoposte ad un dato trattamento
- Cellule di organi diversi dello stesso individuo
- Cellule di tipo diverso (sane/malate, diverse malattie, ecc.)

Ogni esperimento con microarray dà origine a qualche migliaio (o decina di migliaia) di valori reali, ognuno dei quali corrisponde ad un livello di espressione genica.

Nella tabella risultante, ogni riga (o colonna) contiene i livelli di espressione genica ottenuti con un singolo esperimento.

Esempio: analisi di dati da microarray (3)

	Cell. #1	Cell. #2	.....	Cell. #*m*	Classif.
Gene #1	–214	–139	.....	17	TCA
Gene #2	–153	–73	.....	– 229	Ribo
.....	.....	.....	.....	.....	.....
Gene #*n*	–37	– 14	.....	– 16	HTH
Classif.	Sano	Malato	.....	Sano	

Una riga (colonna) finale può essere aggiunta per indicare una classificazione della cellula (del gene) considerata. Tipicamente $m \sim 100$, mentre $n \sim 10000$.

Esempio: analisi di dati da microarray (4)

Se analizziamo la tabella per righe otteniamo *n* punti in uno spazio a *m* dimensioni (*spazio dei geni*), ognuno dei quali rappresenta un determinato gene in diverse cellule o in diversi stati della stessa cellula.

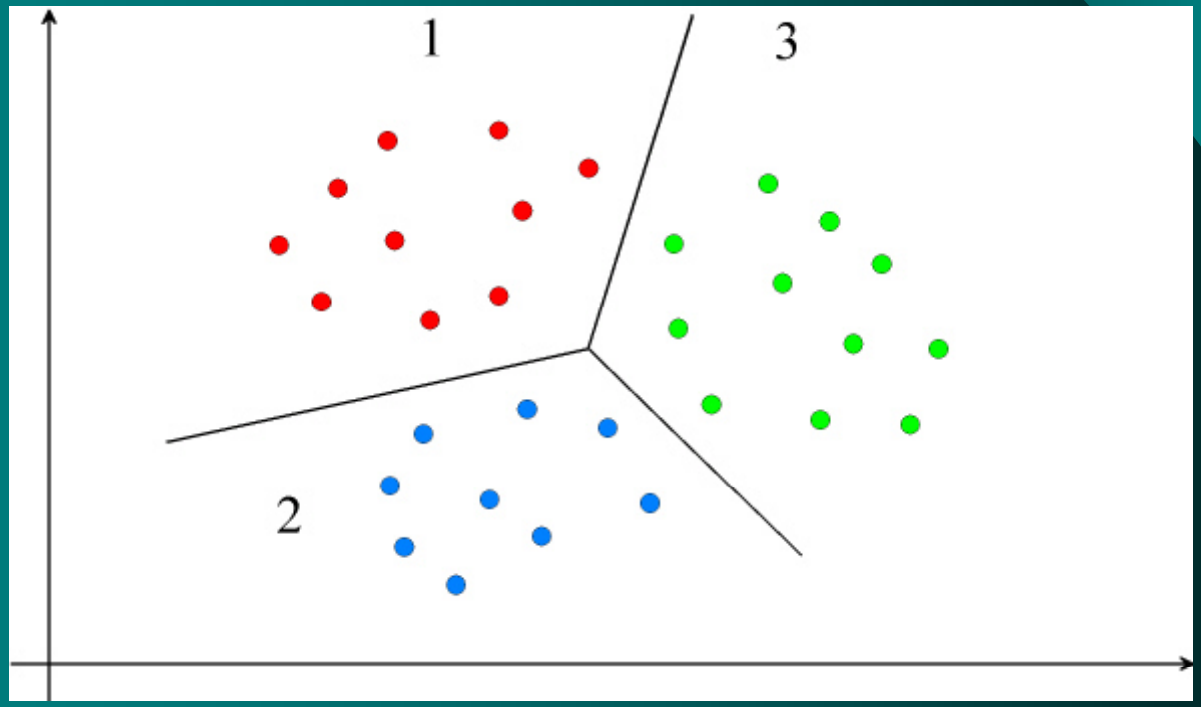
Se analizziamo la tabella per colonne otteniamo *m* punti in uno spazio a *n* dimensioni (*spazio delle cellule*), ognuno dei quali rappresenta una cellula o lo stato di una cellula.

Potremmo ad esempio perseguire i due seguenti obiettivi:

- Individuare cluster di punti simili nello spazio dei geni (riconoscere funzionalità, individuare catene metaboliche,…)
- Classificare i punti nello spazio delle cellule (individuare i geni responsabili di una determinata patologia,…)

Soluzione del problema di clustering

Definita una distanza opportuna tra i punti vengono generati *k*, che suddividono lo spazio in *k* regioni, ognuna delle quali contiene i punti più vicini al cluster associato.



Pro e contro del modello a cluster

Vantaggi:

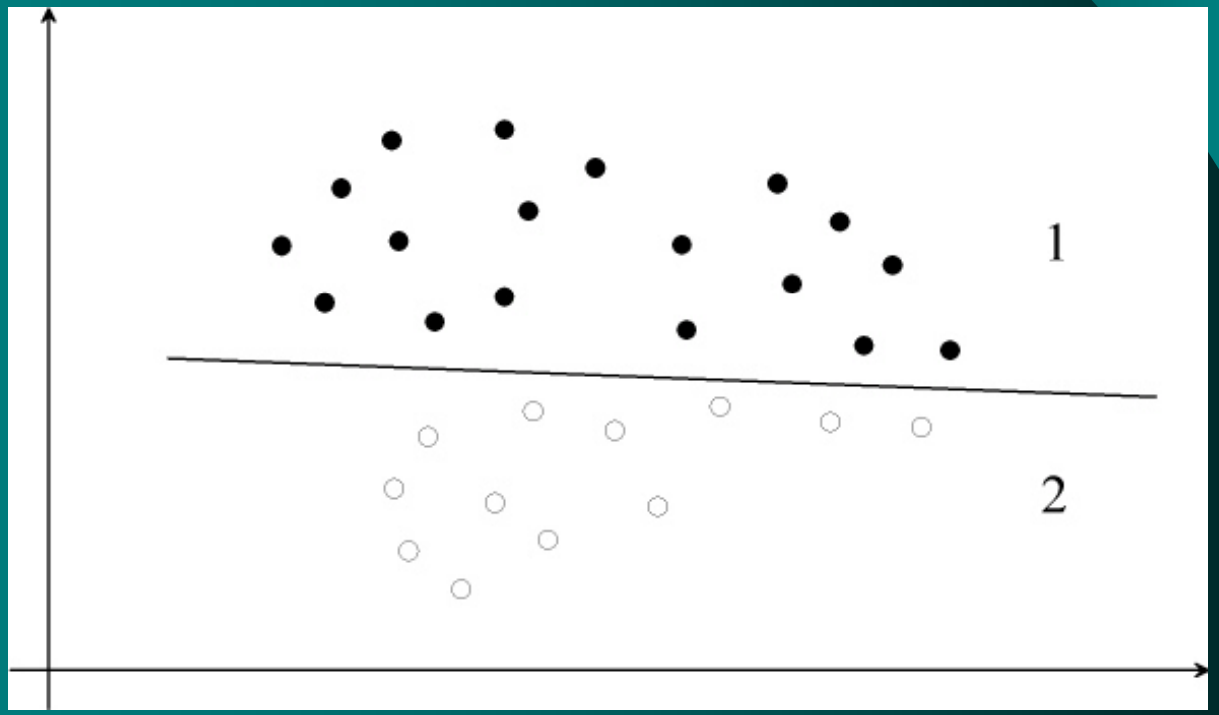
- Non occorre una classificazione a priori dei geni o delle cellule.
- Il tempo di esecuzione è generalmente ridotto.

Svantaggi:

- La scelta della distanza da impiegare è molto critica: il risultato dipende radicalmente da tale scelta.
- È difficile ricavare il numero ottimo *k* di cluster.
- Non si conosce con certezza il significato delle regioni individuate.

Soluzione del problema di classificazione

Data l'assegnazione dei punti del dataset alle varie classi, trovare le linee (o superfici) di separazione tra i gruppi di punti che meglio rispettano la classificazione data.



Pro e contro del modello supervisionato

Vantaggi:

- È sufficiente un minor numero di punti per trovare risultati significativi.
- Non ci sono dubbi sul significato delle regioni individuate.
- La validità dei risultati ottenuti può essere verificata immediatamente.

Svantaggi:

- Sono necessarie informazioni a priori sui dati.
- In genere i metodi richiedono un maggior tempo di esecuzione.

Metodi per la generazione di regole

In entrambi i casi l'impiego di un modello black box può portare al risultato desiderato senza però generare conoscenza sul sistema. Per altri dataset biomedici impiegare una black box può risultare indesiderato o inaccettabile (es. diagnosi medica).

È di gran lunga preferibile una tecnica capace di produrre modelli descritti da un insieme di regole intelligibili, normalmente indicata con il nome di **metodo per la generazione di regole**.

L'approccio più usato di questo tipo è quello degli **alberi decisionali**, che presenta numerose varianti (CART, ID3, C4.5, …)

Più recentemente sono state proposti metodi per la generazione di regole basati sulla **sintesi di funzioni booleane**, che in genere producono modelli con migliore accuratezza.

Esempio n.1: Diagnosi di patologie

Obiettivo: Diagnosticare tempestivamente la presenza di una data patologia, analizzando una serie di sintomi e/o risultati di analisi mediche eseguite su *n* pazienti.

Rule Generation Method

No disease

Sarebbe utile individuare i sintomi e/o le analisi più efficaci nel determinare la presenza della malattia. Estrarre un insieme di regole intelligibili permetterebbe di comprendere il legame tra sintomi e malattia, aprendo così la strada a nuovi metodi di cura.

Esempio n.2: Riconoscimento di segnali nel genoma

Obiettivo: Riconoscere la presenza di uno specifico segnale di regolazione all'interno di una sequenza di basi del DNA appartenente ad un determinato organismo.

```
ATGACCTGATCT
AGCGCGGGTATG
CATGTGAGTGAT
ACATGACTAGAT
CACATGTGCGGC
```

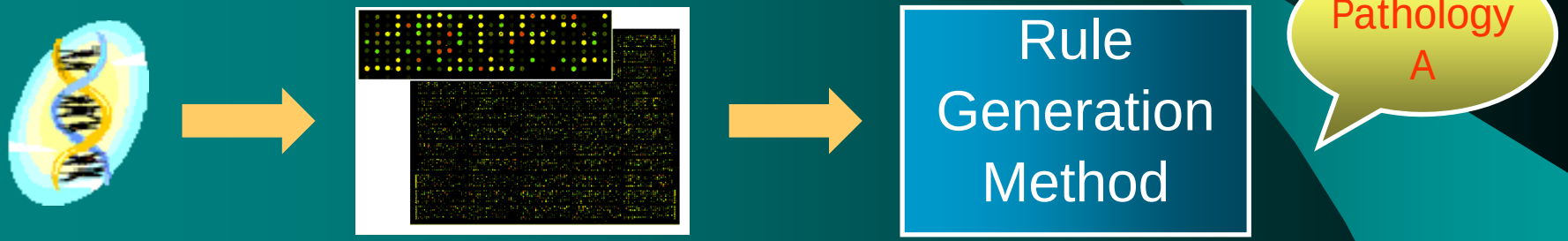
Rule Generation Method

Signal found

Data l'ampia variabilità delle sequenze è impossibile individuare il segnale per semplice ispezione. Si impiega allora una finestra di osservazione più ampia che scorre lungo la sequenza di basi, identificando così eventuali legami tra la presenza del segnale e la composizione delle regioni adiacenti.

Esempio n.3: Analisi di dati da microarray

Obiettivo: Classificare i tessuti di un determinato organismo (sano/malato, patologia A/patologia B, ecc.) sulla base dei dati di espressione genica ottenuti attraverso un DNA microarray.



In questo caso è fondamentale estrarre l'insieme di geni significativi per la malattia in esame (in totale sono decine di migliaia). L'insieme di regole potrebbe evidenziare le interazioni tra geni che conducono all'insorgere della patologia.

Problema generale di supervised learning

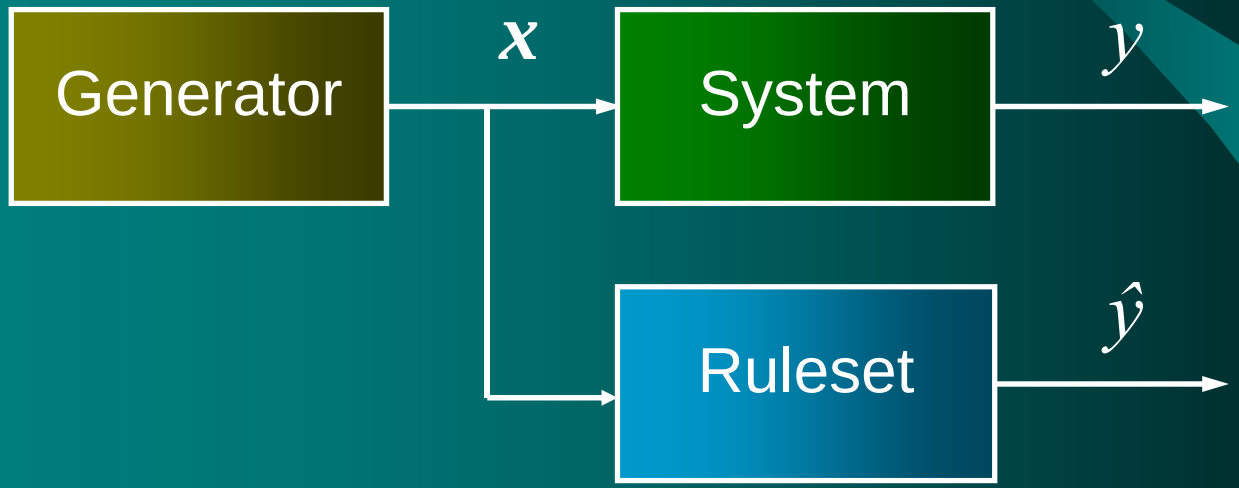
I tre esempi sopra menzionati sono casi particolari di un problema generale di induzione, denominato **supervised learning**.

Dato un sistema fisico, abbiamo a disposizione *n* esempi (sintomi di pazienti, sequenze di basi, vetrini di DNA microarray), ad ognuno dei quali è associata un'etichetta (stato del paziente, presenza o assenza del segnale, stato del tessuto). In genere gli esempi sono affetti da rumore di acquisizione.

Obiettivo: Costruire un modello (insieme di regole) che emuli il comportamento del sistema fisico, ovvero scelga l'etichetta corretta (più probabile) in corrispondenza di una situazione non contemplata negli *n* esempi a disposizione.

Formulazione matematica del problema

Dato un sistema che riceve un vettore di ingressi $\boldsymbol{x}$ da un generatore e produce in risposta un'uscita y



desideriamo costruire, a partire da un insieme di n osservazioni $(\boldsymbol{x}_1,y_1)$, $(\boldsymbol{x}_2,y_2)$,…, $(\boldsymbol{x}_n,y_n)$ (*training set*), un insieme di regole intelligibili che emuli il comportamento del sistema producendo un'uscita $\hat{y}$ che approssimi al meglio la y desiderata.

Esempio: diagnosi di patologia

Supponiamo di dover diagnosticare una determinata patologia, conoscendo la quantità di globuli bianchi (x_1) e rossi (x_2) nel sangue. L'uscita del nostro problema è pertanto lo stato del paziente, che può essere Healthy ($y = 0$) o Sick ($y = 1$).

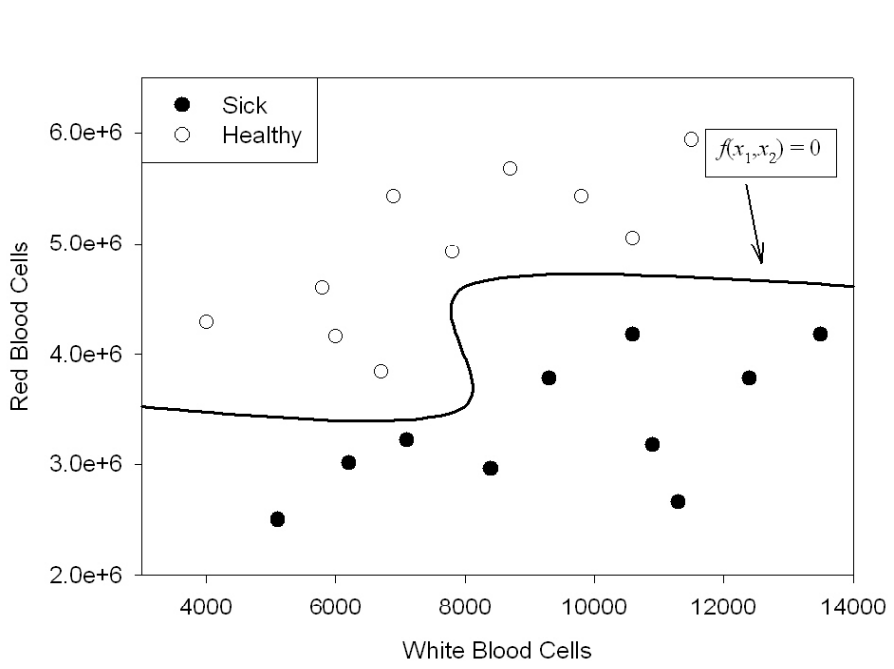
Obiettivo: Determinare un insieme di regole, a partire dal seguente training set, che associ ad ogni coppia (x_1, x_2) il valore corretto dell'uscita y.

White blood cells	Red blood cells	Status	White blood cells	Red blood cells	Status
5100	2500000	Sick	4000	4290000	Healthy
6200	3010000	Sick	5800	4600000	Healthy
7100	3220000	Sick	6000	4160000	Healthy
8400	2960000	Sick	6700	3840000	Healthy
9300	3778000	Sick	6900	5430000	Healthy
10600	4178000	Sick	7800	4920000	Healthy
10900	3170000	Sick	8700	5678000	Healthy
11300	2660000	Sick	9800	5430000	Healthy
12400	3778000	Sick	10600	5050000	Healthy
13500	4178000	Sick	11500	5940000	Healthy

Esempio di problema di apprendimento (cont.)

I casi del training set possono essere rappresentati su un grafico bidimensionale con punti colorati diversamente a seconda del valore dell'uscita y.

Risolvere il problema di apprendimento equivale pertanto a trovare una funzione bidimensionale $f(x_1,x_2)$ che assuma valore > 0 in corrispondenza delle coppie (x_1,x_2) che conducono ad una buona uscita.

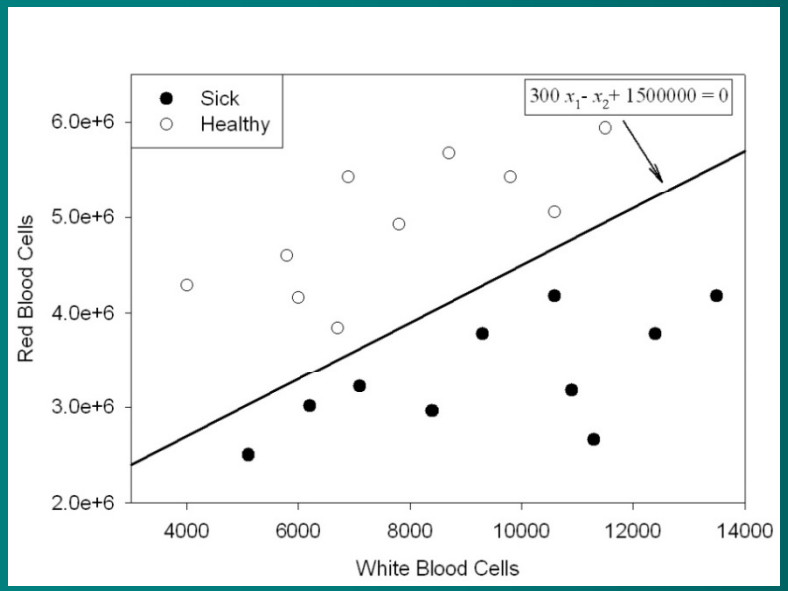


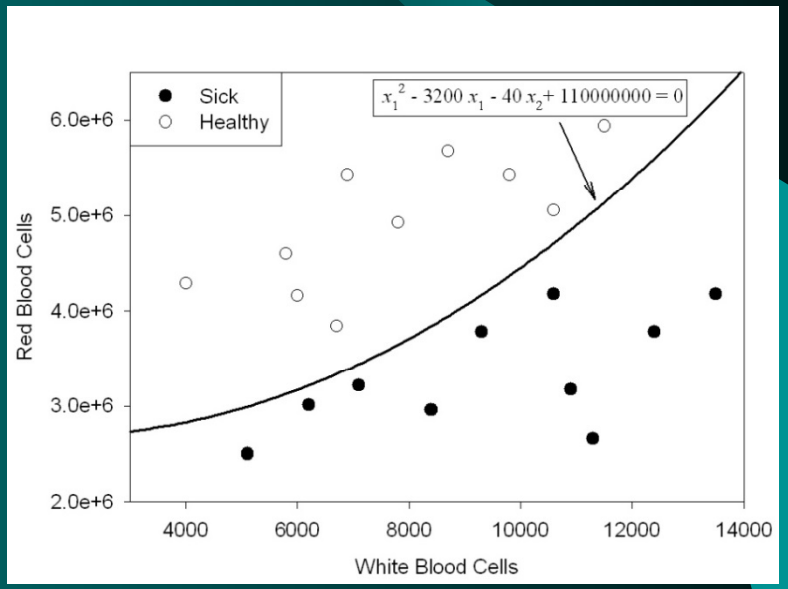
La funzione $f(x_1,x_2)$ desiderata deve soddisfare il maggior numero di casi del training set.

Comprensibilità della soluzione ottenuta

È immediato vedere che nell'esempio in esame esistono infinite funzioni $f(x_1,x_2)$ che verificano tutti i punti del training set.

Due scelte possibili (lineare e quadratica) sono le seguenti:

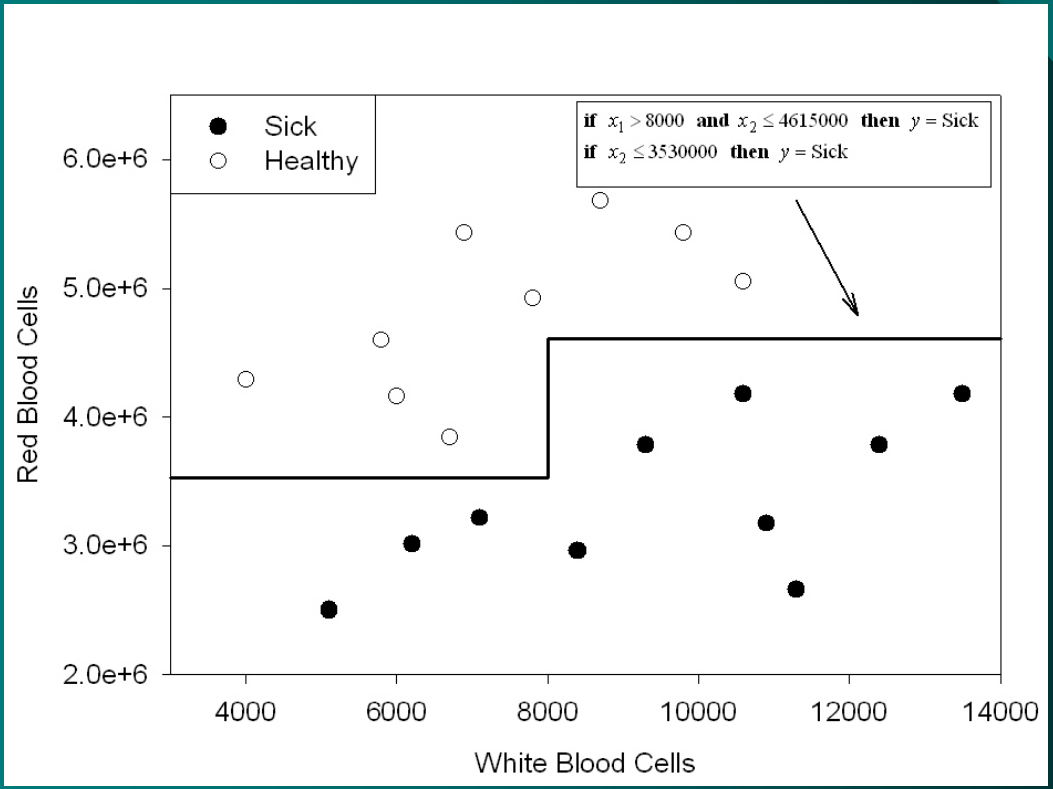




In entrambi i casi si prevede l'uscita, ma è difficile comprendere il funzionamento del dispositivo artificiale che le implementa.

Comprensibilità della soluzione ottenuta (cont.)

Una descrizione del dispositivo artificiale in termini di semplici regole a soglia del tipo **if-then** permette invece di avere una piena comprensione del suo funzionamento.



Metodi per la generazione di regole

I metodi per la generazione di regole hanno lo scopo di estrarre dal training set un insieme di regole intelligibili che descriva il funzionamento del sistema in esame.

Una **regola** è una struttura del tipo:

if <*premessa*> **then** <*conseguenza*>

Esempio: **if** $x_1 \geq 1$ AND $x_2 < 4$ **then** $y = 1$

I metodi per la generazione di regole, che posseggono la base teorica più convincente, sono le tecniche che impiegano alberi.

Tali tecniche presentano generalmente un'accuratezza inferiore a quella delle reti neurali, ma consentono di comprendere il comportamento del sistema. Spesso le regole sono poco specifiche.

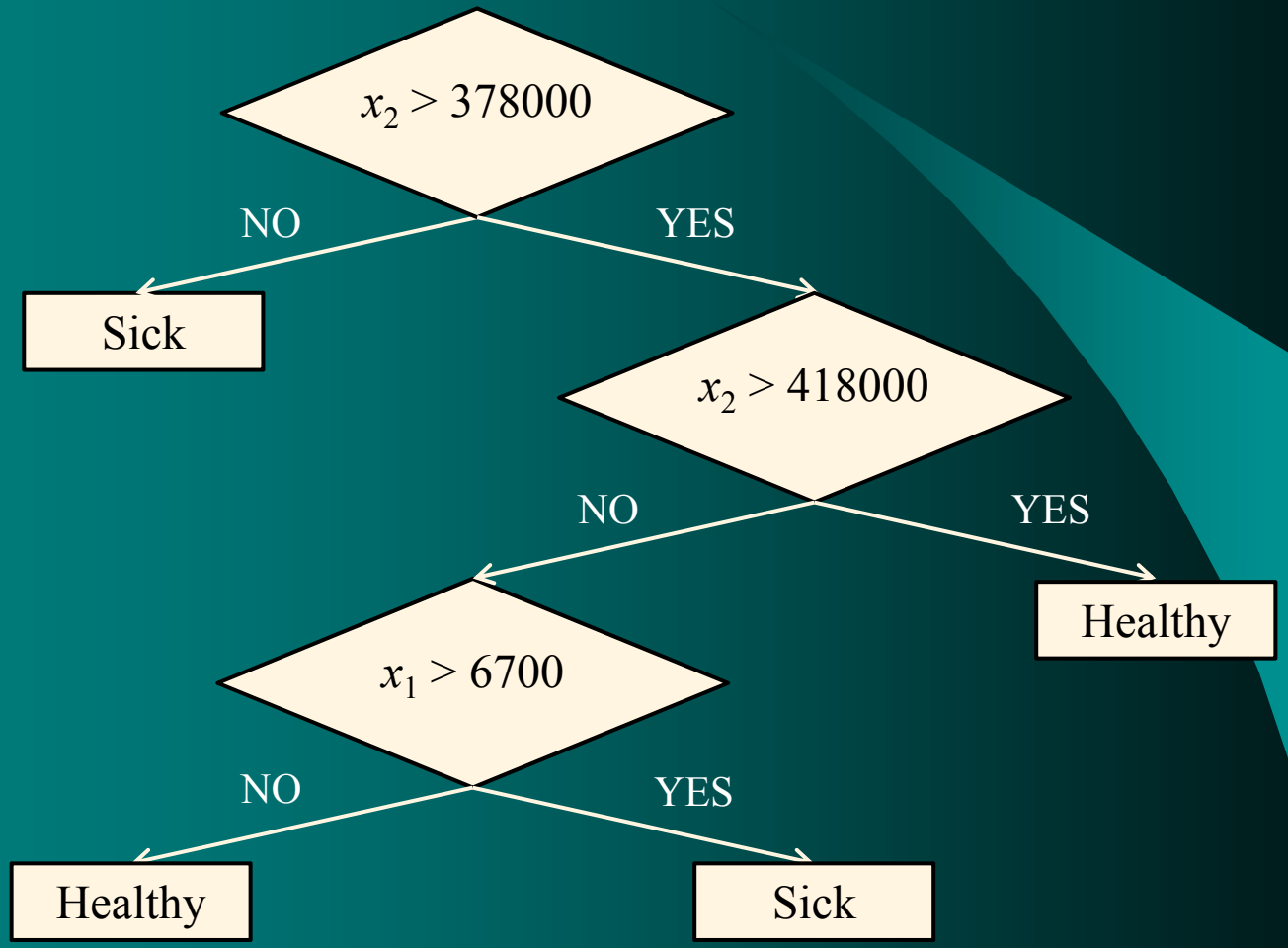
Alberi decisionali

Generano un classificatore ad albero nel quale ogni **foglia** è associata ad un valore d'uscita (classe) ed ogni nodo non terminale è un **nodo di decisione** che specifica un test da eseguire su un singolo ingresso, con un ramo in uscita per ogni possibile esito.

Per produrre la classe a cui associare un dato esempio, si percorre l'albero partendo dalla radice e discendendo lungo di esso finché non si giunge ad una foglia, il cui valore di uscita fornisce la classe desiderata.

È immediato estrarre da un albero decisionale un insieme di regole intelligibili. Partendo da ogni foglia si percorre a ritroso l'albero fino alla radice: l'AND logico delle condizioni incontrate forma l'antecedente di una regola.

Diagnosi di patologia: albero risultante



Diagnosi di patologia: esempio e regola

È immediato vedere che ai punti del dataset vengono associati i corretti valori di uscita. Ad esempio, il punto avente $x_1 = 6000$ e $x_2 = 416000$ viene classificato come Healthy in quanto discende l'albero seguendo i rami YES, NO e NO in corrispondenza dei 3 test.

Inoltre partendo dalla foglia in basso a sinistra si ottiene la regola:

if $x_1 \leq 6700$ AND $378000 < x_2 \leq 418000$ **then** y = Healthy

Costruzione dell'albero decisionale: divide and conquer

La costruzione dell'albero decisionale avviene suddividendo progressivamente il dataset iniziale in sottoinsiemi via via sempre più omogenei quanto a classe in uscita (**divide and conquer**).

Ogni suddivisione procede sulla base di un test su una determinata variabile d'ingresso. Sia la variabile su cui agire che il test specifico vengono scelti valutando la diminuzione dell'entropia ottenuta effettuando la suddivisione risultante dal test.

Se Y è la variabile aleatoria che determina i valori dell'uscita abbiamo dalla definizione di entropia:

$$H(Y) = -Pr(Y = \text{Sick}) \log_2 Pr(Y = \text{Sick}) - Pr(Y = \text{Healthy}) \log_2 Pr(Y = \text{Healthy})$$

Valutazione di un test

Nell'esempio in esame si ha $Pr(Y = \text{Sick}) = Pr(Y = \text{Healthy}) \sim 0.5$ e pertanto $H(Y) = 1$.

L'esecuzione del test $x_1 > a$, indicato con $t(X_1,a)$, conduce alla creazione di due sottoinsiemi (contenenti i punti aventi $x_1 \leq a$ e $x_1 > a$). L'entropia di tale situazione risulta essere pertanto

$$H(Y \mid t(X_1,a)) = Pr(X_1 \leq a)\, H(Y \mid X_1 \leq a) + Pr(X_1 > a)\, H(Y \mid X_1 > a)$$

dove l'entropia condizionata $H(Y \mid X_1 > a)$ è data da

$$H(Y \mid X_1 > a) = -Pr(Y = \text{Sick} \mid X_1 > a) \log_2 Pr(Y = \text{Sick} \mid X_1 > a)$$
$$- Pr(Y = \text{Healthy} \mid X_1 > a) \log_2 Pr(Y = \text{Healthy} \mid X_1 > a)$$

Valutazione di un test (cont.)

Se consideriamo per esempio il test $t(X_1,7800)$, otteniamo:

$Pr(X_1 \leq 7800) \sim 9/20 = 0.45$, $\quad Pr(X_1 \leq 7800) \sim 11/20 = 0.55$

$Pr(Y = \text{Sick} \mid X_1 \leq 7800) = 1 - Pr(Y = \text{Healthy} \mid X_1 \leq 7800) \sim 3/9 = 0.333$

$Pr(Y = \text{Sick} \mid X_1 > 7800) = 1 - Pr(Y = \text{Healthy} \mid X_1 > 7800) \sim 7/11 = 0.636$

per cui si ottiene:

$H(Y \mid X_1 \leq 7800) \sim 0.918$, $\quad H(Y \mid X_1 \leq 7800) \sim 0.946$

$H(Y \mid t(X_1,7800)) \sim 0.45 \cdot 0.918 + 0.55 \cdot 0.946 = 0.933$

Di conseguenza la diminuzione di entropia dovuta al test è pari a:

$H(Y) - H(Y \mid t(X_1,7800)) \sim 1 - 0.933 = 0.067$

Costruzione di un nodo

Calcolando la stessa quantità per tutti i possibili valori delle due variabili, si ottiene che la massima diminuzione di entropia è associata al test $t(X_2,378000)$

$$H(Y) - H(Y \mid t(X_2,378000)) \sim 1 - 0.39 = 0.61$$

che costituisce il primo nodo dell'albero decisionale.

Notiamo ora che tutti i punti del dataset con $x_2 \leq 378000$ appartengono alla classe Sick. Pertanto, non è necessaria alcuna ulteriore suddivisione di tale sottoinsieme e possiamo porre una foglia nell'albero risultante con label Sick.

Ripetendo il procedimento per il secondo sottoinsieme ($x_2 > 378000$) si ottengono i due test aggiuntivi $t(X_2,418000)$ e $t(X_1,6700)$.

Pruning dell'albero decisionale

Generalmente il processo di suddivisione conduce ad un numero eccessivo di foglie, a cui corrispondono sottoinsiemi troppo piccoli. Gli alberi decisionali così frammentati presentano una ridotta accuratezza.

Per risolvere tale problema viene effettuata al termine della costruzione dell'albero una fase di **pruning** avente lo scopo di semplificarne la struttura rimuovendo nodi e sottoalberi non necessari.

Il processo di pruning viene guidato da una stima dell'accuratezza ottenuta su un insieme indipendente di dati o attraverso una valutazione di tipo statistico.

Metodi basati sulla sintesi di funzioni booleane

In pratica, i metodi di apprendimento devono effettuare il fitting di una funzione incognita $y = f(\boldsymbol{x})$ sulla base dei dati a disposizione. In genere, questo è un problema mal posto.

Un modo alternativo è quello di mappare opportunamente il dominio dei dati in un insieme binario $\{0,1\}^b$ ed impiegare le tecniche di sintesi delle funzioni booleane per effettuare il fitting.

La procedura generale impiegata da tale approccio è la seguente:

1. Discretizza le variabili continue.
2. Binarizza le variabili discrete (ordinate e nominali).
3. Esegui la sintesi della funzione booleana.
4. Trasforma ogni implicante in una regola intelligibile.

Discretizzazione e codifica binaria

Esistono in letteratura diverse tecniche efficaci per discretizzare le variabili continue, ad es. EntMDL (Kohavi & Sahami, 1996), Chi2 (Liu & Setiono, 1997), LAD (Boros et al., 2000).

Più delicata è la scelta della codifica binaria per le variabili discrete, ordinate e nominali. Per ogni ingresso x dobbiamo definire una trasformazione $\lambda : I_k \rightarrow \{0,1\}^b$, essendo $I_k = \{1, \ldots, k\}$.

Per mantenere l'ordinamento dei valori, occorre che la funzione λ sia **full order preserving**, ovvero $\lambda(u) \leq \lambda(v)$ se e solo se $u \leq v$. Per mantenere le informazioni relative alla distanza, occorre che la funzione λ sia un'**isometria**, cioè $d_b(\lambda(u), \lambda(v)) = \alpha\, d(u,v)$, essendo d_b e d rispettivamente le distanze in $\{0,1\}^b$ e I_k.

Normalmente in $\{0,1\}^b$ viene usata la **distanza di Hamming** $d_b(\boldsymbol{x},\boldsymbol{z})$, pari al numero di bit differenti nelle stringhe $\boldsymbol{x}$ e $\boldsymbol{z}$.

Codifica binaria degli ingressi

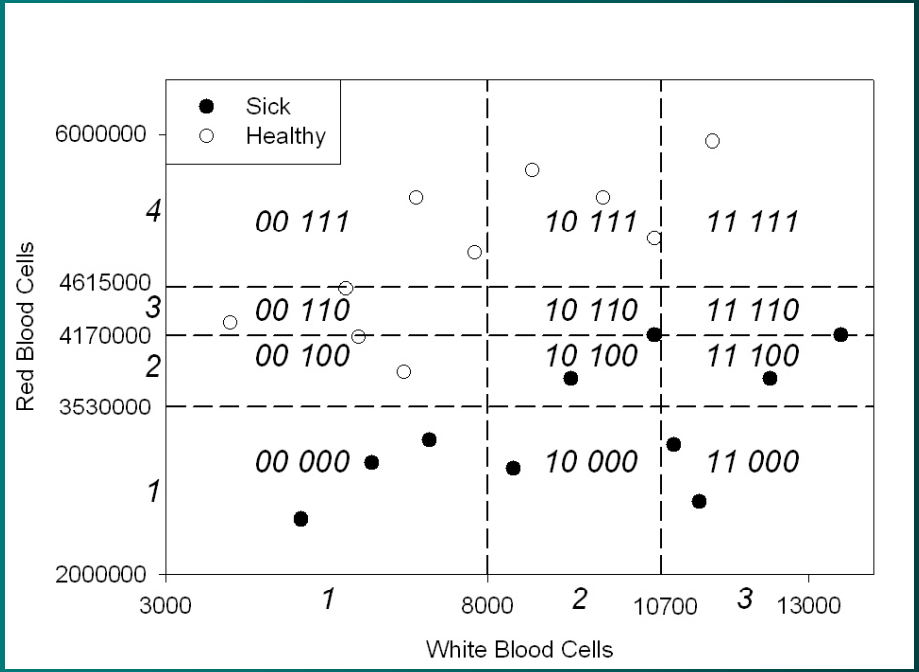
Nel caso di variabili ordinate una scelta valida per λ è data dalla **codifica a termometro**, che pone a 1 l'i-esimo bit se e solo se $i < u$. Deve pertanto essere $b = k - 1$.

Nel caso di variabili nominali una scelta valida per λ è data dalla **codifica only-one**, che pone a 1 l'i-esimo bit se e solo se $i = u$. Quindi si ha $b = k$.

Valore	Cod. standard	Cod. a termometro	Cod. only-one
1	0000	000000000	1000000000
2	0001	100000000	0100000000
…	…	…	…
9	1000	111111110	0000000010
10	1001	111111111	0000000001

Diagnosi di patologia: discretizzazione

Nell'esempio sulla diagnosi di patologia i dati sono continui; quindi, la loro codifica in formato binario richiede una discretizzazione:



Le due variabili x_1 e x_2 vengono così trasformate in due variabili discrete u_1 e u_2 e si ha $u_i = k$ se x_i cade nel k-esimo intervallo.

Diagnosi di patologia: codifica binaria

Poiché le due variabili u_1 e u_2 sono ordinate, impiegheremo la codifica a termometro per costruire la codifica binaria dei dati. Per l'uscita poniamo $y = 1$ se il paziente è Sick, $y = 0$ altrimenti.

x_1	x_2	u_1	u_2	**z**	y	x_1	x_2	u_1	u_2	**z**	y
5100	2500000	1	1	00 000	1	4000	4290000	1	3	00 110	0
6200	3010000	1	1	00 000	1	5800	4600000	1	3	00 110	0
7100	3220000	1	1	00 000	1	6000	4160000	1	2	00 100	0
8400	2960000	2	1	10 000	1	6700	3840000	1	2	00 100	0
9300	3778000	2	2	10 100	1	6900	5430000	1	4	00 111	0
10600	4178000	2	3	10 110	1	7800	4920000	1	4	00 111	0
10900	3170000	3	1	11 000	1	8700	5678000	2	4	10 111	0
11300	2660000	3	1	11 000	1	9800	5430000	2	4	10 111	0
12400	3778000	3	2	11 100	1	10600	5050000	2	4	10 111	0
13500	4178000	3	3	11 110	1	11500	5940000	3	4	11 111	0

Soluzione del problema di apprendimento

Concatenando le stringhe generate dall'applicazione delle codifiche λ scelte per le variabili d'ingresso si perviene ad una trasformazione β dallo spazio degli ingressi all'insieme delle stringhe binarie. In generale β è iniettiva e non surgettiva.

Impiegando un metodo per la sintesi di reti logiche capace di generalizzare, si ottiene quindi una funzione booleana $f(\mathbf{z})$ da cui si ricava il classificatore desiderato $g(\boldsymbol{x}) = f(\beta(\boldsymbol{x}))$.

Possibili metodi di sintesi sono Logical Analysis of Data (Boros et al., 2000) ed Hamming Clustering (Muselli & Liberati, 2000).

Essi producono un insieme di **implicanti**, ovvero stringhe di elementi dell'insieme {0,1,*}, essendo * un simbolo *don't care*. Ad ogni implicante corrisponde un prodotto logico, a cui è immediato associare una regola intelligibile del tipo **if-then**.

Sintesi di funzioni booleane

Attraverso il mapping β il training set originale $\{(\boldsymbol{x}_i,y_i),\ i=1,\ldots,n\}$ viene trasformato in un insieme di esempi $S=\{(\mathbf{z}_i,y_i),\ i=1,\ldots,n\}$, dove $\mathbf{z}_i=\beta(\boldsymbol{x}_i)$ è una stringa binaria.

La funzione booleana $f(\mathbf{z})$ viene ricostruita attraverso i seguenti passi:

1. Poni $T=S$ e $C=\varnothing$.
2. Partendo dall'implicante $***\cdots*$, trasforma alcuni don't care in bit in modo da ottenere un implicante $\boldsymbol{c}$ che copra il maggior numero di esempi in T con uscita $y_i=1$ e nessun esempio in T con $y_i=0$.
3. Aggiungi l'implicante $\boldsymbol{c}$ all'insieme C. Elimina da T tutti gli esempi $(\boldsymbol{x}_i,y_i)$ con uscita $y_i=1$ e $\boldsymbol{x}_i$ coperto da $\boldsymbol{c}$. Se T non è vuoto vai a 2.
4. Semplifica l'insieme C estraendo un sottoinsieme minimo.

Sintesi di funzioni booleane (cont.)

Mentre LAD costruisce l'implicante $\boldsymbol{c}$ al passo 2 attraverso una ricerca esaustiva (algoritmo *NP*-completo), HC cambia un don't care alla volta impiegando un criterio di scelta efficace (procedura greedy).

Quindi HC è un algoritmo subottimo e conduce in generale a soluzioni meno buone di LAD, ma la sua applicabilità è più ampia potendo trattare problemi ad elevata dimensionalità (ad es. analisi di dati da DNA microarray).

Diagnosi di patologia: sintesi della funzione booleana

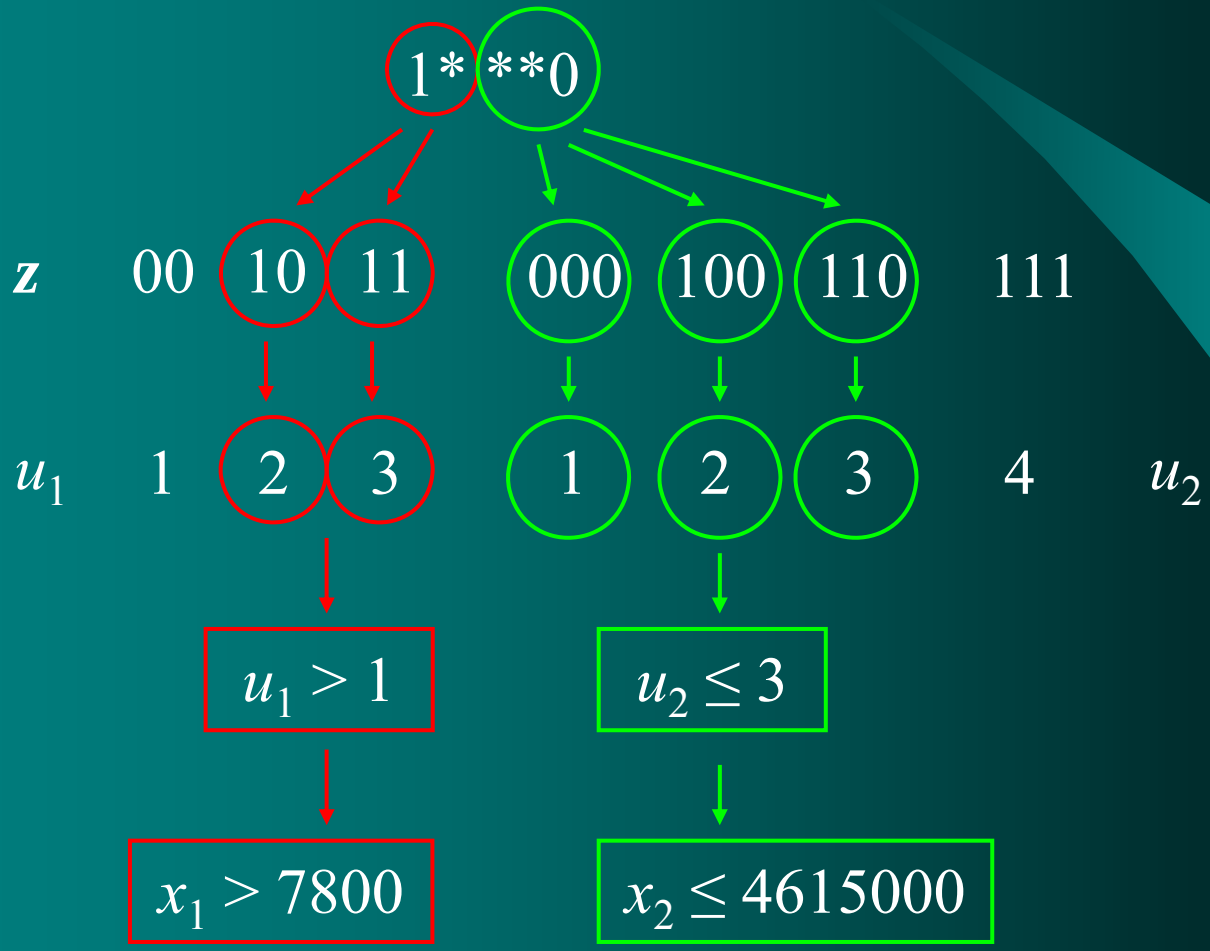
Nell'esempio in esame HC genera i due implicanti '1* **0' e '** 000' per l'uscita $y = 1$, che corrispondono alla funzione booleana $f(\mathbf{z}) = z_1\bar{z}_5 + \bar{z}_3\bar{z}_4\bar{z}_5$.

Un semplice controllo mostra che l'implicante '1* **0' soddisfa (copre) 7 esempi del training set con uscita $y = 1$ ('10 000', '10 100', '10 100', '10 110', '11 000', '11 100', '11 110'), ottenibili sostituendo opportuni valori binari ad ogni '*'. Analogamente l'implicante '** 000' copre 6 esempi.

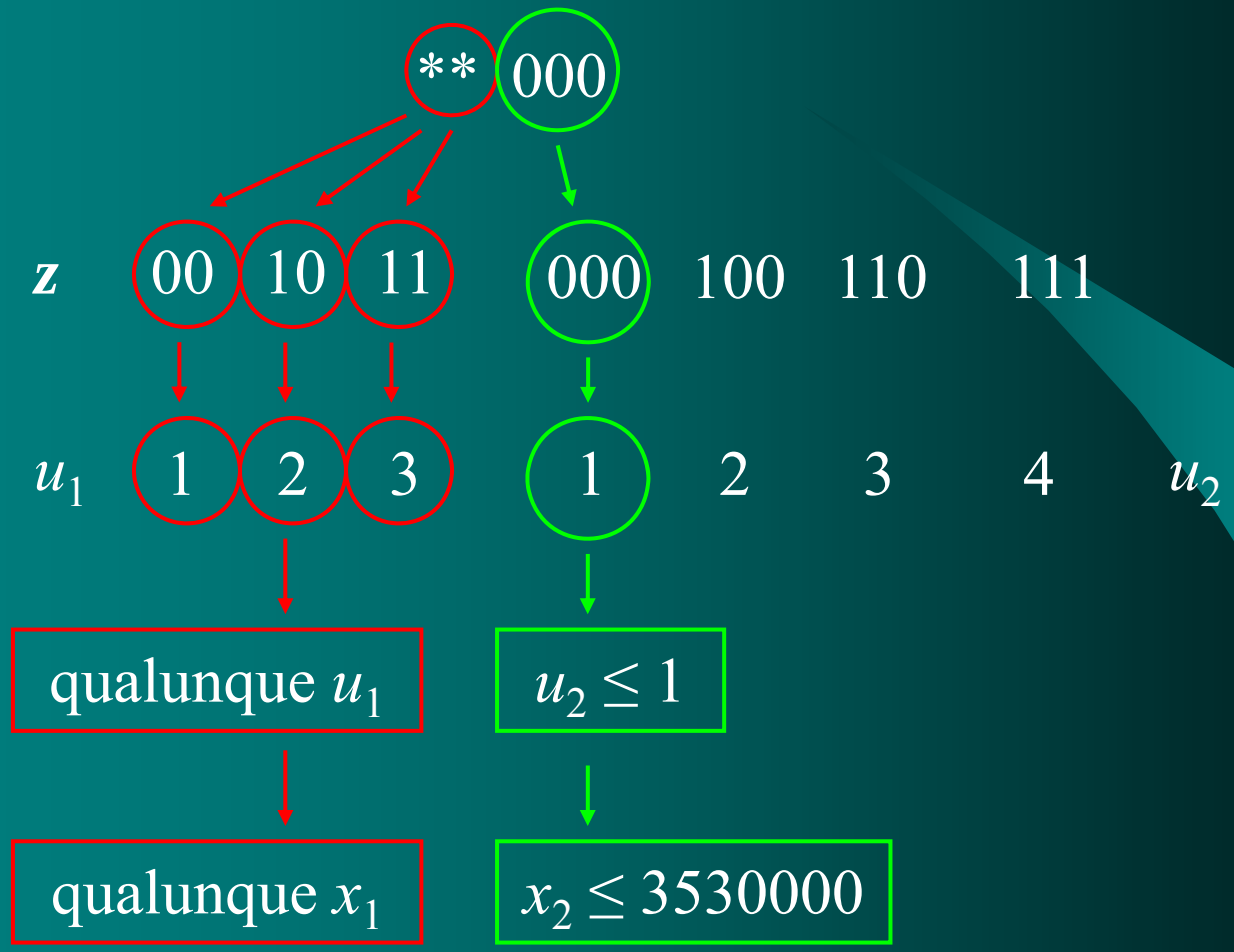
La funzione booleana generata è consistente con il training set, in quanto nessun implicante copre esempi con uscita $y = 0$.

Diagnosi di patologia: generazione delle regole

Ad ogni implicante può essere associata una regola intelligibile: è sufficiente osservare tutte le stringhe binarie da esso coperte.



Diagnosi di patologia: generazione delle regole (cont.)



Diagnosi di patologia: generazione delle regole (cont.)

Quindi la funzione booleana $f(\mathbf{z}) = z_1\bar{z}_5 + \bar{z}_3\bar{z}_4\bar{z}_5$ equivale all'unione delle due regole

- **if** $u_1 > 1$ AND $u_2 \leq 3$ **then** $y = 1$
- **if** $u_2 \leq 1$ **then** $y = 1$

che corrispondono alle seguenti regole sulle variabili continue x_1 e x_2

- **if** $x_1 > 7800$ AND $x_2 \leq 4615000$ **then** $y = 1$
- **if** $x_2 \leq 3530000$ **then** $y = 1$

Problemi nell'approccio booleano

La codifica binaria impiegata è ridondante: esistono molte stringhe binarie che non hanno alcun significato. Alcune di queste hanno distanza di Hamming minima (unitaria) dalle stringhe generate attraverso la codifica e ostacolano il processo di sintesi.

Ad es. nel problema di esempio la stringa '10 101' non ha significato pur trovandosi a distanza unitaria da '10 100' e da '10 111'.

Esistono implicanti a cui non corrisponde alcuna regola ('** 101') e regole a cui corrispondono più implicanti (sia '** *0*' che '** *00' corrispondono a $u_2 \leq 2$).

Si può dimostrare che questi problemi possono essere risolti eliminando l'impiego dell'operazione NOT e considerando esclusivamente le **funzioni monotone booleane**, esprimibili attraverso un'espressione contenente solo operatori AND e OR.

Codifica e distanze

Eliminare l'operazione NOT equivale a considerare $\{0,1\}^n$ come **reticolo booleano** (**Boolean lattice**) e non come algebra booleana. In questo caso si dimostra che la codifica only-one (inversa) è full order preserving (per variabili ordinate) e un'isometria.

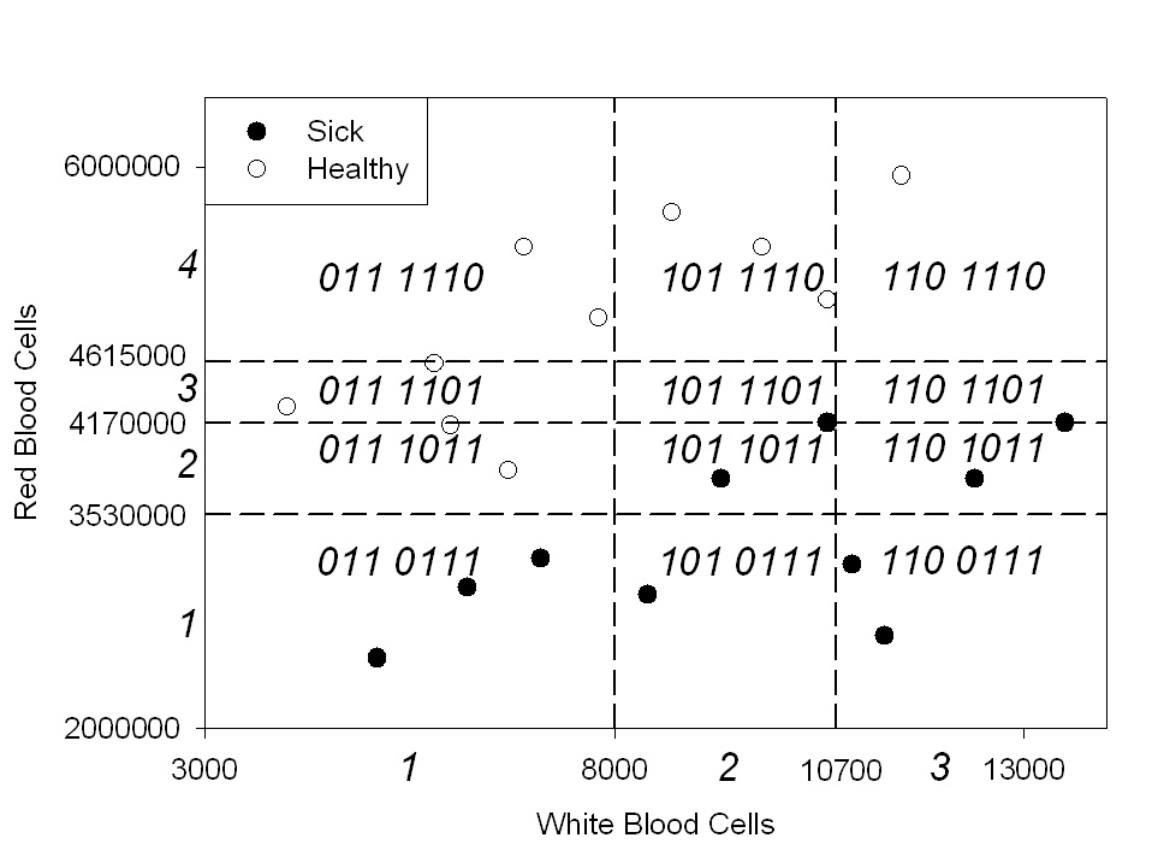
La distanza tra stringhe binarie viene definita in modo diverso a seconda che la variabile da codificare sia ordinata o nominale.

Nel primo caso si usa l'**ordinamento lessicografico** ($u < v$ se e solo se $u_i < v_i$ per la componente i-esima e $u_k = v_k$ per ogni $k < i$) e la distanza usuale in questo ordinamento. Così si ha 11110 < 10111, mentre la distanza tra le due stringhe è pari a 3.

Nel caso la variabile da codificare sia nominale si adotta la distanza uniforme. In questo modo la distanza tra 11110 e 10111 è pari a 1.

Diagnosi di patologia: nuova binarizzazione

Con la codifica only-one inversa la partizione dello spazio degli ingressi diventa:



Diagnosi di patologia: codifica nel reticolo booleano

Impiegando la codifica only-one inversa per il problema di apprendimento sopra analizzato, sono richiesti tre bit per la variabile u_1 e quattro bit per la variabile u_2.

x_1	x_2	u_1	u_2	**z**	y	x_1	x_2	u_1	u_2	**z**	y
5100	2500000	1	1	011 0111	1	4000	4290000	1	3	011 1101	0
6200	3010000	1	1	011 0111	1	5800	4600000	1	3	011 1101	0
7100	3220000	1	1	011 0111	1	6000	4160000	1	2	011 1011	0
8400	2960000	2	1	101 0111	1	6700	3840000	1	2	011 1011	0
9300	3778000	2	2	101 1011	1	6900	5430000	1	4	011 1110	0
10600	4178000	2	3	101 1101	1	7800	4920000	1	4	011 1110	0
10900	3170000	3	1	110 0111	1	8700	5678000	2	4	101 1110	0
11300	2660000	3	1	110 0111	1	9800	5430000	2	4	101 1110	0
12400	3778000	3	2	110 1011	1	10600	5050000	2	4	101 1110	0
13500	4178000	3	3	110 1101	1	11500	5940000	3	4	110 1110	0

Implicanti e regole

Un aspetto importante delle funzioni monotone booleane è legato agli implicanti, che in tal caso sono dati da stringhe di elementi nell'insieme {0,1}.

Il valore 0 funge da simbolo *don't care* e può essere sostituito dal valore binario 1 al fine di determinare le stringhe che soddisfano un certo implicante.

Così, nel problema di esempio, l'implicante '000 0111' è soddisfatto dalle stringhe '011 0111', '101 0111', '110 0111' e corrisponde pertanto alla condizione $u_2 \leq 1$.

Si può inoltre dimostrare che esiste una corrispondenza biunivoca tra implicanti e regole: non esistono più implicanti senza regole corrispondenti, né regole associate a più implicanti.

Shadow Clustering (SC)

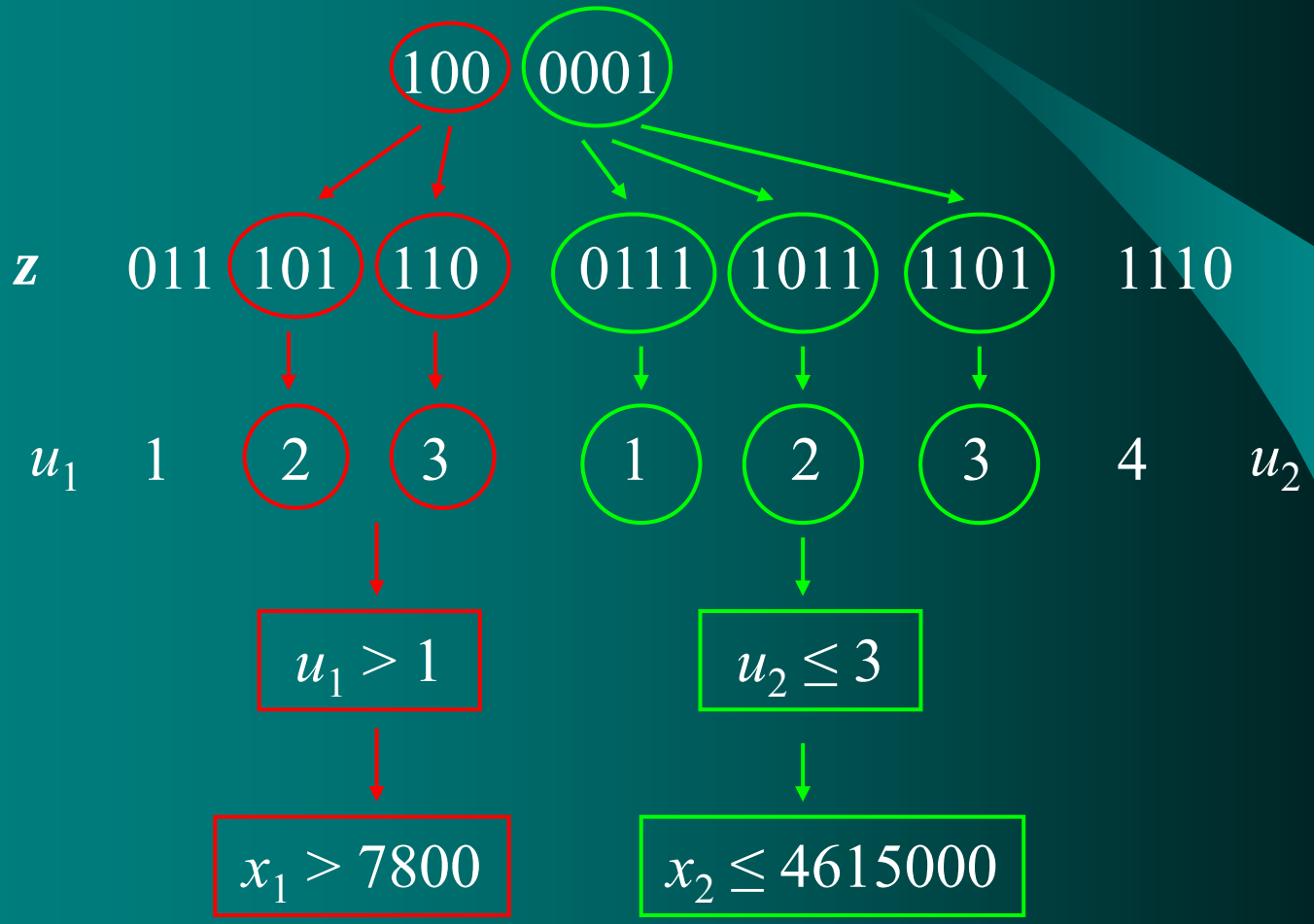
L'esigenza di operare nel reticolo booleano richiede l'impiego di un metodo per la sintesi di funzioni monotone booleane.

Si è pertanto sviluppato un algoritmo apposito, chiamato **Shadow Clustering (SC)**, che opera in modo simile ad HC:

1. Poni $T = S$ e $C = \varnothing$.
2. Partendo dall'implicante $000\cdots0$, trasforma alcuni 0 in 1 così da ottenere un implicante $\boldsymbol{c}$ che copra il maggior numero di esempi in T con uscita $y_i = 1$ e nessun esempio in T con $y_i = 0$.
3. Aggiungi l'implicante $\boldsymbol{c}$ all'insieme C. Elimina da T tutti gli esempi $(\boldsymbol{x}_i, y_i)$ con uscita $y_i = 1$ e $\boldsymbol{x}_i$ coperto dall'implicante $\boldsymbol{c}$. Se T non è vuoto vai al passo 2.
4. Semplifica l'insieme C estraendo un sottoinsieme minimo.

Shadow Clustering (SC)

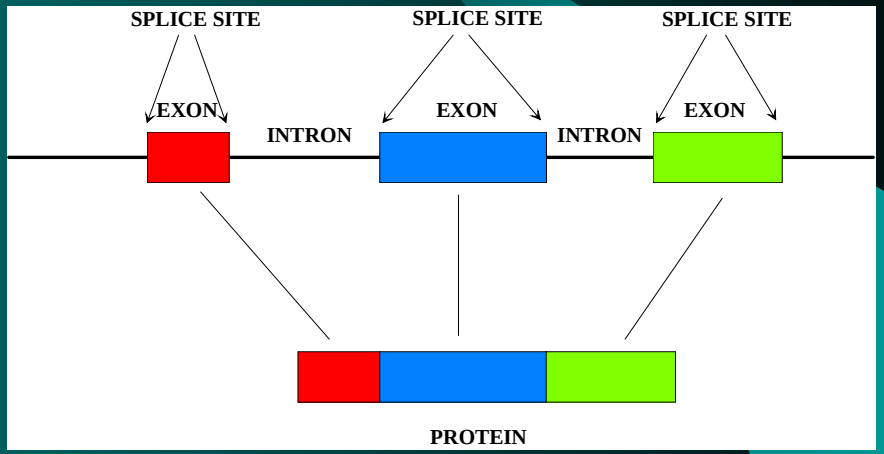
Nel problema di esempio SC produce gli implicanti ‘100 0001’ e ‘000 0111’, che corrispondono alle stesse regole generate da HC.



Applicazione: riconoscimento di splice site nel genoma umano

Gli **splice site** sono punti di una sequenza di DNA in corrispondenza dei quali vengono eliminate basi superflue durante il processo di creazione della proteina in organismi superiori.

Si desidera riconoscere gli splice site, cioè i punti di separazione tra **esoni** (le parti di DNA che vengono mantenute) e **introni** (le parti di DNA ignorate).



Vengono esaminate sequenze di 120 basi nucleotidiche estratte dal DNA umano e contenenti in posizione 60 il segnale considerato. Due casi diversi sono esaminati: i confini esone-introne (EI), detti **donatori**, e i confini introne-esone (IE), detti **accettori**.

Riconoscimento di splice site

43335 sequenze con 120 nucleotidi sono state considerate per generare e testare l'insieme di regole prodotto da SC. Di queste 14026 contengono il segnale EI, mentre 14309 includono il segnale IE. Le restanti 15000 sequenze sono state generate casualmente e servono da controesempi per entrambi i segnali.

Ogni regola prodotta da SC può essere convertita direttamente in una sequenza di simboli, detta **prototipo**, secondo il codice IUB:

```
A → 0111, M = {A,C} → 0011, K = {G,T} → 1100
C → 1011, R = {A,G} → 0101, V = {A,C,G} → 0001
G → 1101, W = {A,T} → 0110, H = {A,C,T} → 0010
T → 1110, S = {C,G} → 1001, D = {A,G,T} → 0100
          Y = {C,T} → 1010, B = {C,G,T} → 1000
```

Un prototipo può essere usato per determinare la presenza del segnale considerato in una nuova sequenza genomica.

Riconoscimento di splice site: risultati

Impiegando metà delle sequenze per generare le regole con SC e l'altra metà per valutare la loro capacità di generalizzazione, abbiamo ottenuto un valore di accuratezza pari a 95.44% per il riconoscimento del segnale EI e 92.94% per il segnale IE.

Questi risultati sono confrontabili con quelli ottenuti applicando i migliori metodi di machine learning a black-box (Support Vector Machines) e sono stati ottenuti senza commettere alcun errore sul training set.

La possibilità di ottenere regole intelligibili apre però la strada ad ulteriori tipi di indagine. L'esecuzione di SC ha infatti condotto a 305 prototipi per il segnale EI e 579 prototipi per il segnale IE.

Riconoscimento di splice site: risultati (cont.)

Il più rilevante per EI è il seguente:

```
1     ...     50          6|0          70     ...     100     ...     120

.....................V.R|GTARG.....................D.........
```

mentre il più rilevante per IE è

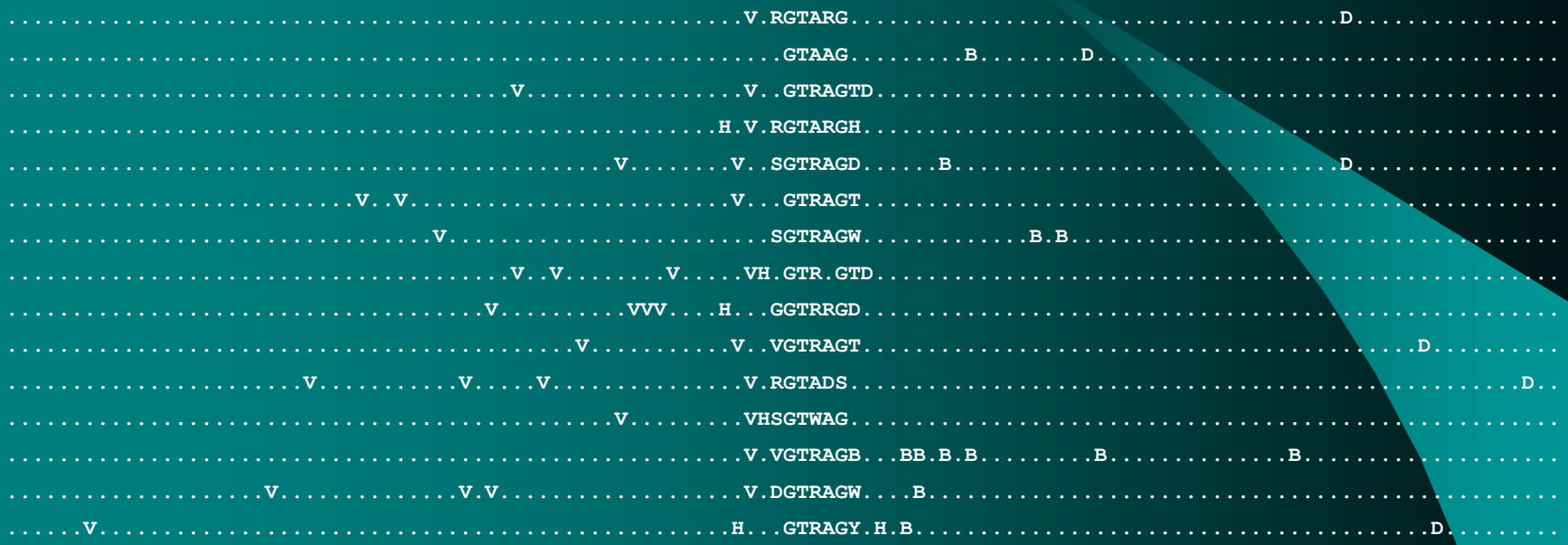
```
1     ...     40          50          6|0     ...     120

.................Y...YYYYBYYYY.YAG|............
```

L'analisi dei prototipi permette di rafforzare risultati già presenti in letteratura, quali l'importanza delle sequenze GTRAGT e YAG per la determinazione dei donatori e degli accettori.

Ma consente anche di estrarre informazioni aggiuntive circa la caratterizzazione delle basi nucleotidiche nelle parti di DNA che precedono e seguono gli splice site.

Riconoscimento di splice site: analisi delle regole

Ad es. le prime 15 regole per il segnale EI sono:



Si può notare una presenza di V (≠ T) nella porzione che precede lo spice site (esone) e una presenza di B (≠ A) nella parte immediatamente successiva (introne).

Riconoscimento di splice site: analisi delle regole (cont.)

Analoghe informazioni possono essere derivate esaminando i prototipi per il segnale IE; ecco i primi 10:

```
.....................................Y...YYYYBYYYY.YAG.....
.....................................H..HB...YYYBYYYYH.YAG.....
.....................................H.........HBBB.H.BYYYYBY.YAG.....
.....................................B.Y.BBYBYYBYYYB.YAG.....
.....................................HHY..YBBYYBBYYY.YAG.....
.....................................Y.B.YHYYBYYYY.YAGV.....
.....................................H...B.BYYYYYBBYYY.YAG.....
.....................................YB.YYYBBYYBHYH.YAG.....
.....................................BY..HYYHYBYYYY.YAG.....
.....................................H.BBBYBHYYBBYYB.YAGD.....
```

Qui si può notare una forte presenza di Y (basi pirimidiniche C, T) nella parte finale dell'introne. Anche questo fenomeno non è adeguatamente evidenziato in letteratura.

Applicazione: analisi di dati da microarray

Analizzando i dati prodotti da esperimenti con microarray si vuole discriminare due diversi stati fisio-patologici di un tessuto (malato/sano, patologia A/patologia B) a partire dai livelli di espressione di un sottoinsieme dei suoi geni.

SC può essere impiegato per questo scopo, permettendo di ottenere un insieme di regole intelligibili che legano tra loro i geni significativi per la discriminazione dei due stati funzionali.

Come sottoprodotto del processo di apprendimento SC identificherà quindi l'insieme dei geni rilevanti per i due stati fisio-patologici in esame.

Dataset analizzati

Sono stati considerati i sequenti insiemi di dati disponibili in rete:

1. Il dataset Leukemia (Golub et al., 1999), che consiste di 72 esperimenti con 7129 genes (47 relativi a tessuti con Leucemia linfoblastica acuta (ALL) and 25 affetti da Leucemia mieloide acuta (AML)).
2. Il dataset Colon (Alon et al., PNAS, 1999), che contiene 62 esempi con 2000 geni (40 relativi a tessuti con cancro al colon e 22 provenienti da tessuti sani).
3. Il dataset Lymphoma (Alizadeh et al., Nature, 2000), che include 81 esempi con 4682 geni (43 relativi a tessuti con Diffuse Large B-cell Lymphoma (DLBCL) and 38 affetti da altre due malattie, B-cell Chronic Lymphocytic Leukemia (B-CLL) e Follicular Lymphoma (FL)).

Risultati ottenuti

Ogni dataset è stato suddiviso in due parti pressoché uguali: la prima di essi è stata usata per effettuare la discriminazione e la selezione dei geni rilevanti, mentre la seconda parte è stata impiegata come test set per stimare l'accuratezza.

SNN è stato confrontato con il metodo di Golub et al. (1999) e con SVM, per valutare la sua qualità. L'accuratezza è stata misurata sia con la tecnica leave-one-out che con il test set.

	Leukemia		Colon		Lymphoma	
Metodo	Leave-one-out	Test set	Leave-one-out	Test set	Leave-one-out	Test set
SNN	**97.4%**	**94.1%**	**83.9%**	**87.1%**	89.6%	**91.7%**
Golub	89.5%	82.3%	80.6%	**87.1%**	89.6%	87.5%
SVM	94.7%	**94.1%**	77.4%	83.9%	**91.7%**	**91.7%**

Insiemi di regole ottenuti da SNN per Leukemia

Regole per la classe ALL:

if $x_{4847} \leq 994$ **then** $y = 1$

if $x_{3258} \leq 2909.5$ AND $x_{4211} > 1470$ **then** $y = 1$

if $x_{1926} \leq 83.5$ AND $x_{3877} \leq 348.5$ **then** $y = 1$

if $x_{1882} \leq 1419.5$ AND $x_{5985} \leq 1866$

Regole per la classe AML:

if $x_{4847} > 994$ **then** $y = 0$

if $x_{2233} \leq 80.5$ AND $x_{3320} > 1341$ **then** $y = 0$

if $x_{1621} \leq 311$ AND $x_{2020} > 1346$ **then** $y = 0$

if $x_{1239} \leq 3559$ AND $x_{6041} > 992.5$ **then** $y = 0$

Insiemi di regole ottenuti da SNN per Colon

Regole per la classe “Cancro al colon presente”:

if $x_{1058} \leq 0.703$ AND $x_{1671} > -0.898$ **then** $y = 1$

if $x_{992} > -1.061$ AND $x_{1423} \leq 1.113$ **then** $y = 1$

if $x_{1884} \leq 0.945$ AND $x_{1917} \leq 1.116$ AND $x_{1998} > -0.298$ **then** $y = 1$

if $x_{194} \leq 1.117$ AND $x_{533} > -1.055$ AND $x_{1998} > -0.298$ **then** $y = 1$

Regole per la classe “Cancro al colon assente”: :

if $x_{839} > -0.286$ AND $x_{1398} \leq 0.024$ **then** $y = 0$

if $x_{739} > 0.399$ AND $x_{1334} \leq 0.364$ **then** $y = 0$

if $x_{267} > 0.293$ AND $x_{1181} \leq 0.303$ **then** $y = 0$

if $x_{249} > 0.449$ AND $x_{1328} > -0.388$ **then** $y = 0$

Insiemi di regole ottenuti da SNN per Lymphoma

Regole per la classe "DLBCL presente":

if $x_{3099} > -0.49$ AND $x_{1258} > -0.07$ **then** $y = 1$

if $x_{1803} \leq 0.165$ AND $x_{2722} > -0.16$ **then** $y = 1$

if $x_{1017} \leq 0.535$ AND $x_{2627} > -0.085$ AND $x_{3791} > -0.345$ **then** $y = 1$

if $x_{1016} \leq 0.32$ AND $x_{1734} \leq 0.375$ AND $x_{2759} > -0.27$ **then** $y = 1$

Regole per la classe "DLBCL non presente":

if $x_{3765} \leq 1.12$ AND $x_{3875} \leq 0.605$ AND $x_{3997} \leq 0.015$ **then** $y = 0$

if $x_{3265} \leq 0.845$ AND $x_{3387} \leq 0.51$ AND $x_{3766} \leq 1.045$ **then** $y = 0$

if $x_{1030} > -0.155$ AND $x_{3533} \leq 1.385$ AND $x_{3767} \leq 0.905$ **then** $y = 0$

if $x_{953} > -0.335$ AND $x_{1418} \leq 0.49$ AND $x_{2515} \leq 0.49$ **then** $y = 0$

Proprietà dei metodi per la generazione di regole

- Non risolvono il problema di apprendimento con una “black-box”, il cui funzionamento è incomprensibile all’utente, ma applicano un insieme di regole intelligibili univocamente determinate.
- Ad ogni regola è assegnato un valore di rilevanza che fornisce una misura della sua affidabilità.
- Attraverso un opportuno mapping nello spazio booleano, l’intero insieme di regole può essere implementato su una FPGA o altro dispositivo che consenta la realizzazione di una rete AND-OR.
- Non risentono di problemi legati alla precisione con la quale i dati sono memorizzati sul supporto fisico, in quanto non vengono eseguite operazioni in virgola mobile.

Proprietà dei metodi per la generazione di regole

- Il processo di apprendimento fissa il numero di bit necessari per codificare le variabili di ingresso; in genere il loro numero è estremamente ridotto in quanto la codifica non suddivide uniformemente il range dei possibili valori.
- Possono trattare in maniera naturale problemi a ingressi misti, alcuni di tipo ordinato (discreto o continuo) e altri di tipo nominale, senza la necessità di trasformare questi ultimi in variabili continue.
- Individuano automaticamente gli ingressi significativi per il problema in esame, fornendo una misura della loro rilevanza.

Vantaggi dei metodi basati sulla sintesi di funzioni booleane

- Presentano valori di accuratezza superiori a quelli delle tecniche per alberi decisionali e paragonabili a quelli dei migliori metodi di apprendimento automatico.
- Il loro costo computazionale è al più quadratico.
- Al momento SC è l'unico algoritmo disponibile per la costruzione di funzioni monotone booleane a partire da esempi.

Nel campo dell'analisi di dati biologici hanno condotto ad interessanti risultati sia nella diagnosi di patologie, che nel riconoscimento di segnali nel genoma, che nell'analisi di dati da microarray.

Conclusioni

Nonostante gli enormi progressi in campo tecnologico, la creazione di modelli (conoscenza) a partire dai dati è un'attività che richiede un deciso contributo dell'esperto.

Metodi automatici d'induzione, sia statistici che di machine learning, sono stati proposti in letteratura. Pur essendo capaci di raggiungere buoni valori di accuratezza, conducono normalmente a modelli black-box.

Le tecniche per la generazione di regole offrono la possibilità di derivare modelli comprensibili all'esperto e pertanto conoscenza utilizzabile anche per ulteriori indagini.

Gli alberi decisionali e i metodi basati sulle funzioni booleane sono le due tipologie più usate e promettenti in tale classe.

Logic Learning Machine, Rulex e Impara

Un'implementazione efficiente di un metodo per la generazione di regole basato sulla sintesi di funzioni booleane (Switching Neural Network) è stata inserita all'interno della suite Rulex® per l'analisi assistita di dati, sviluppata e distribuita da Impara Srl.

Nata nel 2007 come spin-off del CNR per esplorare le possibilità le possibilità commerciali delle più avanzate tecniche di machine learning, costituisce un esempio di trasferimento dei risultati scientifici, anche teorici, per la soluzione di problemi reali.

impara
intelligent machines

rulex
rule generation

Applicazioni attuali delle LLM

Previsione di volumi di vendita in eventi promozionali e mediatici (Danone, Granarolo, De Cecco, …)

Previsione di variazione nei volumi di vendita dovuti alla stagionalità (Lennox)

Analisi delle possibilità di successo di nuovi prodotti (RS Comp.)

Analisi delle perdite inventariali (Novacoop)

Segmentazione della clientela (Poste Mobile)

Studio di fenomeni di instabilità in centrali elettriche (Ansaldo En.)

Diagnostica medica ed analisi omica (IST, Rizzoli, Gaslini,…)

Dataset analizzati nella demo

1. GDS4336 (Zhang et al., *PLOS*, 2012), che contiene 90 campioni relativi all'Adenocarcinoma del pancreas con 33297 probe set (45 relativi a tessuti tumorali e 45 a tessuti sani).
2. GDS4471 (Robinson et al., *Nature*, 2012), che consiste di 76 campioni relativi al Medulloblastoma con 54675 probe sets (39 con tipologia G4 e 37 con altre tipologie).
3. Neuroblastoma (Cangelosi et al., *BMC Bioinformatics*, 2013), che include 182 campioni relativi al Neouroblastoma con 62 probe set e 3 attributi aggiuntivi (51 relativi a pazienti con prognosi sfavorevole e 131 con decorso favorevole).
4. Diabetes (UCI Repository) relativo al diabete femminile negli indiani Pima che include 768 campioni con 8 attributi in ingresso (500 pazienti sani e 268 malati).