Università degli Studi di Milano – Bicocca
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

# Classi di funzioni calcolabili con circuiti a soglia di profondità costante

Appunti per il corso di Apprendimento Automatico

Scritti da Alberto Leporati

Anno Accademico 2006–2007

# Introduzione

Lo studio dei circuiti a soglia come dispositivi per calcolare funzioni Booleane è iniziato negli anni '70, ed è progressivamente andato espandendosi a causa dei risultati ottenuti e dei molti problemi aperti che si sono via via presentati.

Obiettivo di questo documento è caratterizzare alcune classi di funzioni Booleane calcolabili da circuiti a soglia con vincoli sulla profondità o sui pesi, sistematizzando e integrando la miriade di risultati ottenuti fino ad oggi — per la maggior parte sparsi nelle decine di articoli scientifici pubblicati dai ricercatori di questo campo — in un quadro uniforme che renda evidente per ciascun risultato il problema che lo ha ispirato e le conseguenze da esso implicate.

L'utilizzo di neuroni binari a soglia all'interno dei circuiti qui considerati potrebbe far pensare che ci si possa avvantaggiare degli innumerevoli studi effettuati sulle *reti neuronali*. Anche se in linea di principio i circuiti a soglia possono essere considerati reti neuronali a tutti gli effetti, il contesto in cui si collocano i risultati qui esposti è leggermente diverso. La maggior parte (anzi, la quasi totalità) della ricerca svolta attualmente sulle reti neuronali riguarda infatti le reti *cicliche*; di tali reti si studiano le proprietà di convergenza (velocità, numero di stati stabili) e, dal punto di vista della Complessità Strutturale, la quantità di risorse (numero di neuroni, numero e dimensione degli strati, dimensione dei pesi e tempo di calcolo) richieste per computare una certa classe di funzioni, confrontando tali risorse con quelle richieste da altri modelli di computazione parallela, ove possibile, allo scopo di ottenere una conoscenza più approfondita sia dei dispositivi di calcolo che delle funzioni da essi calcolate.

Il modello computazionale trattato in questi appunti, invece, è quello delle reti neuronali *acicliche* (note anche come *circuiti neuronali*); per tali reti non ha senso parlare di "stati stabili" e di "convergenza", e quindi l'interesse che esse destano è legato all'approccio tipico della Complessità Strutturale. In questo senso, le reti acicliche possono essere pensate come delle estensioni dei circuiti combinatori standard (costituiti da porte logiche `AND/OR/NOT`), in cui i singoli elementi del circuito hanno una potenza espressiva maggiore di quella delle porte logiche tradizionali.

Nel Capitolo 1, dopo aver definito il modello di neurone binario a soglia utilizzato (il cosiddetto *threshold gate*) ne vengono studiate le capacità computazionali. In particolare, viene definita la classe $LT_1$ delle funzioni

Booleane calcolabili da un threshold gate con pesi interi senza vincoli sulla dimensione dei pesi, che possono quindi crescere in maniera esponenziale al crescere del numero degli ingressi. Vincolando i pesi ad una crescita al più polinomiale rispetto al numero degli ingressi si ottiene poi la sottoclasse $\widehat{LT}_1$, che risulta propriamente inclusa in $LT_1$ (J. Håstad, Teorema 1.2).

Si dimostrano in particolare due teoremi di chiusura della classe $LT_1$ rispetto alla disgiunzione o alla congiunzione logica di una variabile Booleana $x_n$ con una funzione $f(x_1, x_2, \ldots, x_{n-1}) \in LT_1$ che *non* dipende da $x_n$. Grazie alle leggi di De Morgan, i due teoremi vengono estesi anche ai casi di disgiunzione o congiunzione logica di $f$ con $\overline{x_n}$.

Dal Capitolo 2 in poi, l'attenzione si sposta sui *circuiti a soglia*, ovvero sui circuiti formati da strati di threshold gate. Vengono definite le classi $TC^0$ e $\widehat{TC^0}$ di funzioni Booleane calcolabili da circuiti a soglia di dimensione polinomiale, profondità costante e, rispettivamente, peso esponenziale e polinomiale, e le loro sottoclassi $LT_d$ e $\widehat{LT}_d$ costituite dalle funzioni calcolabili da circuiti di profondità $d$ fissata. Viene mostrato come un risultato di Goldmann, Håstad e Razborov implichi che le classi $\widehat{LT}_d$ ed $LT_d$ formano una gerarchia rispetto alla relazione di inclusione: $\widehat{LT}_d \subseteq LT_d \subseteq \widehat{LT}_{d+1}$ per ogni intero $d \geq 1$, e come da tale gerarchia segua che $TC^0 = \widehat{TC^0}$.

Dal confronto della suddetta gerarchia con il risultato di Håstad esposto nel Capitolo 1 si presenta in maniera naturale il problema della separazione delle classi della gerarchia. Viene definito quindi il concetto di *funzione separatrice*, e vengono esposti alcuni risultati ricavati dalla letteratura: $LT_1 \subset \widehat{LT}_2$ (funzione separatrice: `PARITY`), $\widehat{LT}_2 \subset LT_2$ (per motivi di conteggio), $\widehat{LT}_2 \subset \widehat{LT}_3$ (funzione separatrice: `INNER PRODUCT MOD 2`) e infine $LT_2 \subset \widehat{LT}_3$ (funzione separatrice: ancora `INNER PRODUCT MOD 2`).

Poiché le separazioni successive costituiscono tutte dei problemi aperti, vengono esaminate alcune proprietà delle classi $\widehat{LT}_d$ e $LT_d$ — quali la chiusura rispetto alla negazione, alla digiunzione e alla congiunzione logica di un numero al più polinomiale di funzioni — allo scopo di ottenere una comprensione migliore delle caratteristiche delle funzioni che le compongono.

Nella seconda metà del capitolo il discorso si sposta sui circuiti di profondità 2 e 3. Viene allora dimostrato — con una tecnica che può essere utilizzata in alcuni casi per abbassare la profondità del circuito che calcola una data funzione Booleana — che alcune classi di funzioni (tra cui le funzioni simmetriche) appartengono alla classe $\widehat{LT}_2$. La dimostrazione del fatto che la suddetta tecnica può essere applicata ad ogni funzione Booleana che vale 1 in al più un numero polinomiale di intervalli è un risultato originale dell'autore. Viene poi esposta interamente la dimostrazione di Hajnal, Maass, Pudlák, Szegedy, e Turán della separazione fra le classi $\widehat{LT}_2$ e $\widehat{LT}_3$, e viene presentata una tabella che riporta le profondità necessarie (lower bound) e quelle sufficienti (upper bound $P$-uniformi) per calcolare alcune funzioni originariamente studiate in rapporto con i circuiti

logici.

Nel Capitolo 3 viene introdotto il concetto di *approssimazione* di una funzione Booleana, e viene individuata un'importante classe di funzioni approssimabili, $APP_d$, per le quali l'approssimazione consiste di una combinazione lineare — a coefficienti polinomiali — di un numero al più polinomiale di funzioni appartenenti alla classe $\widehat{LT}_d$. Si passa quindi ad esaminare alcune proprietà che legano le classi $APP_d$ e $\widehat{LT}_d$, scoprendo così che formano una gerarchia molto simile a quella esaminata nel Capitolo 2: $\widehat{LT}_d \subseteq APP_d \subseteq \widehat{LT}_{d+1}$ per ogni intero $d \geq 1$. Una prima conseguenza di ciò è che, definita $APP$ come l'unione infinita di tutte le classi $APP_d$, vale $TC^0 = APP$. Il risultato più importante del capitolo è quello in cui si dimostra che le classi $APP_d$ sono chiuse rispetto alle funzioni Booleane di grado costante; si mostra inoltre come l'ipotesi che una di tali classi sia chiusa rispetto alle funzioni Booleane di grado polinomiale porti al collasso dell'intera gerarchia sulla classe stessa.

Nel Capitolo 4 vengono illustrate le tecniche spettrali. Viene definita la *base di Walsh* $W_n$ per lo spazio vettoriale $\Re^{2^n}$, consistente in $2^n$ funzioni distinte derivate dalla funzione Booleana `PARITY`. Ogni funzione Booleana $f$ di $n$ variabili può allora essere espressa attraverso un'opportuna combinazione lineare degli elementi della base $W_n$: i coefficienti di tale combinazione lineare sono chiamati *coefficienti di Fourier* di $f$. L'insieme dei coefficienti di Fourier di $f$ (lo *spettro* di $f$) individua univocamente la funzione $f$, ovvero ne costituisce una rappresentazione. Definita la $L_1$-norma di $f$ come la somma dei valori assoluti dei coefficienti di Fourier di $f$, il principale risultato del capitolo mostra che ogni funzione Booleana avente $L_1$-norma polinomiale appartiene alla classe $\widehat{LT}_2$. L'importanza delle tecniche spettrali viene ribadita dal fatto che esse introducono un'interpretazione geometrica dei risultati riguardanti la capacità computazionale dei circuiti a soglia, interpretazione che consente molto spesso di semplificare e/o estendere i risultati stessi.

# Indice

# Capitolo 1

# Il neurone formale

In questo primo capitolo, dopo aver definito il modello di neurone binario a soglia (il *threshold gate*) che viene utilizzato nel corso di tutta l'esposizione, cominciamo a studiarne alcune proprietà computazionali, gettando le basi per un analogo studio sui circuiti.

Definita la classe delle funzioni Booleane calcolabili con un threshold gate, vedremo come una semplice interpretazione geometrica delle funzioni che vi appartengono ci consenta di determinare alcune proprietà dell'intera classe; tra i tanti risultati ottenuti, spiccano sugli altri quelli relativi all'equivalenza di tutti i threshold gate ad ingressi e uscite binarie, nonché quelli relativi alla chiusura della classe rispetto alle operazioni logiche di disgiunzione e congiunzione.

Infine, definita la sottoclasse di funzioni calcolabili da un threshold gate con pesi al più polinomiali rispetto al numero degli ingressi, mostriamo la coincidenza di tale sottoclasse con quella delle funzioni calcolate da un altro importante neurone binario a soglia (il *majority gate*), apparentemente meno potente.

## 1.1 Descrizione del modello

I neuroni formali sono dispositivi di calcolo elementari che, opportunamente collegati fra loro, costituiscono i circuiti a soglia. È allora di fondamentale importanza definire tali elementi, in modo da caratterizzarne dal punto di vista matematico il funzionamento e le proprietà delle funzioni da essi calcolate.

**Definizione 1.1.** Un *neurone formale* (Figura 1.1) è un dispositivo computazionale costituito da $n$ ingressi $x_1, x_2, \ldots, x_n$ e un'uscita $y$, caratterizzato dai seguenti parametri interni di funzionamento:

- $n$ numeri reali $w_1, w_2, \ldots, w_n$, detti *pesi*, ciascuno dei quali è associato al corrispondente ingresso;
- un numero reale $w_0$, detto *soglia*. Solitamente, per semplicità e uniformità, non si fa distinzione tra la soglia e gli altri pesi; questo perché si può sempre

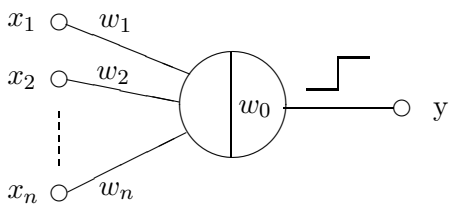



Figura 1.1: *Modello di neurone formale.*

"eliminare" la soglia aggiungendo un ingresso dotato di peso opportuno (si veda di seguito la definizione di funzione calcolata). Tale modifica non altera in alcun modo l'insieme delle funzioni calcolabili dal neurone;

- una *funzione di trasferimento* $\sigma : \Re \to \Re$.

Un neurone riceve i segnali di ingresso da altri neuroni o dall'ambiente esterno; calcola il suo segnale di uscita $y$ effettuando una somma pesata (combinazione lineare) degli ingressi tramite i pesi $w_j$, sottraendo il valore della *soglia* $w_0$, e applicando la funzione di trasferimento $\sigma$ al risultato:

$$y = \sigma\left(\sum_{j=1}^{n} w_j x_j - w_0\right) \tag{1.1}$$

La funzione $y = f(x_1, x_2, \ldots, x_n)$ definita da (1.1) è, per definizione, la *funzione calcolata dal neurone.* ∎

Esistono alcune varianti del modello computazionale appena definito: mentre nel caso più generale tutti i parametri del modello — i pesi e i segnali di ingresso e di uscita — possono essere numeri reali arbitrari, si possono anche studiare neuroni in cui i pesi e/o gli ingressi sono ristretti a valori interi arbitrari, a reali o interi in un intervallo [−M,...,M], o, come spesso accade, agli insiemi $\{-1, 1\}, \{0, 1\}$ o $\{-1, 0, 1\}$. Per il momento assumiamo il modello più generale possibile, in cui i pesi, gli ingressi e le uscite sono numeri reali qualsiasi; successivamente, dato che saremo interessati a calcolare funzioni Booleane, imporremo una restrizione sugli ingressi e sulle uscite, consentendo loro di assumere solamente valori binari.

Le funzioni di trasferimento $\sigma$ più utilizzate nella letteratura sono:

- la funzione a gradino:

$$step(t) = \begin{cases} 1 & \text{se } t \geq 0 \\ 0 & \text{se } t < 0 \end{cases} \tag{1.2}$$

- la funzione a "sigmoide":
$$\sigma(t) = \frac{1}{1+e^{-t}}$$
- varie approssimazioni lineari a tratti o polinomiali della sigmoide; di queste approssimazioni, la più semplice e più utilizzata è la funzione multilineare:
$$\sigma(t) = \begin{cases} 0 & \text{se } t < 0 \\ t & \text{se } 0 \leq t \leq 1 \\ 1 & \text{se } t > 1 \end{cases}$$

Le funzioni di trasferimento menzionate hanno tutte la caratteristica di limitare i segnali di uscita all'intervallo $[0, 1]$; per intervalli di uscita diversi da quest'ultimo, le funzioni di trasferimento devono essere modificate di conseguenza. Ad esempio, qualora si desideri che le uscite del neurone assumano valori in $\{-1, 1\}$, come viene fatto più avanti anche in questo documento, alla funzione *step* va sostituita la funzione *sign*:

$$sign(t) = \begin{cases} +1 & \text{se } t \geq 0 \\ -1 & \text{se } t < 0 \end{cases} \tag{1.3}$$

I neuroni con funzione di trasferimento a gradino $step(t)$ vengono comunemente chiamati *perceptroni*, o *threshold gate* nel caso che gli ingressi siano binari. Il nome *perceptrone* deriva dalla prima applicazione di questo dispositivo, fatta negli anni '60 da parte di F. Rosenblatt a partire da alcune idee dello psicologo D. Hebb sul funzionamento del sistema nervoso centrale, all'acquisizione ed elaborazione delle immagini, nel tentativo di replicare il funzionamento della retina dell'occhio umano. Il nome *threshold gate* si riferisce evidentemente al fatto che tale dispositivo "scatta" (cioè emette un 1) se e solo se la quantità $\sum_{j=1}^{n} w_j x_j$ supera o eguaglia il valore di soglia $w_0$. Infine, un tipo importante di threshold gate è il *majority gate*, in cui tutti i pesi sono uguali a 1, mentre la soglia ha valore pari alla metà del numero degli ingressi. Esso deve il suo nome al fatto che esso calcola la funzione:

$$\texttt{MAJORITY}(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{j=1}^{n} x_j \geq \frac{n}{2}$$

che vale 1, cioè, se e solo se almeno metà degli ingressi valgono 1.

In questi appunti ci occuperemo esclusivamente di threshold gate e majority gate con funzione di trasferimento a gradino, ai quali andranno in ingresso $n$-uple di valori binari. Le funzioni calcolate da tali dispositivi saranno pertanto funzioni Booleane.

Per comodità, supporremo di avere a disposizione, oltre ai segnali di ingresso $x_1, x_2, \ldots, x_n$, anche le loro negazioni $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$, oltre a un segnale costantemente uguale a 0 e a un segnale costantemente uguale a 1.

## 1.2 Funzioni Booleane e loro rappresentazione

Richiamiamo rapidamente le ben note nozioni di variabile binaria e funzione Booleana. Una variabile binaria è una variabile che assume valori in un insieme di due elementi; senza perdere di generalità, assumeremo per ora l'insieme $\{0,1\}$. Una funzione Booleana ad $n$ ingressi è una qualsiasi funzione $f(x_1, x_2, \ldots, x_n) : \{0,1\}^n \to \{0,1\}$.

Come è noto, una $n$-upla di variabili binarie può assumere esattamente $2^n$ configurazioni distinte, e può essere pensata come un vettore a $n$ componenti binarie. La seguente nozione ci permetterà in seguito di interpretare dal punto di vista geometrico i risultati ottenuti.

**Definizione 1.2.** Dati due vettori $\mathbf{u}$ e $\mathbf{v}$ a $n$ componenti (non necessariamente binarie), si definisce *distanza di Hamming* tra $\mathbf{u}$ e $\mathbf{v}$ il numero di componenti corrispondenti in cui $\mathbf{u}$ e $\mathbf{v}$ differiscono:

$$d_H(\mathbf{u}, \mathbf{v}) = \#\{i : (\mathbf{u})_i \neq (\mathbf{v})_i\}$$

■

La distanza di Hamming è una misura della differenza (o somiglianza) di due vettori, e può essere utilizzata per definire una sorta di *correlazione* tra vettori o funzioni (si veda a questo proposito il Capitolo 4).

Preso lo spazio vettoriale $\Re^n$ con la sua base canonica, e fissata l'usuale metrica euclidea, è possibile costruire un $n$-cubo di lato 1 avente un vertice nell'origine e gli $n$ lati che si dipartono da tale vertice coincidenti con i versori della base canonica. Si possono allora associare biunivocamente le $2^n$ configurazioni binarie con i vertici dell'$n$-cubo, in modo tale che:

- al vertice sia associata la $n$-upla $(0, 0, \ldots, 0)$;
- ad ogni versore della base sia associata una e una sola variabile;
- se una coppia di vertici è unita da un lato, le corrispondenti $n$-uple binarie hanno distanza di Hamming uguale a 1, e la componente che varia da una $n$-upla all'altra è quella corrispondente alla variabile associata al versore della base parallelo al lato.

Una funzione Booleana a $n$ ingressi può essere vista, pertanto, come l'assegnamento di un'etichetta (0 o 1) ai vertici dell'$n$-cubo, intendendo ovviamente che l'etichetta di un vertice è 1 se e solo se la funzione Booleana, per quella configurazione di ingresso, vale 1.

Solitamente, anziché considerare una singola funzione Booleana a $n$ ingressi, si considera una *famiglia* $\boldsymbol{\mathcal{F}}$ di funzioni Booleane, ovvero una successione $\{f_n\}_{n \in \boldsymbol{N}}$ di funzioni Booleane che hanno in maniera evidente una "forma" comune. Ad esempio, si parla solitamente della funzione `AND` (congiunzione logica) intendendo

in realtà la famiglia $\{\texttt{AND}_n\}$ di congiunzioni logiche di $n$ variabili, al variare di $n$ nell'insieme dei numeri naturali. Poiché un threshold gate ha un numero fissato di ingressi, per calcolare una famiglia $\boldsymbol{\mathcal{F}}$ di funzioni Booleane ci vorrà una famiglia $\{T_n\}_{n\in\boldsymbol{\mathcal{N}}}$ di threshold gate, in cui chiaramente ogni singolo threshold gate $T_n$ calcola la corrispondente funzione $f_n$.

## 1.3 Il threshold gate

La funzione Booleana calcolata da un threshold gate a $n$ ingressi $x_1, x_2, \ldots, x_n$ è, come abbiamo visto:

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{se } \sum_{j=1}^{n} w_j x_j \geq w_0 \\ 0 & \text{altrimenti} \end{cases}$$

Questa forma deriva facilmente dall'equazione (1.2), sostituendo a $t$ la combinazione lineare degli ingressi.

Detti $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ e $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$ rispettivamente il vettore colonna degli ingressi e il vettore colonna dei pesi associati agli ingressi, questa formula può essere riscritta in maniera più compatta:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{se } \mathbf{w}^T\mathbf{x} \geq w_0 \\ 0 & \text{altrimenti} \end{cases}$$

Come si può notare, il vettore dei pesi $\mathbf{w}$ e la soglia $w_0$ individuano univocamente un iperpiano nello spazio $\Re^n$: tale iperpiano è l'insieme dei punti $\mathbf{x}$ di $\Re^n$ che soddisfano l'equazione $\mathbf{w}^T\mathbf{x} = w_0$. La funzione computata dal threshold gate, quindi, vale 1 se il vettore di ingresso corrisponde a un punto di $\Re^n$ situato nel semispazio affine (compreso l'iperpiano stesso) individuato dall'iperpiano e dall'orientazione del vettore dei pesi, 0 altrimenti; in altre parole, un threshold gate individua un iperpiano nello spazio dei valori di ingresso, che divide le due controimmagini dei valori di uscita. Per questo motivo, le funzioni calcolate dai threshold gate, oltre che *funzioni threshold*, vengono anche dette *linearmente separabili*.

Una prima domanda che ci possiamo porre è se sia necessario utilizzare per i pesi tutta la capacità espressiva dei numeri reali, e non siano invece sufficienti i numeri razionali. Si noti poi che utilizzare pesi razionali significa, in ultima analisi, utilizzare pesi *interi*, poiché è sempre possibile moltiplicare i membri della disuguaglianza $\mathbf{w}^T\mathbf{x} \geq w_0$ per un'opportuna potenza di 10 senza alterarne l'insieme delle soluzioni. Il primo teorema che enunciamo risponde proprio a questa domanda, ed è stato formulato da Muroga nel 1971 [11].

**Teorema 1.1.** *Ogni funzione threshold di n variabili può essere computata da un threshold gate con pesi interi $w_i$ tali che $|w_i| \leq (n+1)^{(n+1)/2}/2^n$ per ogni $i = 0, \ldots, n$.* ∎

Quindi, quando si parla di threshold gate, si può sempre supporre che i pesi e la soglia siano interi senza perdere di generalità. Da ora in poi si utilizzeranno indifferentemente pesi interi e/o razionali, sottintendendo che ci si può sempre ricondurre a pesi interi nel modo suddetto.

Un particolare che salta subito all'occhio nell'enunciato del Teorema 1.1 è che il vincolo sui pesi non è molto stretto: è sufficiente che i pesi siano minori di una quantità che cresce in maniera esponenziale al crescere del numero degli ingressi. Adottiamo allora il punto di vista tipico della Teoria della Complessità, che considera praticamente realizzabili solo le risorse polinomiali, e ci chiediamo se sia possibile imporre ai pesi un vincolo polinomiale. Ci chiediamo, cioè, se sia possibile calcolare tutte le funzioni threshold tramite threshold gate in cui si abbia:

$$\max_{0\leq i\leq n} |w_i| \leq p(n)$$

per un opportuno polinomio $p(n)$. La risposta negativa è stata fornita da J. Håstad nel 1992 [9], ed è espressa dal seguente teorema.

**Teorema 1.2.** *Per infiniti n, esistono funzioni threshold di n variabili che richiedono pesi dell'ordine di $n^{n/2}/2^n$ per poter essere computate da un singolo threshold gate.* ∎

Abbiamo quindi un primo risultato di separazione tra classi di funzioni calcolate dai threshold gate (che possono essere visti, al limite, come circuiti costituiti da un solo elemento): la classe delle funzioni calcolate dai threshold gate con pesi polinomiali *è strettamente contenuta* nella classe delle funzioni calcolate dai threshold gate senza alcuna limitazione sui pesi.

Diamo ora alcune definizioni che saranno utili nel corso di tutta la presente esposizione.

**Definizione 1.3.** Chiamiamo $\widehat{LT}_1$ la classe delle funzioni calcolate dai threshold gate con pesi polinomiali, e $LT_1$ la classe delle funzioni calcolate dai threshold gate con pesi illimitati[1]. Chiamiamo inoltre $BIN_n$ l'insieme delle funzioni Booleane a $n$ argomenti, e $BIN$ l'insieme di tutte le funzioni Booleane:

$$BIN = \bigcup_{n=1}^{+\infty} BIN_n$$

∎

Alla luce delle definizioni appena date possiamo dire che il Teorema 1.2 afferma che:

$$\widehat{LT}_1 \subset LT_1$$

[1]Questi sono i nomi comunemente usati in letteratura, oltre a $\widehat{TC}_1^0$ e $TC_1^0$ rispettivamente. Mentre è immediato dire che $LT$ sta per "Linear Threshold" e che $TC$ sta per "Threshold Circuit", per il significato da attribuire alle costanti 0 e 1 si rimanda al secondo capitolo, quando verranno definite le funzioni calcolate dai circuiti a soglia.

Inoltre, per definizione, vale ovviamente:

$$LT_1 \subseteq BIN$$

Meno banale è chiedersi se vale l'inclusione stretta, ovvero se esiste una funzione Booleana che non sia anche una funzione threshold. La risposta è affermativa, come si può facilmente vedere considerando la funzione:

$$\texttt{PARITY}(x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n$$

Il simbolo $\oplus$ rappresenta l'operatore logico di disgiunzione esclusiva, e la funzione `PARITY` ne è quindi la naturale estensione a $n$ variabili. Il nome della funzione deriva dal fatto che essa vale 1 per tutte e sole le configurazioni di variabili in ingresso che contengono un numero *dispari* di 1. Vale il seguente teorema.

**Teorema 1.3.** *La funzione* `PARITY` *non è linearmente separabile.*

*Dimostrazione.* Si consideri l'$n$-cubo delle configurazioni di ingresso, e si considerino i seguenti iperpiani:

$$\mathbf{1}^T\mathbf{x} = 2i + 1 \qquad i = 0, 1, \ldots, \left\lfloor \frac{n-1}{2} \right\rfloor \tag{1.4}$$

dove $\mathbf{1}^T$ rappresenta un vettore a $n$ componenti tutte uguali a 1, e $\lfloor \frac{n-1}{2} \rfloor$ indica il più grande numero intero minore o uguale a $\frac{n-1}{2}$.

È facile rendersi conto che tutti i vertici dell'$n$-cubo corrispondenti a configurazioni di ingresso $x_1, x_2, \ldots, x_n$ per le quali $\texttt{PARITY}(x_1, x_2, \ldots, x_n) = 1$ giacciono su esattamente uno degli iperpiani proposti. È facile altresì rendersi conto che tutti i vertici dell'$n$-cubo corrispondenti a configurazioni di ingresso $x_1, x_2, \ldots, x_n$ per le quali $\texttt{PARITY}(x_1, x_2, \ldots, x_n) = 0$ giacciono su esattamente uno dei seguenti iperpiani:

$$\mathbf{1}^T\mathbf{x} = 2i \quad i = 0, 1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor \tag{1.5}$$

e che *tutti* gli iperpiani qui presi in considerazione sono tra loro paralleli. L'osservazione del fatto che gli iperpiani dell'equazione (1.4) sono frapposti a quelli dell'equazione (1.5) è sufficiente ad affermare che non esiste alcun iperpiano che separa i vertici dell'$n$-cubo in cui la funzione vale 1 da quelli in cui la funzione vale 0. ∎

Dato che la funzione `PARITY` non è linearmente separabile (o, che è la stessa cosa, non è una funzione threshold), si ha che vale la seguente catena di inclusioni strette:

$$\widehat{LT}_1 \subset LT_1 \subset BIN$$

Poiché non tutte le funzioni Booleane sono funzioni threshold, ci chiediamo se lo sono per lo meno alcune funzioni di base che consideriamo "irrinunciabili". I teoremi che seguono mostrano come alcune di queste funzioni possono essere calcolate dai threshold gate.

**Teorema 1.4.** *Le seguenti funzioni Booleane a n ingressi:*

- $f(x_1, \ldots, x_n) \equiv 1$
- $f(x_1, \ldots, x_n) \equiv 0$
- $f(x_1, \ldots, x_n) = \bigvee_{i=1}^{n} x_i$
- $f(x_1, \ldots, x_n) = \bigwedge_{i=1}^{n} x_i$

*sono funzioni threshold (e appartengono a* $\widehat{LT}_1$*).*

*Dimostrazione.* Per calcolare le funzioni dell'enunciato basta prendere un threshold gate a $n$ ingressi, porre i pesi tutti uguali a 1, e porre la soglia rispettivamente uguale a 0, $n+1$, 1, $n$. ∎

**Teorema 1.5.** *La funzione Booleana unaria* $f(x) = x$ *appartiene a* $\widehat{LT}_1$*.*

*Dimostrazione.* Sia $T$ un threshold gate a un ingresso, con peso $w$ e soglia $w_0$ tali che $0 < w_0 \leq w$, e sia:

$$f_T(x) = \begin{cases} 1 & \text{se } w \cdot x \geq w_0 \\ 0 & \text{se } w \cdot x < w_0 \end{cases}$$

la funzione calcolata da $T$.

Per $x = 1$ si ha $w \cdot x = w \geq w_0$, per cui $f_T(1) = 1$. Per $x = 0$ si ha $w \cdot x = 0 < w_0$, per cui $f_T(0) = 0$. ∎

**Teorema 1.6.** *La funzione Booleana a n argomenti* $f(x_1, x_2, \ldots, x_n) = x_i$ *(proiezione dell'i-esima componente del vettore in ingresso) appartiene a* $\widehat{LT}_1$*.*

*Dimostrazione.* Si consideri un threshold gate $T$ a $n$ ingressi con rispettivi pesi:

$$\begin{aligned} w_i &= 1 \\ w_j &= 0 \qquad \forall j \neq i \end{aligned}$$

e avente soglia $w_0 = 1$. È facile verificare che $T$ calcola la funzione richiesta. ∎

Definiamo poi due funzioni che saranno molto utili in seguito: esse sono probabilmente, a parte le funzioni $f \equiv 1$ e $f \equiv 0$ del Teorema 1.4, le più semplici funzioni threshold immaginabili.

**Definizione 1.4.** Indichiamo con $T^n_{\geq k}$ e $T^n_{\leq k}$ le seguenti funzioni Booleane di $n$ variabili Booleane:

$$T^n_{\geq k}(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{j=1}^{n} x_j \geq k$$

$$T^n_{\leq k}(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{j=1}^{n} x_j \leq k$$

Ovvero, $T^n_{\geq k}$ vale 1 se e solo se *almeno* $k$ degli $n$ ingressi assumono valore 1, e $T^n_{\leq k}$ vale 1 se e solo se *al più* $k$ degli $n$ ingressi assumono valore 1. ∎

È semplicissimo mostrare che vale il seguente teorema.

**Teorema 1.7.** *Le funzioni $T^n_{\geq k}$ e $T^n_{\leq k}$ appartengono a $\widehat{LT}_1$.*

*Dimostrazione.* Per calcolare $T^n_{\geq k}$, basta prendere un threshold gate a $n$ ingressi, con pesi $w_1 = w_2 = \ldots = w_n = 1$ e soglia $w_0 = k$. Per quanto riguarda $T^n_{\leq k}$, basta osservare che:

$$T^n_{\leq k}(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{j=1}^{n} x_j \leq k \iff -\sum_{j=1}^{n} x_j \geq -k$$

Pertanto un threshold gate a $n$ ingressi con pesi $w_1 = w_2 = \ldots = w_n = -1$ e soglia $w_0 = -k$ è sufficiente per calcolare $T^n_{\leq k}$. ∎

Richiamiamo ora due definizioni universalmente utilizzate nello studio delle funzioni Booleane.

**Definizione 1.5.** Un *letterale* $X_i$ è una variabile Booleana diretta ($x_i$) o negata ($\overline{x_i}$). Data una funzione Booleana $f(x_1, x_2, \ldots, x_n) \in BIN_n$, un *mintermine* è una congiunzione logica $X_1 \wedge X_2 \wedge \ldots \wedge X_n$ di letterali per i quali vale $f(X_1, X_2, \ldots, X_n) = 1$. Un mintermine, cioè, è una rappresentazione di una delle $2^n$ configurazioni di ingresso per la quale la funzione vale 1. ∎

È ben noto dallo studio delle funzioni Booleane che ogni funzione Booleana $f$ può essere espressa come disgiunzione logica di tutti i suoi mintermini; si dice in tal caso che la funzione è espressa in forma normale disgiuntiva (DNF). Questa proprietà tornerà utile nel prossimo capitolo, quando verranno esposte alcune tecniche elementari per calcolare le funzioni Booleane con circuiti a soglia di profondità 2. Per il momento ci limitiamo ad enunciare il seguente teorema, in cui si mostra che ogni mintermine è calcolabile da un threshold gate.

**Teorema 1.8.** *Sia $f(x_1, x_2, \ldots, x_n)$ una funzione Booleana che vale 1 in corrispondenza di una sola delle $2^n$ configurazioni di ingresso; allora $f \in \widehat{LT}_1$.*

*Dimostrazione.* Siano $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ le variabili che assumono valore 1 nella configurazione prescelta, e $x_{i_k+1}, x_{i_k+2}, \ldots, x_{i_n}$ quelle che assumono valore 0. È sufficiente allora prendere un threshold gate a $n$ ingressi, con pesi $w_{i_1} = w_{i_2} = \ldots = w_{i_k} = 1$, $w_{i_k+1} = w_{i_k+2} = \ldots = w_{i_n} = -1$ e soglia $w_0 = k$. Con tali pesi, la quantità $\sum_{j=1}^{n} w_j x_j$ raggiunge il valore della soglia solo in corrispondenza della configurazione prescelta. Essendo i pesi e la soglia del threshold gate costanti, $f \in \widehat{LT}_1$.

Dal punto di vista geometrico, il threshold gate proposto corrisponde a un iperpiano che interseca l'$n$-cubo esattamente nel vertice corrispondente alla configurazione prescelta. ∎

Il teorema che segue consente di dire che la classe delle funzioni calcolate dai threshold gate è chiusa rispetto all'operazione logica di negazione. La dimostrazione è costruttiva, e ha una semplice interpretazione geometrica: se $f$ è la funzione Booleana calcolata da un threshold gate $T$, il quale individua nello spazio degli ingressi un iperpiano $\boldsymbol{\pi}$, per ottenere un threshold gate $T'$ che calcola $\overline{f}$ è sufficiente considerare l'iperpiano $\boldsymbol{\pi}'$ che coincide con $\boldsymbol{\pi}$ e che individua il semispazio affine complementare a quello individuato da $\boldsymbol{\pi}$. In pratica, è sufficiente cambiare di segno i pesi di $T$. Poiché i pesi sono polinomiali o meno indipendentemente dal loro segno, avremo in particolare che valgono le seguenti affermazioni:

$$\text{se } f \in \widehat{LT}_1 \text{ allora } \overline{f} \in \widehat{LT}_1$$
$$\text{se } f \in LT_1 \setminus \widehat{LT}_1 \text{ allora } \overline{f} \in LT_1 \setminus \widehat{LT}_1$$

**Teorema 1.9.** *Sia $T$ un threshold gate a $n$ ingressi avente pesi $w_1, w_2, \ldots, w_n$ e soglia $w_0$, e sia $f(x_1, x_2, \ldots, x_n)$ la funzione Booleana computata da $T$. Esiste un threshold gate $T'$ avente pesi $w'_1, \ldots, w'_n$ e soglia $w'_0$ che calcola la funzione $\overline{f(x_1, x_2, \ldots, x_n)}$.*

*Dimostrazione.* Per il Teorema 1.15, si può supporre che $f(x_1, x_2, \ldots, x_n) = 1$ sse $\sum_{i=1}^{n} w_i x_i > w_0$. Posto $w'_i = -w_i \quad \forall i = 0, 1, \ldots, n$, la funzione computata da $T'$ vale 1 sse $\sum_{i=1}^{n} w'_i x_i - w'_0 \geq 0$, ovvero sse $-\left(\sum_{i=1}^{n} w_i x_i - w_0\right) \geq 0$, ovvero sse $\sum_{i=1}^{n} w_i x_i \leq w_0$ (ovvero sse $\sum_{i=1}^{n} w_i x_i < w_0$).

Pertanto, la funzione calcolata da $T'$ vale 1 sse $f(x_1, x_2, \ldots, x_n) = 0$, cioè la funzione calcolata da $T'$ è $\overline{f(x_1, x_2, \ldots, x_n)}$. ∎

Alla luce del risultato espresso da questo teorema, è naturale chiedersi se $\widehat{LT}_1$ e $LT_1$ sono chiusi anche rispetto alla disgiunzione e alla congiunzione logica. Considerando per il momento solo la disgiunzione logica, ciò equivale a chiedersi se, dati due threshold gate $T_1$ e $T_2$, che calcolano rispettivamente le funzioni Booleane $f_1$ e $f_2$, esiste un threshold gate $T$ che calcola la funzione $f_1 \vee f_2$. La risposta (negativa) a questa domanda è data dal seguente teorema.

**Teorema 1.10.** *Siano $f_1(x_1, x_2, \ldots, x_n)$ e $f_2(x_1, x_2, \ldots, x_n)$ due funzioni Booleane appartenenti a $\widehat{LT}_1$. La funzione Booleana:*

$$f(x_1, x_2, \ldots, x_n) = f_1(x_1, x_2, \ldots, x_n) \vee f_2(x_1, x_2, \ldots, x_n)$$

*non appartiene necessariamente a $LT_1$ (e quindi tantomeno a $\widehat{LT}_1$).*

*Dimostrazione.* Indicando per comodità con $\mathbf{x}$ il vettore $(x_1, x_2, \ldots, x_n)$ degli ingressi, con $\mathbf{0}_n$ il vettore nullo in $\Re^n$ e con $\mathbf{1}_n$ il vettore in $\Re^n$ avente tutte le

componenti uguali a 1, si considerino le due funzioni Booleane:

$$f_1(\mathbf{x}) = 1 \iff \mathbf{x} = \mathbf{0}_n$$
$$f_2(\mathbf{x}) = 1 \iff \mathbf{x} = \mathbf{1}_n$$

Dato che sia $f_1$ che $f_2$ valgono 1 in corrispondenza di una sola delle $2^n$ configurazioni di ingresso, per il Teorema 1.8 esse appartengono alla classe $\widehat{LT}_1$.

Si noti inoltre che le due configurazioni $\mathbf{0}_n$ e $\mathbf{1}_n$ corrispondono a due vertici opposti dell'$n$-cubo, così che non esiste alcun iperpiano che separi l'insieme $\{\mathbf{0}_n, \mathbf{1}_n\}$ dagli altri vertici. Allora la funzione:

$$f(x_1, x_2, \ldots, x_n) = f_1(x_1, x_2, \ldots, x_n) \vee f_2(x_1, x_2, \ldots, x_n)$$

che vale 1 su un vertice o sull'altro (o su entrambi contemporaneamente) non è linearmente separabile, ovvero non appartiene a $LT_1$. ∎

Un teorema analogo vale per la congiunzione.

**Teorema 1.11.** *Siano $f_1(x_1, x_2, \ldots, x_n)$ e $f_2(x_1, x_2, \ldots, x_n)$ due funzioni Booleane appartenenti a $\widehat{LT}_1$. La funzione Booleana:*

$$f(x_1, x_2, \ldots, x_n) = f_1(x_1, x_2, \ldots, x_n) \wedge f_2(x_1, x_2, \ldots, x_n)$$

*non appartiene necessariamente a $LT_1$ (e quindi tantomeno a $\widehat{LT}_1$).*

*Dimostrazione.* Se così non fosse, poiché per le leggi di De Morgan vale:

$$f_1 \vee f_2 = \overline{\overline{f_1} \wedge \overline{f_2}}$$

per il Teorema 1.9 si avrebbe che la disgiunzione logica di due funzioni threshold è sempre una funzione threshold, contraddicendo il teorema precedente. ∎

Anche se la disgiunzione e la congiunzione logica di due funzioni threshold non sono *sempre* funzioni threshold, può essere utile individuare i casi in cui ciò effettivamente avviene. Un caso molto particolare è, ad esempio, quello in cui si ha una funzione threshold $f(x_1, x_2, \ldots, x_{n-1})$ che *non dipende* dalla variabile Booleana $x_n$, e si vuole calcolare la funzione $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1}) \vee x_n$. A questo proposito vale il seguente teorema.

**Teorema 1.12.** *Sia $T$ un threshold gate avente ingressi $x_1, x_2, \ldots, x_{n-1}$, pesi $w_1, w_2, \ldots, w_{n-1}$ e soglia $w_0$, e sia $f(x_1, x_2, \ldots, x_{n-1})$ la funzione calcolata da $T$. Esiste un threshold gate $T'$ avente ingressi $x_1, x_2, \ldots, x_n$, pesi $w_1, w_2, \ldots, w_{n-1}, \overline{w}_n$ e soglia $w_0$ che calcola la funzione $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1}) \vee x_n$.*

*Dimostrazione.* Sia $T'$ un threshold gate a $n$ ingressi avente pesi $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_n$ e soglia $\overline{w}_0$. Vogliamo che sia

$$\sum_{i=1}^{n} \overline{w}_i x_i < \overline{w}_0 \quad \text{sse} \quad \sum_{i=1}^{n-1} w_i x_i < w_0 \quad \text{e} \quad x_n = 0$$

Supponiamo che sia $\overline{w}_i = w_i \quad \forall i = 1, \ldots, n-1$; se così fosse, il nuovo neurone sarebbe ottenibile da $T$ aggiungendo semplicemente il nuovo ingresso con un peso opportuno. Sotto tale ipotesi, la condizione precedente diventa:

$$\sum_{i=1}^{n-1} w_i x_i + \overline{w}_n x_n < \overline{w}_0 \quad \text{sse} \quad \sum_{i=1}^{n-1} w_i x_i < w_0 \quad \text{e} \quad x_n = 0$$

Possiamo distinguere due casi, a seconda del valore assunto dal nuovo ingresso:

$x_n = 0$ : $g(x_1, x_2, \ldots, x_n) = 0$ sse $f(x_1, x_2, \ldots, x_{n-1}) = 0$ sse $\sum_{i=1}^{n-1} w_i x_i < \overline{w}_0$ sse $\sum_{i=1}^{n-1} w_i x_i < w_0$ sse $\overline{w}_0 = w_0$. Come ci si poteva aspettare, se si mantiene a 0 il nuovo ingresso senza variare i pesi associati agli ingressi $x_1, x_2, \ldots, x_{n-1}$, l'unica soglia che calcola correttamente la funzione $g$ (che in questo caso coincide con $f$) è $w_0$.

$x_n = 1$ : in questo caso deve essere $g(x_1, x_2, \ldots, x_n) = 1$, cioè $\sum_{i=1}^{n-1} w_i x_i + \overline{w}_n - \overline{w}_0 \geq 0$ che, per il caso precedente, diventa $\sum_{i=1}^{n-1} w_i x_i + \overline{w}_n - w_0 \geq 0$. Pertanto deve essere $\sum_{i=1}^{n-1} w_i x_i - w_0 \geq -\overline{w}_n$, ovvero $\overline{w}_n \geq -\left(\sum_{i=1}^{n-1} w_i x_i - w_0\right)$, cioè $\overline{w}_n \geq w_0 - \sum_{i=1}^{n-1} w_i x_i$. Questa disuguaglianza deve valere per ogni possibile valore di $\sum_{i=1}^{n-1} w_i x_i$, pertanto basterà prendere $\overline{w}_n$ in modo tale che sia

$$\begin{aligned}
\overline{w}_n &\geq \max_{(x_1, x_2, \ldots, x_{n-1})} \left( w_0 - \sum_{i=1}^{n-1} w_i x_i \right) \\
&= \max_{\{(x_1, x_2, \ldots, x_{n-1}) | f(x_1, x_2, \ldots, x_{n-1}) = 0\}} \left( w_0 - \sum_{i=1}^{n-1} w_i x_i \right) \\
&= w_0 - \min_{\{(x_1, x_2, \ldots, x_{n-1}) | f(x_1, x_2, \ldots, x_{n-1}) = 0\}} \sum_{i=1}^{n-1} w_i x_i
\end{aligned}$$

Si noti infine che se $f \in \widehat{LT}_1$, si può sempre scegliere un valore di $\overline{w}_n$ che verifichi la disuguaglianza e che sia polinomiale, così che $g \in \widehat{LT}_1$. ∎

Un teorema analogo vale per la congiunzione logica: data una qualunque funzione threshold $f(x_1, x_2, \ldots, x_{n-1})$, è possibile costruire un threshold gate che ne calcoli la congiunzione con una *nuova* variabile Booleana $x_n$.

**Teorema 1.13.** *Sia $T$ un threshold gate avente ingressi $x_1, x_2, \ldots, x_{n-1}$, pesi $w_1, w_2, \ldots, w_{n-1}$ e soglia $w_0$, e sia $f(x_1, x_2, \ldots, x_{n-1})$ la funzione calcolata da $T$. Esiste un threshold gate $T'$ avente ingressi $x_1, x_2, \ldots, x_n$, pesi $w_1, w_2, \ldots, w_{n-1}, \overline{w}_n$ e soglia $\overline{w}_0$ che calcola la funzione $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1}) \wedge x_n$.*

*Dimostrazione.* Sia $T'$ un threshold gate a $n$ ingressi avente pesi $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_n$ e soglia $\overline{w}_0$. Vogliamo che sia

$$\sum_{i=1}^{n} \overline{w}_i x_i \geq \overline{w}_0 \quad \text{sse} \quad \sum_{i=1}^{n-1} w_i x_i \geq w_0 \quad \text{e} \quad x_n = 1 \tag{1.6}$$

Supponiamo che sia $\overline{w}_i = w_i \quad \forall i = 1, \ldots, n-1$; se così fosse, il nuovo neurone sarebbe ottenibile da $T$ aggiungendo semplicemente il nuovo ingresso con un peso opportuno e modificando la soglia. Sotto tale ipotesi, la (1.6) diventa:

$$\sum_{i=1}^{n-1} w_i x_i + \overline{w}_n x_n \geq \overline{w}_0 \quad \text{sse} \quad \sum_{i=1}^{n-1} w_i x_i \geq w_0 \quad \text{e} \quad x_n = 1$$

Possiamo distinguere due casi, a seconda del valore assunto dal nuovo ingresso:

$$\begin{aligned} x_n = 0: \quad & g(x_1, x_2, \ldots, x_n) = 0 \text{ sse } \sum_{i=1}^{n} \overline{w}_i x_i < \overline{w}_0 \text{ sse } \sum_{i=1}^{n-1} \overline{w}_i x_i < \overline{w}_0 \text{ sse} \\ & \sum_{i=1}^{n-1} w_i x_i < \overline{w}_0 \\ x_n = 1: \quad & g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1}) \text{ sse } \sum_{i=1}^{n} \overline{w}_i x_i \geq \overline{w}_0 \text{ sse} \\ & \sum_{i=1}^{n-1} \overline{w}_i x_i + \overline{w}_n \geq \overline{w}_0 \text{ sse } \sum_{i=1}^{n-1} w_i x_i + \overline{w}_n \geq \overline{w}_0 \end{aligned}$$

Quindi,

$$\sum_{i=1}^{n-1} w_i x_i < \overline{w}_0 \leq \sum_{i=1}^{n-1} w_i x_i + \overline{w}_n$$

da cui si deduce che deve essere $\overline{w}_n > 0$. Inoltre, se $x_n = 1$, si ha:

$1 =$ $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1})$ sse $\sum_{i=1}^{n-1} w_i x_i \geq \overline{w}_0 - \overline{w}_n$
e $\sum_{i=1}^{n-1} w_i x_i \geq w_0$, cioè sse $\sum_{i=1}^{n-1} w_i x_i \geq max\{\overline{w}_0 - \overline{w}_n, w_0\}$.
$max\{\overline{w}_0 - \overline{w}_n, w_0\} = w_0$ (vedere il prossimo caso) e quindi questo caso non impone vincoli sul valore di $\overline{w}_0$ e di $\overline{w}_n$. In altre parole, viene qui riconfermato ciò che era già stato detto nell'equazione (1.6).

$0 =$ $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1})$ sse $\sum_{i=1}^{n-1} w_i x_i \geq \overline{w}_0 - \overline{w}_n$ e
$\sum_{i=1}^{n-1} w_i x_i < w_0$, cioè sse $\overline{w}_0 - \overline{w}_n \leq \sum_{i=1}^{n-1} w_i x_i < w_0$

Allora, ricapitolando,

$$\overline{w}_0 - \overline{w}_n \leq \sum_{i=1}^{n-1} w_i x_i \qquad (\text{se } x_n = 1)$$

e

$$\sum_{i=1}^{n-1} w_i x_i < \overline{w}_0 \qquad (\text{se } x_n = 0)$$

Cioè

$$\overline{w}_0 - \overline{w}_n \leq \sum_{i=1}^{n-1} w_i x_i < \overline{w}_0$$

Detti $v_0$ e $v_1$ rispettivamente il minimo e il massimo dei valori assunti da $\sum_{i=1}^{n-1} w_i x_i$ al variare di tutte le $2^{n-1}$ possibili configurazioni di $x_1, \ldots, x_{n-1}$, basta porre, ad esempio, $\overline{w}_0 = v_1 + 1$ e $\overline{w}_n = (v_1 - v_0) + 1$.

Si noti infine che se $f \in \widehat{LT}_1$, è sempre possibile scegliere $\overline{w}_0$ e $\overline{w}_n$ in modo che siano polinomiali e che soddisfino le disuguaglianze indicate. In tal caso si avrà che $g \in \widehat{LT}_1$. ∎

Gli ultimi due teoremi dimostrati, insieme al Teorema 1.9 e alle leggi di De Morgan, ci consentono di ricavare e dimostrare immediatamente gli analoghi casi in cui la nuova variabile $x_n$ è negata anziché diretta.

**Teorema 1.14.** *Sia $T$ un threshold gate avente ingressi $x_1, x_2, \ldots, x_{n-1}$, pesi $w_1, w_2, \ldots, w_{n-1}$ e soglia $w_0$, e sia $f(x_1, x_2, \ldots, x_{n-1})$ la funzione calcolata da $T$. Valgono le seguenti affermazioni:*

1. *Esiste un threshold gate $T'$ avente ingressi $x_1, x_2, \ldots, x_n$, pesi $w'_1, \ldots, w'_n$ e soglia $w'_0$ che calcola la funzione $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1}) \wedge \overline{x_n}$.*
2. *Esiste un threshold gate $T''$ avente ingressi $x_1, x_2, \ldots, x_n$, pesi $w''_1, \ldots, w''_n$ e soglia $w''_0$ che calcola la funzione $g(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_{n-1}) \vee \overline{x_n}$.*

*Dimostrazione.*

1. Per le leggi di De Morgan, valgono le seguenti uguaglianze:

$$g = f \wedge \overline{x_n} = \overline{\overline{f \wedge \overline{x_n}}} = \overline{\overline{f} \vee x_n}$$

   Per il Teorema 1.9, dal threshold gate $T$ si può passare ad un threshold gate $T_1$ che calcola la funzione $\overline{f}$. Da questo, per il Teorema 1.12, si può poi passare ad un threshold gate $T_2$ che calcola la funzione $\overline{f} \vee x_n$; infine, applicando ancora una volta il Teorema 1.9, si perviene al threshold gate $T'$ menzionato nell'enunciato.

2. Si procede esattamente come nel primo caso.

■

I Teoremi 1.12, 1.13 e 1.14 possono essere facilmente generalizzati al caso in cui $g(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m) = f(x_1, x_2, \ldots, x_n) \wedge \bigwedge_{i=1}^{m} Y_i$, dove $Y_i$ può essere sia $y_i$ che $\overline{y_i}$, e al caso in cui $g(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m) = f(x_1, x_2, \ldots, x_n) \vee \bigvee_{i=1}^{m} Y_i$: basterà applicare $m$ volte, rispettivamente, il Teorema 1.13 (1.14) o il Teorema 1.12 (1.14).

## 1.4 Teoremi di equivalenza per i threshold gate

I teoremi che seguono mostrano che, lavorando con i threshold gate, si possono all'occorrenza fare alcune assunzioni che semplificano i calcoli e/o consentono di evidenziare meglio certe proprietà delle funzioni calcolate; il tutto senza alterare la capacità computazionale dei threshold gate.

Cominciamo anzitutto col dare la seguente definizione.

**Definizione 1.6.** Due threshold gate $T$ e $T'$ si dicono *equivalenti* se calcolano la stessa funzione. ■

Ad esempio, se si moltiplicano per una costante $k$ positiva i pesi e la soglia di un threshold gate $T$ si ottiene un threshold gate $T'$ equivalente a $T$.

Il termine *equivalente* non è scelto a caso: indicando con $TG$ la classe formata da tutti i possibili threshold gate, e definita la relazione binaria $\sim \subseteq TG^2$:

$$T \sim T' \iff T \text{ calcola la stessa funzione di } T'$$

è immediato verificare che $\sim$ è una relazione di equivalenza, e che l'insieme quoziente $TG/\sim$ può essere messo in corrispondenza biunivoca con $LT_1$. In modo analogo, imponendo la condizione di polinomialità per i pesi dei threshold gate, si può costruire un insieme quoziente isomorfo a $\widehat{LT}_1$.

Resta infine da notare che, in tutti i teoremi d'equivalenza che seguono, se il threshold gate $T$ di partenza ha tutti i pesi polinomiali, il threshold gate $T'$ equivalente ricavato ha anch'esso tutti i pesi polinomiali.

Il primo teorema di equivalenza che vediamo afferma che, all'occorrenza, si può supporre che nessun vertice dell'$n$-cubo giaccia sull'iperpiano individuato da un threshold gate.

**Teorema 1.15.** *Sia $T$ un threshold gate avente pesi reali $w_1, w_2, \ldots, w_n$ e soglia reale $w_0$, e sia $f(x_1, x_2, \ldots, x_n)$ la funzione Booleana computata da $T$. Esiste un threshold gate $T'$ equivalente a $T$, avente pesi $w_1, w_2, \ldots, w_n$ e soglia $k$, tale che:*

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{se } \sum_{j=1}^{n} w_j x_j > k \\ 0 & \text{se } \sum_{j=1}^{n} w_j x_j < k \end{cases}$$

*Dimostrazione.* In corrispondenza di ciascuna delle $2^n$ configurazioni di ingresso resta determinato il valore di $\sum_{j=1}^{n} w_j x_j$ che, per quanto visto in precedenza, può essere maggiore, minore, o uguale a $w_0$.

Sia $v_1$ il massimo dei valori di tali sommatorie tale che $v_1 < w_0$, e sia $v_2$ il minimo dei suddetti valori tale che $w_0 \leq v_2$. Se $w_0 < v_2$, si ha subito $k = w_0$ e $T' = T$, in quanto $T$ soddisfa già le proprietà richieste; se $w_0 = v_2$, si può prendere $k$ fra uno qualsiasi dei valori strettamente compresi fra $v_1$ e $v_2$ (ad esempio il punto medio). ∎

Il prossimo teorema afferma che, all'occorrenza, si può supporre che tutti i pesi di un threshold gate siano non negativi (ovvero positivi, eliminando fisicamente le connessioni che hanno peso nullo). Si noti che il teorema vale solo nell'ipotesi che abbiamo fatto, di avere disponibili all'ingresso anche le negazioni $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$ dei segnali di ingresso.

**Teorema 1.16.** *Sia $T$ un threshold gate a $n$ ingressi e avente pesi $w_1, w_2, \ldots, w_n$, di cui alcuni negativi. Esiste un threshold gate $T'$ a $n$ ingressi, avente pesi $w'_1, w'_2, \ldots, w'_n$ tutti non negativi, equivalente a $T$.*

*Dimostrazione.* Sia $k$ la soglia, che supporremo non negativa (se fosse negativa, potremmo aggiungere un nuovo ingresso sempre uguale a 1, con peso $-k$, e porre la soglia a 0).

Senza perdere di generalità, supponiamo che sia $w_1 < 0$. Poiché $w_1 x_1 = w_1(1 - \overline{x_1})$, possiamo scrivere:

$$\sum_{i=1}^{n} w_i x_i - k = w_1 x_1 + \sum_{i=2}^{n} w_i x_i - k = w_1(1 - \overline{x_1}) + \sum_{i=2}^{n} w_i x_i - k =$$

$$= w_1 - w_1 \overline{x_1} + \sum_{i=2}^{n} w_i x_i - k$$

Posto $w'_1 = -w_1$ otteniamo:

$$\sum_{i=1}^{n} w_i x_i - k = w'_1 \overline{x_1} + \sum_{i=2}^{n} w_i x_i - (k + w'_1)$$

Iterando il procedimento $n$ volte, ponendo $w'_i = |w_i| \quad \forall i = 1, 2, \ldots, n$ e aggiustando opportunamente la soglia ogni volta che $w_i < 0$, si ottiene il threshold gate $T'$ che soddisfa l'enunciato. ∎

Pur trattando con valori di ingresso e di uscita binari, può capitare che i valori 0 e 1 non siano i più "comodi" con cui lavorare. Può ad esempio accadere che alcune proprietà delle funzioni threshold siano dimostrabili in maniera più semplice e/o diretta qualora gli ingressi e l'uscita dei threshold gate assumano i valori $-1$ e 1. Mentre è facile uniformare l'uscita a tali valori (è sufficiente adottare la funzione di trasferimento *sign* al posto di *step*), resta la questione della validità dei risultati così ottenuti.

Per risolvere tale questione, diamo anzitutto la seguente definizione:

**Definizione 1.7.** Sia $f : \{a, b\}^n \to \{a, b\}$ la funzione calcolata da un threshold gate $T$ ad $n$ ingressi e un'uscita binari a valori in $\{a, b\}$[2]. Analogamente, sia $g : \{c, d\}^n \to \{c, d\}$ la funzione calcolata da un threshold gate $T'$ ad $n$ ingressi e un'uscita binari a valori in $\{c, d\}$. Diciamo che $T'$ *simula* $T$ se, fissata la funzione biunivoca $\delta : \{a, b\} \to \{c, d\}$ in modo tale che:

$$\delta(a) = c$$
$$\delta(b) = d$$

per ogni configurazione delle variabili di ingresso $x_1, x_2, \ldots, x_n$ nell'insieme $\{a, b\}^n$ vale la seguente uguaglianza:

$$\delta\big(f(x_1, x_2, \ldots, x_n)\big) = g\big(\delta(x_1), \delta(x_2), \ldots, \delta(x_n)\big)$$

Diremo in tal caso che $T$ e $T'$ calcolano la stessa funzione (ovvero sono *equivalenti*) a meno dell'isomorfismo $\delta$. ∎

Il seguente teorema afferma la simulabilità di qualunque threshold gate $T$ a $n$ ingressi e un'uscita binari a valori in $\{a, b\}$ (con $a < b$), con un threshold gate $T'$ a valori in $\{c, d\}$ (con $c < d$). Come si può vedere nella dimostrazione, il motivo di tale equivalenza è essenzialmente il fatto che i valori $\{a, b\}$ sono legati ai valori $\{c, d\}$ attraverso relazioni lineari (trasformazioni affini). Tali relazioni, come è noto, descrivono traslazioni e/o dilatazioni dello spazio vettoriale in cui è immerso l'$n$-cubo delle configurazioni di ingresso; è chiaro che queste operazioni sull'$n$-cubo non alterano in alcun modo la separabilità dei suoi vertici da parte degli iperpiani.

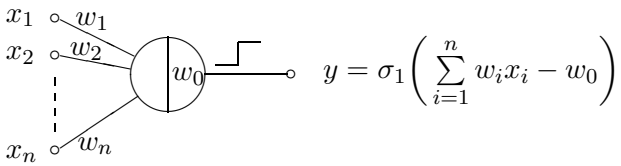
**Teorema 1.17.** *Sia $T$ un threshold gate a $n$ ingressi e un'uscita, che possono assumere ciascuno valori in $\{a, b\}$ (con $a < b$). Esiste un threshold gate $T'$ a*

[2]Sebbene utilizziamo la notazione insiemistica, interpretiamo i valori $a$ e $b$ nell'ordine indicato; diremo allora che una variabile binaria rappresenta la costante Booleana `falso` se assume il valore $a$, e rappresenta `vero` se assume il valore $b$.

*n ingressi e un'uscita a valori in $\{c, d\}$ (con $c < d$), equivalente a $T$ a meno dell'isomorfismo $\delta : \{a, b\} \to \{c, d\}$ tale che:*

$$\delta(a) = c \qquad \delta(b) = d$$

*Dimostrazione.* Sia $T$ il seguente threshold gate:




dove $x_i \in \{a, b\}$ per ogni $i = 1, \ldots, n$, $y \in \{a, b\}$ e

$$\sigma_1(t) = \begin{cases} b & \text{se } t \geq 0 \\ a & \text{se } t < 0 \end{cases}$$

Si definiscano:

$$S_i = \frac{d-c}{b-a}\,(x_i - a) + c \qquad \forall\, i = 1, \ldots, n$$
$$S_y = \frac{d-c}{b-a}\,(y - a) + c$$
$$\vartheta = \frac{d-c}{b-a}\,w_0 - \left(a\,\frac{d-c}{b-a} - c\right)\sum_{j=1}^{n} w_j$$

e sia

$$\sigma_2(t) = \begin{cases} d & \text{se } t \geq 0 \\ c & \text{se } t < 0 \end{cases}$$

È immediato verificare che valgono le seguenti uguaglianze:

$$S_i = \delta(x_i) \qquad \forall\, i = 1, 2, \ldots, n$$
$$S_y = \delta(y)$$
$$\sigma_2(t) = \delta\big(\sigma_1(t)\big) \qquad \forall\, t \in \Re$$

Allora il threshold gate $T'$, con gli stessi pesi di $T$ ma con la soglia uguale a $\vartheta$, ingressi $S_1, \ldots, S_n$ e funzione di trasferimento $\sigma_2$, è equivalente a $T$ a meno dell'isomorfismo $\delta$. Infatti, posto:

$$k = \frac{d-c}{b-a}$$

si ha:

$$\begin{aligned}
S_y &= \sigma_2\left(\sum_{j=1}^{n} w_j S_j - \vartheta\right) = \\
&= \sigma_2\left(\sum_{j=1}^{n} w_j\Big(k(x_j - a) + c\Big) - kw_0 + (ak - c)\sum_{j=1}^{n} w_j\right) = \\
&= \sigma_2\left(\sum_{j=1}^{n} w_j(kx_j + c - ka) - kw_0 + (ak - c)\sum_{j=1}^{n} w_j\right) = \\
&= \sigma_2\left(k\sum_{j=1}^{n} w_j x_j + (c - ka)\sum_{j=1}^{n} w_j - kw_0 + (ak - c)\sum_{j=1}^{n} w_j\right) = \\
&= \sigma_2\left(k\sum_{j=1}^{n} w_j x_j - kw_0\right) = \sigma_2\left[k\left(\sum_{j=1}^{n} w_j x_j - w_0\right)\right]
\end{aligned}$$

Essendo $k > 0$, l'ultima quantità ottenuta è uguale a:

$$\sigma_2\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

Infine, si ottiene:

$$k\,(y - a) + c = \sigma_2\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

$$y = \frac{1}{k}\,\sigma_2\left(\sum_{j=1}^{n} w_j x_j - w_0\right) - \frac{c}{k} + a = \sigma_1\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

da cui si deduce che $T'$ emette in uscita il valore $c$ se e solo se $T$ produce in uscita il valore $a$, altrimenti emette il valore $d$. ∎

Poiché, agli effetti pratici, gli unici valori binari utilizzati solitamente oltre a $\{0, 1\}$ sono $\{-1, 1\}$, affermiamo esplicitamente l'equivalenza in questo caso particolare, attraverso i due teoremi seguenti.

**Teorema 1.18.** *Sia $T$ un threshold gate a $n$ ingressi e un'uscita, che possono assumere ciascuno valori in $\{0, 1\}$. Esiste un threshold gate $T'$ a $n$ ingressi e un'uscita, equivalente a $T$, tale che ciascun ingresso e l'uscita possono assumere valori in $\{-1, 1\}$.*

*Dimostrazione.* Basta porre, nella dimostrazione del Teorema 1.17:

$$a = 0 \quad b = 1 \quad c = -1 \quad d = 1$$

∎

**Teorema 1.19.** *Sia $T'$ un threshold gate a $n$ ingressi e un'uscita, che possono assumere ciascuno valori in $\{-1, 1\}$. Esiste un threshold gate $T$ a $n$ ingressi e un'uscita, equivalente a $T'$, tale che ciascun ingresso e l'uscita possono assumere valori in $\{0, 1\}$.*

*Dimostrazione.* Basta porre, nella dimostrazione del Teorema 1.17:

$$a = -1 \quad b = 1 \quad c = 0 \quad d = 1$$

■

Si noti infine che, grazie al Teorema 1.9, è possibile estendere il risultato del Teorema 1.17 anche al caso in cui si abbia $a < b$ e $c > d$.

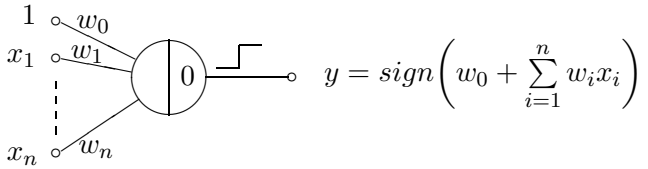
## 1.5 Il Majority gate

Come è stato detto, un *majority gate* a $n$ ingressi e un'uscita (a valori in $\{0, 1\}$) ha tutti i pesi uguali a 1 e la soglia pari a $\frac{n}{2}$. Nel caso in cui ingressi e uscita possano assumere valori in $\{-1, 1\}$, anche ai pesi viene consentito di assumere valori in $\{-1, 1\}$; tale concessione non altera la capacità computazionale del dispositivo se, come abbiamo supposto all'inizio, oltre ai segnali di ingresso $x_1, x_2, \ldots, x_n$ si hanno a disposizione le loro negazioni $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$ [3].

A causa della limitazione sui valori assunti dai pesi, si potrebbe pensare che i majority gate siano dispositivi di calcolo poco potenti. Se però aggiriamo questo ostacolo consentendo collegamenti multipli tra un majority gate e gli ingressi, allora i majority gate diventano sorprendentemente potenti. Vale infatti il seguente teorema.

**Teorema 1.20.** *Dato un threshold gate $T$ a $n$ ingressi e con pesi polinomiali, sia $f : \{-1, 1\}^n \to \{-1, 1\}$ la funzione calcolata da $T$. Esiste allora un majority gate $M$ con un numero polinomiale di connessioni in ingresso tali che $M$ calcola $f$.*

*Dimostrazione.* Sia $T$ il seguente threshold gate:




[3] Si noti infatti che, nel dominio $\{-1, 1\}$, la negazione logica è data dal cambio di segno della variabile interessata; quindi, qualunque sia il peso $w_i$, vale:

$$-w_i x_i = w_i \overline{x_i}$$

Risulta pertanto equivalente utilizzare pesi a valori $\{-1, 1\}$ con gli ingressi $x_1, x_2, \ldots, x_n$ o pesi tutti uguali a 1 con gli ingressi $x_1, x_2, \ldots, x_n$ e $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$.

dove $x_1, x_2, \ldots, x_n$ sono variabili che possono assumere esclusivamente valori in $\{-1, 1\}$.

Si consideri un majority gate avente $2\sum_{j=0}^{n} |w_j|$ ingressi disposti in modo tale che per ogni $i = 1, \ldots, n$ ci sono $|w_i|$ ingressi collegati a $x_i$ e aventi ciascuno peso pari a $sign(w_i)$, $|w_0|$ ingressi che assumono sempre valore 1 e aventi ciascuno peso pari a $sign(w_0)$, e $\tau = \sum_{j=0}^{n} |w_j|$ ingressi che assumono sempre valore 1 e aventi ciascuno peso pari a 1.

È facile verificare che il majority gate così ottenuto calcola la stessa funzione $f$ del threshold gate di partenza: il majority gate fornisce in uscita 1 se e solo se:

$$\sum_{i=1}^{n} \sum_{j=1}^{|w_i|} sign(w_i) x_i + \sum_{j=1}^{|w_0|} sign(w_0) + \sum_{j=1}^{\tau} 1 \;\geq\; \sum_{j=0}^{n} |w_j|$$

se e soltanto se

$$\sum_{i=1}^{n} \frac{w_i}{|w_i|} |w_i| x_i + \frac{w_0}{|w_0|} |w_0| + \tau \geq \sum_{j=0}^{n} |w_j|$$

se e soltanto se

$$\sum_{i=1}^{n} w_i x_i + w_0 \geq 0$$

■

Con le assunzioni fatte, quindi, un majority gate diventa a tutti gli effetti equivalente a un threshold gate. Si noti che la simulazione descritta nella dimostrazione del teorema vale in ogni caso, ma si giunge a un numero polinomiale di connessioni se e solo se il threshold gate di partenza ha tutti i pesi polinomiali. Il risultato espresso dal Teorema 1.20 ci consente brevemente di dire che ogni funzione Booleana appartenente a $\widehat{LT}_1$ può essere calcolata da un majority gate avente al più un numero polinomiale di connessioni; ciò tornerà utile, come vedremo, quando si cercherà di estendere i risultati visti finora alle funzioni calcolate dai circuiti.

# Capitolo 2

# Circuiti di neuroni

A partire da questo capitolo, l'attenzione si sposta sui *circuiti a soglia*, ovvero sui circuiti formati da threshold gate. Dopo aver definito le quantità che caratterizzano la capacità computazionale dei circuiti (dimensione, peso e profondità), vengono definite le classi di funzioni calcolabili dai circuiti a soglia di dimensione polinomiale e profondità costante. Purtroppo, si scopre subito che caratterizzare le funzioni appartenenti a tali classi non è facile come per i threshold gate singoli, e quindi il massimo che si può fare è studiare alcune proprietà delle classi (quali la chiusura rispetto alle operazioni logiche di disgiunzione e congiunzione) e illustrare i risultati ottenuti nella letteratura.

Si pongono così in maniera naturale alcuni problemi pratici, quali la separabilità delle classi di funzioni in base alla profondità dei circuiti che le calcolano, e la determinazione della quantità minima di risorse necessarie a calcolare una funzione data.

Nel corso della discussione, si determinano alcune importanti sottoclassi di funzioni calcolabili da circuiti a soglia di profondità 2 e dimensione e peso polinomiali; una semplice osservazione legata alla costruzione di tali circuiti porta ad individuare una tecnica che consente a volte di abbassare la profondità di un circuito che calcola una funzione data.

## 2.1 Descrizione del modello

Collegando fra loro i neuroni formali descritti nel capitolo precedente è possibile costruire alcuni modelli di *reti neuronali*.

Per collegamento tra due neuroni $N_1$ e $N_2$ si intende una connessione fisica dell'uscita di $N_1$ con uno (o più) degli ingressi di $N_2$, e/o viceversa; nel caso più generale, non vi è alcuna limitazione al modo in cui i neuroni possono essere collegati fra loro, ed è anche possibile che l'uscita di un neurone torni retroattivamente in ingresso al neurone stesso.

Possiamo quindi dare una prima definizione molto generale.

**Definizione 2.1.** Una *rete neuronale* è un grafo orientato i cui nodi sono neuroni formali (si veda la Definizione 1.1) e i cui lati sono costituiti dalle linee di ingresso e uscita dei neuroni stessi. ∎

La rete comunica con l'ambiente esterno attraverso delle *linee di ingresso* $x_1, x_2, \ldots, x_n$, collegate agli ingressi di uno o più neuroni, e delle *linee d'uscita* $y_1, y_2, \ldots, y_m$, prese dalle uscite di $m$ neuroni prescelti. In ogni istante, su ciascuna delle linee di ingresso arriva alla rete un simbolo tratto da un *alfabeto di ingresso*, e su ciascuna delle linee d'uscita la rete deposita un simbolo tratto da un *alfabeto di uscita*.

Poiché in questi appunti siamo interessati al calcolo di (famiglie di) funzioni Booleane tramite reti neuronali, restringiamo la nostra attenzione al caso in cui gli alfabeti di ingresso e uscita sono alfabeti binari. Anche con questa restrizione, la definizione di rete appena data resta troppo generica: a causa dei cicli di retroazione presenti nella rete, il comportamento è influenzato dal metodo scelto per l'aggiornamento delle uscite dei singoli neuroni; inoltre, non è affatto detto che tenendo fissi i valori delle linee di ingresso si ottengano dei valori costanti sulle linee d'uscita. Poiché in questo scenario ci sarebbe qualche difficoltà perfino a definire la funzione calcolata da una rete siffatta, restringiamo il nostro campo d'azione ponendo dei vincoli alla topologia della rete.

La prima restrizione che imponiamo è che la rete sia *aciclica* e connessa.

**Definizione 2.2.** Un *circuito neuronale* è una rete neuronale aciclica e connessa. ∎

D'ora in poi, a meno che non venga indicato esplicitamente il contrario, supporremo sempre che l'uscita di un circuito sia una sola, e la indicheremo semplicemente con $y$.

Si noti che, nel caso di un circuito, non viene richiesto che il grafo sia un albero, ma basta che sia aciclico e connesso; la differenza consiste nel fatto che il termine *aciclico* si riferisce all'assenza di cicli *orientati*. In tali condizioni si può pensare al circuito come costituito da diversi *strati* di neuroni, e vale la proprietà che per ogni coppia di neuroni $A$ e $B$ appartenenti allo stesso strato, l'uscita di $A$ non è un ingresso di $B$. Dato che, in virtù del Teorema 1.6, è sempre possibile inserire nel circuito dei neuroni che riportano semplicemente in uscita uno dei segnali in ingresso, restringeremo la nostra attenzione ai circuiti in cui ogni cammino da un qualsiasi ingresso all'uscita attraversa sempre lo stesso numero di neuroni.

La Figura 2.1 mostra un esempio di circuito a 3 ingressi e un'uscita.

Si può a questo punto definire la funzione calcolata da un circuito. Dati i valori delle linee di ingresso il circuito viene, come si suol dire, *valutato in ordine topologico*, cioè strato dopo strato. Il primo strato ad essere valutato è quello direttamente collegato alle linee di ingresso: viene calcolato il valore di uscita di ciascun neurone dello strato secondo l'equazione (1.1). I valori di uscita così ottenuti costituiscono gli ingressi dei neuroni del secondo strato; anche per questi

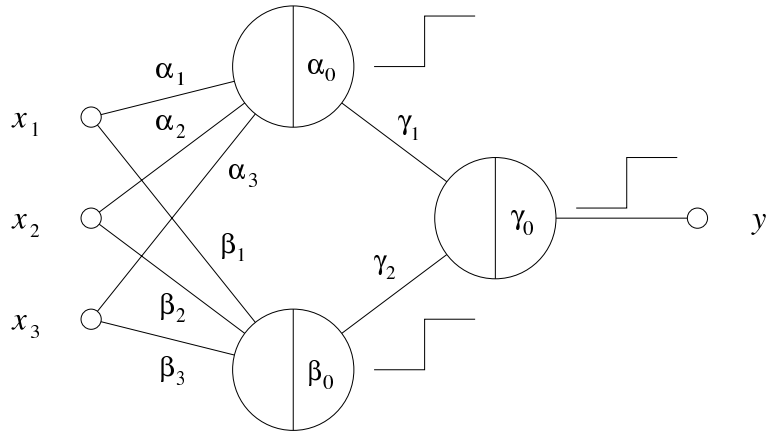



Figura 2.1: *Esempio di circuito neuronale a* 3 *ingressi e* 1 *uscita. Il circuito ha dimensione* 3*, profondità* 2 *e fan-in* 3 *(si vedano le Definizioni 2.5 e 2.8).*

si calcola il valore di uscita secondo l'equazione (1.1), e si procede in questo modo fino ad arrivare all'ultimo strato. Avendo supposto che il circuito neuronale abbia una sola uscita $y$, l'ultimo strato sarà composto da un unico neurone; un'ultima applicazione dell'equazione (1.1) consente di determinare il valore di $y$. È evidente che ad ogni assegnamento delle variabili di ingresso $x_1, x_2, \ldots, x_n$ viene associato un unico valore di $y$, secondo il procedimento di valutazione appena descritto; resta pertanto definita una funzione Booleana:

$$y = f(x_1, x_2, \ldots, x_n)$$

che, per definizione, chiameremo la *funzione calcolata dal circuito*. Tutto ciò è completamente analogo a quanto avviene considerando i *circuiti logici*, ovvero i circuiti costituiti dalle porte logiche `AND/OR/NOT`.

Come abbiamo già detto a proposito dei neuroni singoli, siamo interessati a calcolare famiglie di funzioni Booleane tramite circuiti neuronali. Poiché ogni circuito ha un numero fissato di ingressi, per fare ciò avremo bisogno di considerare intere *famiglie di circuiti*. Ricordiamo che una famiglia $\boldsymbol{\mathcal{F}}$ di funzioni Booleane è una successione $\{f_n\}_{n \in \boldsymbol{N}}$ di funzioni Booleane che hanno in maniera evidente una "forma" comune; parimenti, una famiglia $\boldsymbol{\mathcal{C}}$ di circuiti neuronali è una successione $\{C_n\}_{n \in \boldsymbol{N}}$ di circuiti a $n$ ingressi e un'uscita. Possiamo allora dare la seguente definizione.

**Definizione 2.3.** Diciamo che una famiglia $\boldsymbol{\mathcal{C}}$ di circuiti neuronali calcola una famiglia $\boldsymbol{\mathcal{F}}$ di funzioni Booleane se, per ogni $n \in \boldsymbol{N}$, il circuito $C_n$ calcola la funzione $f_n$. ∎

La seconda restrizione che imponiamo riguarda il tipo di neuroni impiegati: mentre è sicuramente possibile avere circuiti costituiti da diversi tipi di neurone,

supporremo d'ora in poi che i circuiti considerati siano costituiti interamente da un unico tipo di neurone. Si tratta allora di scegliere *quale* modello di neurone formale adottare, potendo scegliere tra perceptroni, threshold gate e majority gate. Il modello di circuito più potente e comunemente studiato in letteratura è la *rete a retropropagazione*: è un circuito di perceptroni in cui tutti i parametri, gli ingressi e le uscite di ogni neurone sono numeri reali arbitrari, e la funzione di trasferimento è una sigmoide. Si noti che i segnali utilizzati per comunicare da uno strato all'altro possono assumere qualsiasi valore reale; ciò, oltre al fatto che in un numero reale dotato di infinite cifre decimali è possibile codificare qualsiasi cosa, in un certo senso "tradisce" lo spirito dei circuiti Booleani, nei quali si intende che ogni elemento costituente il circuito comunica con gli altri solo attraverso segnali binari. Ci concentreremo allora sullo studio dei più semplici circuiti costituiti interamente da threshold gate o da majority gate.

**Definizione 2.4.** Chiamiamo *circuito a soglia* un circuito costituito interamente da threshold gate, mentre con *circuito majority* intendiamo un circuito costituito interamente da majority gate. ∎

Per poter utilizzare i risultati illustrati nel Capitolo 1, supporremo anche qui che oltre ai segnali di ingresso $x_1, x_2, \ldots, x_n$ si abbiano a disposizione le loro negazioni logiche $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$, oltre a un segnale costantemente uguale 1 e a un segnale costantemente uguale a 0.

## 2.2 Dimensione, profondità e peso di un circuito

Nel confrontare tra loro due circuiti che calcolano una data funzione, i parametri rilevanti saranno la *dimensione*, la *profondità* e il *peso* dei circuiti, definiti nel modo seguente.

**Definizione 2.5.** Dato un circuito neuronale $C$ (a soglia o majority), definiamo la *dimensione* e la *profondità* di $C$ rispettivamente come il numero dei neuroni e il numero degli strati che formano il circuito. Definiamo inoltre il *peso* di $C$ come il massimo dei valori assoluti dei pesi (comprese le soglie) dei neuroni di $C$. Indichiamo la dimensione, la profondità e il peso di $C$ rispettivamente con: $dim(C)$, $prof(C)$ e $peso(C)$. ∎

Chiaramente, tra due circuiti che calcolano la stessa funzione, sarà da preferire quello che utilizza meno risorse: per motivi che risulteranno chiari in seguito, la situazione ideale si ha quando un circuito $C$ a $n$ ingressi riesce a calcolare una data funzione $f(x_1, x_2, \ldots, x_n)$ avendo una *profondità costante* e *dimensione e peso polinomiali*, ovvero quando esistono un numero intero $d \geq 1$ e un polinomio $p(n)$ tali che $prof(C) = d$, $dim(C) \leq p(n)$ e $peso(C) \leq p(n)$[1].

[1] Si noti che è stato utilizzato lo stesso polinomio $p(n)$ per limitare sia $dim(C)$ che $peso(C)$; questo per mettere in evidenza il fatto che, se $dim(C) \leq p_1(n)$ e $peso(C) \leq p_2(n)$, con $p_1(n)$ e $p_2(n)$ polinomi distinti, $p(n) = \max\{p_1(n), p_2(n)\}$ è ancora un polinomio per il quale vale $dim(C) \leq p(n)$ e $peso(C) \leq p(n)$.

Essendo interessati alla distinzione tra dimensione e peso polinomiali rispetto a dimensione e peso superpolinomiali (ovvero, nella quasi totalità dei casi, esponenziali), si possono fornire le seguenti definizioni, che risultano essere equivalenti a quelle già date.

**Definizione 2.6.** Dato un circuito neuronale $C$ (a soglia o majority) definiamo la *dimensione* di $C$ come il numero totale di collegamenti fra i neuroni che compongono $C$, e il *peso* di $C$ come la somma totale dei valori assoluti di tutti i pesi del circuito. ∎

Mostriamo anzitutto che è equivalente, per i nostri scopi, definire la dimensione di un circuito come il numero di neuroni o come il numero delle connessioni presenti nel circuito.

Infatti, abbia il circuito un numero polinomiale di neuroni. Poiché il numero di archi in un grafo è al più pari al quadrato del numero dei nodi, anche il numero di connessioni nel circuito sarà polinomiale. Viceversa, un numero polinomiale di collegamenti potrà connettere tra loro al più un numero polinomiale di neuroni e quindi, almeno nel caso polinomiale, le due definizioni sono equivalenti.

Supponiamo ora che il circuito abbia un numero superpolinomiale di neuroni. Poiché un numero polinomiale di collegamenti può connettere al più un numero polinomiale di neuroni, il circuito avrà anche un numero superpolinomiale di connessioni. Viceversa, un numero superpolinomiale di collegamenti è sovrabbondante per connettere tra loro un numero polinomiale di neuroni (dato che un circuito completamente connesso con un numero polinomiale di neuroni ha al più un numero polinomiale di collegamenti) e quindi, anche nel caso superpolinomiale, le due definizioni sono equivalenti.

Per quanto riguarda la nozione di *peso* di un circuito $C$ si noti anzitutto che, essendo i pesi di ogni neurone passibili di moltiplicazione per un fattore positivo qualsiasi senza che la funzione calcolata ne risulti alterata (vedere l'equazione (1.1)), il calcolo del peso del circuito ha senso solo quando tutti i pesi dei neuroni sono numeri interi. D'altra parte, per il Teorema 1.1 è sempre possibile supporre che i pesi di ogni neurone del circuito siano interi. Per togliere ogni dubbio, e per rendere univoco il valore di $peso(C)$, supporremo che, per ogni peso $w$ presente nel circuito $C$:

- $w \in \boldsymbol{Z}$
- $w \cdot 10^{-n} \notin \boldsymbol{Z} \quad \forall n \geq 1.$

Fatte queste dovute precisazioni osserviamo che, dovendo discriminare unicamente i casi in cui $peso(C)$ è o non è polinomiale, è equivalente definire $peso(C)$ come $\max_{w \in C} |w|$ o come $\sum_{w \in C} |w|$, nel senso che la prima quantità è polinomiale se e solo se lo è la seconda. Infatti, sia $\max_{w \in C} |w| \leq p(n)$ per un opportuno polinomio $p(n)$; allora $|w| \leq p(n) \quad \forall w \in C$, da cui si deduce

che $\sum_{w \in C} |w| \leq \sum_{w \in C} p(n) = p(n) \sum_{w \in C} 1$. Essendo il numero dei collegamenti di $C$ polinomiale, esisterà un polinomio $q(n)$ tale che $\sum_{w \in C} 1 \leq q(n)$, da cui $\sum_{w \in C} |w| \leq p(n) \cdot q(n) = r(n)$, dove $r(n)$ è anch'esso un polinomio. Viceversa, sia $\sum_{w \in C} |w| \leq p(n)$ per un opportuno polinomio $p(n)$; se fosse $\max_{w \in C} |w| > q(n)$ *per ogni* polinomio $q(n)$ avremmo la situazione assurda in cui il polinomio $p(n)$ è tale che $p(n) \geq \sum_{w \in C} |w| \geq \max_{w \in C} |w| > q(n)$ *per ogni* polinomio $q(n)$ (incluso p(n) stesso).

Definiti i parametri che ci consentono di misurare le risorse utilizzate da un circuito per calcolare una data funzione, possiamo ribaltare il punto di vista e, fissate le risorse a disposizione, vedere quali sono le funzioni calcolabili che otteniamo. Possiamo così definire, per ogni intero $k \geq 0$, le seguenti classi di funzioni Booleane.

**Definizione 2.7.** Indichiamo con $TC^k$, per $k \geq 0$ intero, la classe delle funzioni computabili da circuiti a soglia di dimensione polinomiale e profondità $O(\log^k n)$. Analogamente, indichiamo con $\widehat{TC^k}$ la classe delle funzioni computabili da circuiti a soglia di dimensione e peso polinomiali e profondità $O(\log^k n)$. ∎

La definizione è analoga a quella della classe $AC^k$ riguardante i circuiti logici (si veda ad esempio il primo volume del libro [1]), con la differenza che, nel caso dei circuiti a soglia, occorre distinguere tra pesi polinomiali e pesi illimitati.

Più che con la classe $AC^k$, chi studia i circuiti logici avrà a che fare con la classe $NC^k$; per comprendere la differenza tra queste due classi, diamo anzitutto la seguente definizione.

**Definizione 2.8.** Si dice *fan-in* il numero di ingressi di un neurone (o di una porta logica); analogamente, si dice *fan-out* il numero delle uscite. Si possono estendere facilmente le due definizioni al caso dei circuiti: il *fan-in di un circuito* $C$ è, per definizione, il massimo tra i fan-in dei neuroni che compongono $C$, e analogamente per il fan-out. ∎

Occorre fare attenzione al fatto che, avendo supposto che ogni neurone produce *un* valore di uscita e che i circuiti considerati hanno un'unica uscita $y$, non sottointendiamo in alcun modo che il fan-out sia sempre uguale a 1: l'uscita di un dato neurone, infatti, può essere utilizzata più di una volta per propagarne il valore ai neuroni dello strato adiacente, aumentando così il valore di fan-out.

Supponiamo allora di avere un circuito $C_n$ di profondità 2 e dimensione polinomiale: il neurone di uscita, se vorrà tener conto delle uscite di ogni neurone del primo strato, dovrà avere un numero di ingressi che cresce in maniera polinomiale al crescere di $n$; quindi, in generale, il fan-in di un circuito dipenderà da $n$. Se richiediamo invece che il fan-in di ogni circuito considerato *non dipenda da* $n$, ovvero sia limitato da una costante, otterremo un'altra classe di funzioni

calcolate, che in genere non coinciderà con la classe delle funzioni calcolate da circuiti aventi fan-in illimitato. Questo è proprio quanto accade con la classe $NC^k$: essa è la classe delle funzioni Booleane calcolate da circuiti logici di profondità $O(\log^k n)$, dimensione polinomiale e fan-in limitato da una costante.

In questi appunti ci occuperemo esclusivamente di circuiti neuronali aventi fan-in e fan-out illimitati. Si noti tuttavia che, lavorando con circuiti di dimensione e peso polinomiali, il fan-in e il fan-out risulteranno anch'essi polinomiali dato che in tal caso, come è già stato osservato a proposito dell'equivalenza delle Definizioni 2.5 e 2.6, il numero di connessioni utilizzate è al più polinomiale. Anche nel caso in cui si voglia sostituire un threshold gate a pesi polinomiali con un majority gate, come spiegato nel Teorema 1.20, il fan-in e il fan-out restano polinomiali.

A proposito della dimensione, il lettore attento si sarà già chiesto come mai, nella Definizione 2.7, non siano state considerate anche le funzioni calcolate da circuiti di dimensione esponenziale; il motivo è che tali circuiti sono *troppo* potenti (e quindi poco interessanti, oltre che dispendiosi in termini di risorse): come si avrà modo di vedere più avanti, bastano i circuiti di profondità 2 e dimensione esponenziale per calcolare *qualsiasi* funzione Booleana.

Ponendo $k = 0$ nella Definizione 2.7 si ottengono le classi $TC^0$ e $\widehat{TC^0}$, ovvero le classi di funzioni calcolate dai circuiti a soglia di dimensione polinomiale e profondità costante. Come avremo modo di vedere, queste due classi contengono funzioni molto interessanti, e costituiscono il nostro oggetto di studio. Le caratteristiche delle funzioni che appartengono a queste due classi non sono ancora completamente note: a conoscenza di chi scrive, per esempio, a tutt'oggi (1998) non si conosce alcuna funzione Booleana che *necessiti* di un circuito a soglia di profondità $O(\log n)$, o comunque dipendente da $n$, per poter essere calcolata.

È anche naturale suddividere le classi $TC^0$ e $\widehat{TC^0}$ in sottoclassi, in base alla profondità effettiva dei circuiti.

**Definizione 2.9.** Indichiamo indifferentemente con $TC^0_d$ o $LT_d$, per $d \geq 1$ intero, la classe delle funzioni computabili da circuiti a soglia di dimensione polinomiale e profondità $d$. Analogamente, indichiamo con $\widehat{TC^0_d}$ o $\widehat{LT}_d$ la classe delle funzioni computabili da circuiti a soglia di dimensione polinomiale, peso polinomiale e profondità $d$. ∎

D'ora in poi utilizzeremo preferibilmente la notazione $LT_d$ e $\widehat{LT}_d$, che è stata introdotta da Siu e Bruck [15] e che si è andata via via affermando.

Si noti come, per $d = 1$, si ottenga una precisa corrispondenza con le classi di funzioni $LT_1$ e $\widehat{LT}_1$ introdotte nella Definizione 1.3: tale corrispondenza è voluta, e corrisponde all'identificazione dei neuroni singoli con i circuiti di profondità 1. Inoltre, si noti come dalla Definizione 2.9 segue che:

$$TC^0 = \bigcup_{d=1}^{+\infty} LT_d \qquad\qquad \widehat{TC^0} = \bigcup_{d=1}^{+\infty} \widehat{LT}_d$$

Si può infine osservare che, come avviene per la classe $AC^0$, si possono definire diverse varianti *uniformi* delle classi $TC^0$ e $\widehat{TC^0}$, corrispondenti a diverse limitazioni sulla costruzione delle famiglie di circuiti. Una limitazione che si trova comunemente in letteratura, ad esempio, è che il circuito $C_n$ a $n$ ingressi di una famiglia $\boldsymbol{\mathcal{C}}$ che calcola una data famiglia $\boldsymbol{\mathcal{F}}$ di funzioni Booleane appartenenti a $TC^0$ (o a $\widehat{TC^0}$) sia "costruibile" da una Macchina di Turing Deterministica utilizzando al più $\log_2 p(n)$ celle del nastro, dove $p(n)$ indica la dimensione di $C_n$[2]. La "costruzione" del circuito $C_n$ ad opera della Macchina di Turing Deterministica avviene calcolando una funzione:

$$M : 1^n \to \overline{C_n}$$

dove $\overline{C_n}$ è una opportuna codifica del circuito $C_n$.

Per ulteriori informazioni riguardo l'uniformità delle famiglie di circuiti si veda il quarto capitolo del secondo volume del libro di Balcázar, Díaz e Gabarró [1], nel quale vengono esposti risultati — anche abbastanza recenti — su alcune famiglie uniformi di circuiti logici derivate dalle classi $AC^k$ e $NC^k$. In questa sede non verranno prese in considerazione simili o altre limitazioni riguardanti la costruzione delle famiglie di circuiti a soglia: adotteremo cioè l'approccio che viene comunemente chiamato *non uniforme*. Secondo questo punto di vista, verranno accettate anche dimostrazioni non costruttive di teoremi che mostrano l'appartenenza di una data funzione Booleana ad una determinata classe, o dimostrazioni "costruttive" per le quali la costruzione indicata del circuito $C_n$ impiega tempo e/o spazio anche esponenziali.

## 2.3 Alcune proprietà delle classi $LT_d$ e $\widehat{LT}_d$

In questo paragrafo cominciamo a vedere alcune proprietà delle classi di funzioni $LT_d$ e $\widehat{LT}_d$; presentiamo inoltre alcune tecniche elementari che consentono di provare che una data funzione appartiene alla classe $\widehat{LT}_2$, e introduciamo il problema della *separazione* delle classi.

Notiamo anzitutto che dalla Definizione 2.9 discende immediatamente che vale l'inclusione:

$$\widehat{LT}_d \subseteq LT_d$$

poiché i circuiti utilizzati per calcolare le funzioni in $\widehat{LT}_d$ sono ottenuti da quelli utilizzati per calcolare le funzioni in $LT_d$ imponendo un vincolo polinomiale ai pesi.

Essendo i circuiti costituiti da neuroni, alcune proprietà di questi ultimi si riflettono immediatamente sulle classi $LT_d$ e $\widehat{LT}_d$. Ad esempio, come è già stato osservato a proposito della definizione di peso di un circuito, applicando il

[2]Si noti che se $p(n)$ è un polinomio, allora $p(n) = O(n^c)$ per qualche costante intera $c > 0$, e $\log_2 p(n) = O(c \cdot \log_2 n) = O(\log n)$. Se invece $p(n) = O(2^n)$, avremo $\log_2 p(n) = O(n \cdot \log_2 2) = O(n)$.

Teorema 1.1 ad ogni threshold gate dei circuiti considerati, è immediato rendersi conto che anche per i circuiti a soglia è sufficiente considerare i pesi interi.

Il Teorema 1.2, invece, non è immediatamente generalizzabile ai circuiti; questo perché si conosce ancora poco la struttura delle classi $LT_d$ e $\widehat{LT}_d$. Supponiamo, infatti, di prendere un circuito $C$ di profondità $d$ e dimensione e peso polinomiali: la funzione calcolata da $C$ appartiene allora a $\widehat{LT}_d$; se ora sostituiamo il neurone d'uscita con uno dei neuroni a pesi esponenziali che calcolano le funzioni menzionate nel Teorema 1.2[3], otteniamo un nuovo circuito che calcola una funzione $f \in LT_d$. Alla semplice e legittima domanda: "$f \in \widehat{LT}_d$?" non c'è attualmente risposta, poiché non ci è ancora dato di sapere se, cambiando opportunamente i pesi negli strati inferiori, non sia possibile ottenere un circuito di peso polinomiale che calcola $f$. Inoltre, come ci si renderà conto in seguito, è un problema generalmente difficile stabilire qual è il circuito di profondità minima che calcola una data funzione Booleana.

Questi sono alcuni dei motivi per cui è interessante studiare le proprietà delle classi $LT_d$ e $\widehat{LT}_d$. Cominciamo pertanto ad enunciare alcuni teoremi che illustrano le proprietà trovate finora.

Il primo teorema è stato dimostrato da Goldmann, Håstad e Razborov [4].

**Teorema 2.1.** *Sia $\widetilde{LT}_d$ la classe delle funzioni calcolate dai circuiti a soglia di profondità $d$ e dimensione polinomiale, in cui i pesi del neurone d'uscita sono polinomiali (mentre non vi è alcuna restrizione sui pesi degli altri neuroni). Allora, per ogni intero $d \geq 1$ fissato, $\widetilde{LT}_d = \widehat{LT}_d$.* ∎

La dimostrazione del teorema è troppo lunga e complicata per riportarla in queste pagine. Si noti solamente che essa si basa sul fatto che, essendo i pesi del neurone d'uscita polinomiali, questo può essere sostituito da un majority gate, come illustrato dal Teorema 1.20, *mantenendo polinomiale la dimensione del circuito*; come è già stato detto a proposito del Teorema 1.20, ciò non è possibile se i pesi del neurone di partenza sono esponenziali.

Cogliamo inoltre l'occasione per osservare che, sempre grazie al Teorema 1.20, $\widehat{LT}_d$ può essere equivalentemente definita come la classe di funzioni Booleane calcolate da circuiti majority di dimensione polinomiale e profondità $d$. Un altro risultato immediato lo si ottiene applicando i Teoremi 1.17 e 1.9 a ciascun neurone dei circuiti considerati: tutti i circuiti ad ingressi e uscite binarie (dove per binario si intende che può assumere due qualsiasi valori distinti) sono equivalenti, nel senso che calcolano la stessa funzione; l'unica restrizione che imponiamo è quella della compatibilità degli ingressi con le uscite: ovvero, se un dato neurone $\mathcal{N}$ produce in uscita valori in $\{a, b\}$, ogni neurone che riceve in ingresso l'uscita di $\mathcal{N}$ dovrà essere in grado di funzionare correttamente con i valori $\{a, b\}$ in ingresso. È facile rendersi conto che tale restrizione è, dal punto di vista della funzione

[3] Non è un caso che si decida di sostituire proprio il neurone d'uscita, anziché un neurone qualunque; il motivo è da ricercarsi nel Teorema 2.1, che risolve la questione proposta per tutti i neuroni tranne che per quello d'uscita.

calcolata dal circuito, equivalente a quella, apparentemente più restrittiva, di imporre che tutti i neuroni del circuito siano uguali; il punto fondamentale dell'osservazione precedente è che tutti i neuroni comunicano tra loro utilizzando valori binari, *qualunque essi siano*: le funzioni calcolate dai circuiti, qualora siano definite correttamente, sono *invarianti* rispetto ai valori scelti.

Vediamo ora come si comportano le classi $LT_d$ e $\widehat{LT}_d$ rispetto alle operazioni logiche. Il primo teorema che dimostriamo è la naturale estensione del Teorema 1.9.

**Teorema 2.2.** *$LT_d$ e $\widehat{LT}_d$ sono chiuse rispetto alla negazione logica.*

*Dimostrazione.* Sia $f(x_1, x_2, \ldots, x_n) \in LT_d$. Esiste allora un circuito a soglia $C_n$ a $n$ ingressi che calcola $f$; applicando il Teorema 1.9 al neurone d'uscita di $C_n$, è possibile costruire un nuovo circuito $C'_n$ che calcola la funzione $\overline{f}$, che di conseguenza appartiene a $LT_d$. Per le considerazioni già fatte a proposito del Teorema 1.9, se la funzione $f$ di partenza appartiene a $\widehat{LT}_d$, anche $\overline{f}$ apparterrà a $\widehat{LT}_d$. ∎

Con l'ausilio del teorema appena dimostrato, e del Teorema 2.1, siamo in grado di dimostrare i prossimi due teoremi.

**Teorema 2.3.** *Siano $f$ e $g$ due funzioni appartenenti a $LT_d$; allora $f \vee g \in \widehat{LT}_{d+1}$.*

*Dimostrazione.* Per ipotesi, esistono due circuiti $C_f$ e $C_g$, eventualmente di peso esponenziale e non necessariamente con lo stesso numero di ingressi, che calcolano rispettivamente $f$ e $g$ in profondità $d$ e dimensione polinomiale.

Per il Teorema 1.4, si può calcolare $f \vee g$ mandando le uscite di $C_f$ e $C_g$ in ingresso a un threshold gate con pesi costanti. Il circuito risultante dimostra che $f \vee g \in \widetilde{LT}_{d+1}$; per il Teorema 2.1, ciò significa che $f \vee g \in \widehat{LT}_{d+1}$. ∎

Si noti che il teorema precedente può essere generalizzato al caso in cui si voglia effettuare la disgiunzione di un numero al più polinomiale di funzioni: anche in tal caso, infatti, il threshold gate d'uscita avrà peso al più polinomiale, così che la disgiunzione apparterrà alla classe $\widetilde{LT}_{d+1}$.

Grazie alle leggi di De Morgan e ai Teoremi 2.2 e 2.3 possiamo dimostrare l'analogo caso della congiunzione logica.

**Teorema 2.4.** *Siano $f$ e $g$ due funzioni appartenenti a $LT_d$; allora $f \wedge g \in \widehat{LT}_{d+1}$.*

*Dimostrazione.* Per le leggi di De Morgan, vale:

$$f \wedge g = \overline{\overline{f} \vee \overline{g}}$$

Per ipotesi, esistono due circuiti $C_f$ e $C_g$, eventualmente di peso esponenziale e non necessariamente con lo stesso numero di ingressi, che calcolano

rispettivamente $f$ e $g$ in profondità $d$ e dimensione polinomiale. Per il Teorema 2.2 esistono allora due circuiti $C_{\overline{f}}$ e $C_{\overline{g}}$ che calcolano rispettivamente $\overline{f}$ e $\overline{g}$, sempre in profondità $d$; per il Teorema 2.3 esiste un circuito di peso polinomiale e profondità $d+1$ che calcola $\overline{f} \vee \overline{g}$. Infine, un'ultima applicazione del Teorema 2.2 basta a dimostrare che $f \wedge g \in \widehat{LT}_{d+1}$.

Si noti come, alternativamente, si possa calcolare $f \wedge g$ mandando le uscite di $C_f$ e $C_g$ a un opportuno threshold gate, come è stato fatto nel Teorema 2.3. ∎

Dopo aver visto gli ultimi due teoremi dimostrati, sorge spontanea una domanda: se $f$ e $g$ sono due funzioni appartenenti a $LT_d$, $f \vee g$ (e quindi, automaticamente, anche $f \wedge g$) appartiene a $LT_d$? Ovvero, $LT_d$ (e/o $\widehat{LT}_d$) sono chiusi rispetto alla disgiunzione e alla congiunzione logica? Anche per quanto concerne la risposta a questa domanda, si rimanda al capitolo riguardante le approssimazioni di una funzione Booleana.

Goldmann, Håstad e Razborov [4] hanno dimostrato una bella relazione che lega le classi $LT_d$ e $\widehat{LT}_d$:

$$LT_d \subseteq \widehat{LT}_{d+1} \qquad \forall d \geq 1$$

cioè è sempre possibile simulare un circuito a soglia di peso esponenziale (e dimensione polinomiale e profondità costante) tramite un circuito di peso e dimensione polinomiale; tale simulazione richiede un solo strato in più di neuroni. I tre autori di questo risultato ne hanno data una dimostrazione esistenziale; successivamente, Goldmann e Karpinski hanno fornito una dimostrazione costruttiva [5], seppure complicata. Basandoci sul Teorema 2.1 (la cui dimostrazione, ricordiamo, è lunga e complessa), siamo in grado di darne qui una dimostrazione semplicissima.

**Teorema 2.5.** *Per ogni intero $d \geq 1$, $LT_d \subseteq \widehat{LT}_{d+1}$.*

*Dimostrazione.* Sia $f(x_1, x_2, \ldots, x_n) \in LT_d$, e sia $C_n$ un circuito a soglia a $n$ ingressi che calcola $f$. Grazie ai Teoremi 1.4 e 1.5 è facile costruire un circuito di profondità $d$ che calcola la funzione $g \equiv 0$, così che $g \in LT_d$. L'applicazione del Teorema 2.3 è sufficiente per concludere che $f \vee g = f \in \widehat{LT}_{d+1}$. ∎

Quest'ultimo teorema, insieme all'inclusione banale già menzionata $\widehat{LT}_d \subseteq LT_d$, ci consente di vedere come le classi $LT_d$ e $\widehat{LT}_d$ formino una gerarchia intrecciata:

$$\widehat{LT}_d \subseteq LT_d \subseteq \widehat{LT}_{d+1} \qquad \forall d \geq 1 \tag{2.1}$$

Una prima conseguenza dell'equazione (2.1) è espressa dal seguente teorema.

**Teorema 2.6.** $TC^0 = \widehat{TC^0}$.

*Dimostrazione.* Sia $f \in TC^0$; esiste allora un intero $d \geq 1$ tale che $f \in LT_d$. Per l'equazione (2.1) $f \in \widehat{LT}_{d+1}$, ovvero $f \in \widehat{TC^0}$.

Viceversa, sia $f \in \widehat{TC^0}$; esiste allora un intero $d \geq 1$ tale che $f \in \widehat{LT}_d$. Per l'equazione (2.1) $f \in LT_d$, ovvero $f \in TC^0$. ∎

A questo punto viene naturale chiedersi se le inclusioni dell'equazione (2.1) siano da considerarsi in senso stretto o in senso lato; abbiamo già visto nel Capitolo 1, ad esempio, che $\widehat{LT}_1 \subset LT_1$, in quanto Håstad ha dimostrato che esiste almeno una funzione threshold che necessita di pesi esponenziali per poter essere calcolata da un threshold gate. La successiva inclusione da esaminare è $LT_1 \subseteq \widehat{LT}_2$; ovvero, le classi $LT_1$ e $\widehat{LT}_2$ sono *separate* o coincidono? Per rispondere a questa domanda è necessario studiare meglio la struttura di $\widehat{LT}_2$ — ovvero le caratteristiche delle funzioni che vi appartengono — allo scopo di determinare, *se esiste*, una *funzione separatrice*, ossia una funzione Booleana $f$ per la quale valga $f \in \widehat{LT}_2$ e $f \notin LT_1$. Questo è quanto viene fatto nel prossimo paragrafo.

## 2.4 Funzioni che appartengono a $\widehat{LT}_2$

Abbiamo già visto, nel Capitolo 1, che le funzioni appartenenti a $LT_1$ sono accomunate dal fatto di essere linearmente separabili; in quell'occasione abbiamo anche visto che esistono funzioni che *non sono* linearmente separabili: una di queste è `PARITY`, come illustrato dal Teorema 1.3. Per comprendere meglio le caratteristiche delle funzioni che compongono $\widehat{LT}_2$, cominciamo col dare la seguente definizione.

**Definizione 2.10.** Una funzione Booleana $f(x_1, x_2, \ldots, x_n)$ si dice *simmetrica* se esiste un insieme $S = \{s_1, s_2, \ldots, s_k\} \subseteq \{0, 1, \ldots, n\}$ tale che:

$$f(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i \in S$$

■

Più semplicemente, una funzione è simmetrica se dipende esclusivamente dai valori che può assumere la quantità $\sum_{i=1}^{n} x_i$. Questa caratteristica delle funzioni simmetriche può essere utilizzata per dimostrare il prossimo teorema, dovuto ad Hajnal, Maass, Pudlák, Szegedy e Turán [7].

**Teorema 2.7.** *Sia $f(x_1, x_2, \ldots, x_n)$ una funzione Booleana simmetrica; allora $f \in \widehat{LT}_2$.*

*Dimostrazione.* Sia $S = \{s_1, s_2, \ldots, s_k\} \subseteq \{0, 1, \ldots, n\}$ l'insieme dei valori di $\sum_{i=1}^{n} x_i$ che definiscono $f$. Ricordando la definizione delle funzioni $T^n_{\geq k}$ e $T^n_{\leq k}$ data nel Capitolo 1, e indicando per brevità il vettore degli ingressi $(x_1, x_2, \ldots, x_n)$ con $\mathbf{x}$, dal fatto che:

$$T^n_{\geq s_j}(\mathbf{x}) + T^n_{\leq s_j}(\mathbf{x}) = \begin{cases} 2 & \text{se } \sum_{i=1}^{n} x_i = s_j \\ 1 & \text{altrimenti} \end{cases}$$

è facile rendersi conto che:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{j=1}^{k} \left(T^n_{\geq s_j}(\mathbf{x}) + T^n_{\leq s_j}(\mathbf{x})\right) - k = \\ &= step\left(\sum_{j=1}^{k} \left(T^n_{\geq s_j}(\mathbf{x}) + T^n_{\leq s_j}(\mathbf{x})\right) - (k+1)\right) = \\ &= T^{2k}_{\geq k+1}\left(T^n_{\geq s_1}(\mathbf{x}), T^n_{\leq s_1}(\mathbf{x}), \ldots, T^n_{\geq s_k}(\mathbf{x}), T^n_{\leq s_k}(\mathbf{x})\right) \end{aligned}$$

Si osservi che il numero degli argomenti di $T^{2k}_{\geq k+1}$ che valgono 1 è sempre $k$ o $k+1$. Per il Teorema 1.7, si può allora costruire un circuito di dimensione $2k+1 \leq 2n+3$ (poiché $k \leq n+1$), profondità 2 e peso polinomiale che calcola $f$, quindi $f \in \widehat{LT}_2$. ∎

La tecnica costruttiva usata nella dimostrazione del Teorema 2.7 può essere generalizzata, mostrando così che una classe ben più ampia di funzioni Booleane appartengono a $\widehat{LT}_2$.

Prima di estendere il Teorema 2.7, osserviamo che esso ci fornisce gli strumenti sufficienti ad affrontare la questione della separazione fra le classi $LT_1$ e $\widehat{LT}_2$, consentendoci di dimostrare che la funzione `PARITY` è una funzione separatrice.

**Teorema 2.8.** `PARITY` $\in \widehat{LT}_2$.

*Dimostrazione.* Per ogni intero $n \geq 1$, consideriamo la funzione $\texttt{PARITY}_n(x_1, x_2, \ldots, x_n)$. Essa vale 1 se e solo se vengono presentati all'ingresso un numero dispari di 1; ovvero se e solo se il valore assunto dalla quantità $\sum_{i=1}^{n} x_i$ è dispari. Questo significa che $\texttt{PARITY}_n$ è una funzione simmetrica, con insieme di definizione:

$$\begin{aligned} S_p &= \{1, 3, 5, \ldots, n-1\} && \text{per } n \text{ pari} \\ S_d &= \{1, 3, 5, \ldots, n\} && \text{per } n \text{ dispari} \end{aligned}$$

Per il Teorema 2.7, $\texttt{PARITY}_n \in \widehat{LT}_2$ per ogni $n$, cioè `PARITY` $\in \widehat{LT}_2$. ∎

Unendo il fatto che `PARITY` $\in \widehat{LT}_2$ con quello, già menzionato, che `PARITY` $\notin LT_1$, possiamo concludere che:

$$LT_1 \subset \widehat{LT}_2$$

Che la classe $\widehat{LT}_2$ non sia una classe di funzioni banali è testimoniato dal fatto che la funzione `PARITY` vi appartiene: è ben noto dalla teoria dei circuiti logici (si veda ad esempio la tesi di dottorato di Håstad [8]) che non esiste alcun circuito logico di profondità 2 e dimensione polinomiale che sia in grado di calcolarla. D'altra parte, per i Teoremi 1.4 e 1.9, ogni funzione computabile da un circuito logico di profondità 2 e dimensione polinomiale (e con fan-in illimitato) appartiene a $\widehat{LT}_2$: basta sostituire ogni porta logica del circuito con un

neurone equivalente. Da ciò segue immediatamente che *ogni funzione che ha un numero polinomiale di mintermini appartiene a* $\widehat{LT}_2$: basta fare la disgiunzione dei mintermini, ciascuno dei quali appartiene a $\widehat{LT}_1$ per il Teorema 1.8. Per il Teorema 1.9, poi, possiamo immediatamente affermare che *ogni funzione che vale* 0 *per un numero polinomiale di configurazioni di ingresso appartiene a* $\widehat{LT}_2$. Questa osservazione è già un'indicazione delle difficoltà cui si va incontro nel cercare una funzione $f$ che non appartenga a $\widehat{LT}_2$ [4]: tale funzione dovrà necessariamente assumere sia il valore 1 che il valore 0 per un numero esponenziale di configurazioni di ingresso. Se al lettore venisse in mente che questa può anche essere una condizione sufficiente, basti osservare che $\texttt{PARITY}_n$ vale 1 per $2^{n-1}$ configurazioni di ingresso, e vale 0 per le restanti $2^{n-1}$.

Mostriamo ora che è possibile estendere il Teorema 2.7 al caso in cui la funzione Booleana $f(x_1, x_2, \ldots, x_n)$ dipenda da $S = \sum_{i=1}^{n} w_i x_i$. Il prossimo teorema, dovuto a Siu e Roychowdhury [16], afferma infatti che se la funzione $f(x_1, x_2, \ldots, x_n)$ vale 1 per valori di $S$ che appartengono ad un numero al più polinomiale di intervalli, allora $f \in \widehat{LT}_2$.

**Teorema 2.9.** *Sia* $S = \sum_{i=1}^{n} w_i x_i$, *e siano* $[l_1, u_1], [l_2, u_2], \ldots, [l_N, u_N]$ *un numero polinomiale di intervalli disgiunti di valori di* $S$. *Se* $f(x_1, x_2, \ldots, x_n)$ *è una funzione Booleana tale che:*

$$f(x_1, x_2, \ldots, x_n) = 1 \iff S \in [l_j, u_j] \quad \textit{per qualche } j = 1, 2, \ldots, N$$

*allora* $f \in \widehat{LT}_2$.

*Dimostrazione.* Per $j = 1, 2, \ldots, N$, si considerino le seguenti quantità:

$$y_{l_j} = step\left(\sum_{i=1}^{n} w_i x_i - l_j\right) = step\big(S - l_j\big) = \begin{cases} 1 & \text{se } S \geq l_j \\ 0 & \text{se } S < l_j \end{cases}$$

$$y_{u_j} = step\left(u_j - \sum_{i=1}^{n} w_i x_i\right) = step\big(u_j - S\big) = \begin{cases} 1 & \text{se } S \leq u_j \\ 0 & \text{se } S > u_j \end{cases}$$

Come si vede chiaramente dalla loro definizione, ciascuna di queste quantità è calcolabile con un singolo threshold gate, che potrà all'occorrenza avere pesi esponenziali.

Si noti che se $f(x_1, x_2, \ldots, x_n) = 0$ allora $S \notin [l_j, u_j]$ per ogni $j = 1, 2, \ldots, N$. Questo significa che:

$$y_{l_j} + y_{u_j} = 1 \qquad \text{per ogni } j = 1, 2, \ldots, N$$

[4] Si noti che è di fondamentale importanza chiedersi se esiste o meno una funzione Booleana $f$ che *non appartiene* a $\widehat{LT}_2$. Infatti, in caso di risposta negativa si può subito concludere che $\widehat{LT}_2 = LT_2 = \widehat{LT}_3 = \ldots = TC^0 = BIN$; in caso di risposta affermativa, invece, diventa interessante determinare $d^* = \min\{d : f \in \widehat{LT}_d \text{ (oppure } f \in LT_d)\}$, così che $f$ faccia da funzione separatrice per le classi $\widehat{LT}_2$ e $\widehat{LT}_{d^*}$ (rispettivamente, $\widehat{LT}_2$ e $LT_{d^*}$).

e quindi:

$$\sum_{j=1}^{N}(y_{l_j} + y_{u_j}) - N = N - N = 0$$

D'altra parte, se $f(x_1, x_2, \ldots, x_n) = 1$ allora $S \in [l_j, u_j]$ per qualche $j \in \{1, 2, \ldots, N\}$. In tal caso avremo:

$$\begin{aligned} &y_{l_j} + y_{u_j} = 2 \\ &y_{l_k} + y_{u_k} = 1 \qquad \text{per ogni } k \neq j \end{aligned}$$

e quindi:

$$\sum_{j=1}^{N}(y_{l_j} + y_{u_j}) - N = N + 1 - N = 1$$

Possiamo allora concludere che valgono le seguenti uguaglianze:

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) &= \sum_{j=1}^{N}(y_{l_j} + y_{u_j}) - N = \\ &= step\left(\sum_{j=1}^{N}(y_{l_j} + y_{u_j}) - N - 1\right) \end{aligned} \tag{2.2}$$

Dall'uguaglianza (2.2) si ottiene subito un circuito di profondità 2 che calcola $f$. Tale circuito sarà costituito da un primo strato di dimensione $2N$ (e quindi polinomiale), formato dai threshold gate che calcolano il valore di ciascuna delle quantità $y_{l_j}$ e $y_{u_j}$, e da un threshold gate di uscita che calcola la quantità indicata nell'ultimo membro dell'equazione (2.2). Come è già stato detto, i neuroni del primo strato potranno occasionalmente avere pesi esponenziali, ma il neurone d'uscita avrà sempre pesi e soglia polinomiali; pertanto, si può concludere che $f \in \widetilde{LT}_2$, ovvero che $f \in \widehat{LT}_2$ per il Teorema 2.1. ∎

Si noti che nell'enunciato del Teorema 2.9 *non* viene richiesto che i valori $w_1, w_2, \ldots, w_n$ che concorrono a formare il valore di $S$ siano polinomiali: è sufficiente che all'interno dell'insieme di tutti i valori assunti dalla quantità $S$ (i quali possono quindi essere anche esponenziali) siano individuabili un numero al più polinomiale di intervalli $[l_1, u_1], [l_2, u_2], \ldots, [l_N, u_N]$ tali che $f$ vale 1 se e solo se $S$ appartiene a uno di questi intervalli. Ciò si verifica banalmente qualora i valori $w_1, w_2, \ldots, w_n$ siano polinomiali: in tal caso, infatti, $S$ potrà assumere al più un numero polinomiale di valori, e di conseguenza si potrà sempre individuare un numero polinomiale di intervalli di tali valori in corrispondenza dei quali $f$ vale 1 (e al di fuori dei quali $f$ vale 0).

Il fatto che il Teorema 2.9 costituisce un'estensione del Teorema 2.7 può essere evidenziato meglio modificando la dimostrazione del primo per farla aderire a quella del secondo.

*Dimostrazione alternativa del Teorema 2.9.* Si definiscano le seguenti funzioni Booleane:

$$S^n_{\geq k}(x_1, x_2, \ldots, x_n) = 1 \iff S = \sum_{i=1}^{n} w_i x_i \geq k$$

$$S^n_{\leq k}(x_1, x_2, \ldots, x_n) = 1 \iff S = \sum_{i=1}^{n} w_i x_i \leq k$$

In maniera del tutto analoga a quanto è stato fatto nella dimostrazione del Teorema 1.7, si può dimostrare facilmente che le funzioni $S^n_{\geq k}$ e $S^n_{\leq k}$ sono calcolabili da un singolo threshold gate, così che entrambe appartengono a $LT_1$. Poiché sia le quantità $w_1, w_2, \ldots, w_n$ che $k$ possono crescere in maniera esponenziale rispetto ad $n$, avremo che in generale $S^n_{\geq k}$ e $S^n_{\leq k}$ *non* appartengono a $\widehat{LT}_1$.

È inoltre immediato verificare che le funzioni $S^n_{\geq l_j}$ e $S^n_{\leq u_j}$ calcolano rispettivamente i valori $y_{l_j}$ e $y_{u_j}$ della dimostrazione precedente; possiamo allora riscrivere l'equazione (2.2) come segue:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{j=1}^{N} \big(S^n_{\geq l_j}(\mathbf{x}) + S^n_{\leq u_j}(\mathbf{x})\big) - N = \\ &= step\left(\sum_{j=1}^{N} \big(S^n_{\geq l_j}(\mathbf{x}) + S^n_{\leq u_j}(\mathbf{x})\big) - (N+1)\right) = \\ &= T^{2N}_{\geq N+1}\big(S^n_{\geq l_1}(\mathbf{x}), S^n_{\leq u_1}(\mathbf{x}), \ldots, S^n_{\geq l_N}(\mathbf{x}), S^n_{\leq u_N}(\mathbf{x})\big) \end{aligned}$$

Si osservi che, per ogni configurazione di ingresso $\mathbf{x} \in \{0,1\}^n$ fissata, il numero di argomenti di $T^{2N}_{\geq N+1}$ che valgono 1 è sempre $N$ o $N+1$.

È allora possibile costruire un circuito di profondità 2 e dimensione $2N+1$ (quindi polinomiale) che calcola $f$. Per il Teorema 1.7, il neurone d'uscita di tale circuito ha pesi e soglia polinomiali: pertanto, si può concludere che $f \in \widetilde{LT}_2$, ovvero che $f \in \widehat{LT}_2$ per il Teorema 2.1. ∎

Illustriamo ora come la tecnica dimostrativa utilizzata per i Teoremi 2.7 e 2.9 possa essere ancora una volta applicata per individuare un'altra classe di funzioni che appartengono a $\widehat{LT}_2$. Per fare ciò, introduciamo il concetto di *variazione*.

**Definizione 2.11.** Sia $f(x_1, x_2, \ldots, x_n) \in BIN_n$, e sia $\mathbf{v}$ il vettore $\big(f(0,0,\ldots,0), f(0,0,\ldots,1), \ldots, f(1,1,\ldots,1)\big)$ a $2^n$ componenti binarie (in pratica, $\mathbf{v}$ è la tabella di verità di $f$). Si dice che si ha una *variazione* in $f$ quando esiste un indice $i$ tale che $(\mathbf{v})_i \neq (\mathbf{v})_{i+1}$. ∎

Siamo particolarmente interessati alle famiglie di funzioni in cui il numero di variazioni cresce al più come un polinomio $p(n)$, poiché siamo in grado, con il seguente teorema, di dimostrare che tali funzioni appartengono a $\widehat{LT}_2$.

**Teorema 2.10.** *Sia $\boldsymbol{\mathcal{F}}$ una famiglia di funzioni per le quali il numero di variazioni cresce al più in maniera polinomiale; allora $\boldsymbol{\mathcal{F}} \in \widehat{LT}_2$.*

*Dimostrazione.* Sia $f(x_1, x_2, \ldots, x_n) \in \boldsymbol{\mathcal{F}}$ e sia $\mathbf{v}$ il vettore $[f(0,0,\ldots,0), f(0,0,\ldots,1), \ldots, f(1,1,\ldots,1)]$. Per ipotesi, possiamo suddividere $\mathbf{v}$ in un numero $l$ di sottosequenze, con $l \leq p(n)$ per un opportuno polinomio $p(n)$, in modo tale che ciascuna sottosequenza sia formata interamente da 1 o da 0. Consideriamo solo le sequenze costituite da 1, e indichiamo con:

$$[i_1, j_1], [i_2, j_2], \ldots, [i_m, j_m]$$

gli intervalli di indici di $\mathbf{v}$ che delimitano tali sequenze. Ovviamente varrà $m \leq l \leq p(n)$.

Definiamo ora le seguenti funzioni Booleane:

$$D^n_{\geq k}(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i 2^{i-1} \geq k$$
$$D^n_{\leq k}(x_1, x_2, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i 2^{i-1} \leq k$$

In pratica, se indichiamo con $\langle \mathbf{x} \rangle$ il numero naturale la cui rappresentazione binaria è $x_n x_{n-1} \cdots x_1$ (con $x_n$ bit più significativo), possiamo riscrivere la definizione come:

$$D^n_{\geq k}(x_1, x_2, \ldots, x_n) = 1 \iff \langle \mathbf{x} \rangle \geq k$$
$$D^n_{\leq k}(x_1, x_2, \ldots, x_n) = 1 \iff \langle \mathbf{x} \rangle \leq k$$

Poiché esiste una chiara corrispondenza biunivoca tra i valori di $\langle \mathbf{x} \rangle$ e gli indici di $\mathbf{v}$, numeriamo gli indici di $\mathbf{v}$ da 0 a $2^n - 1$; abbiamo così che $D^n_{\geq k}$ vale 1 sugli indici $\geq k$ e $D^n_{\leq k}$ vale 1 sugli indici $\leq k$.

Sia ora $\mathbf{x}$ una configurazione degli ingressi per la quale $f(\mathbf{x}) = 1$; allora $\langle \mathbf{x} \rangle \in [i_r, j_r]$ per qualche $r \in \{1, 2, \ldots, m\}$. Questo significa che:

$$D^n_{\geq i_r}(\mathbf{x}) + D^n_{\leq j_r}(\mathbf{x}) = 2$$
$$D^n_{\geq i_l}(\mathbf{x}) + D^n_{\leq j_l}(\mathbf{x}) = 1 \qquad \text{per ogni } l \neq r$$

e quindi:

$$\sum_{r=1}^{m} \left( D^n_{\geq i_r}(\mathbf{x}) + D^n_{\leq j_r}(\mathbf{x}) \right) - m = m + 1 - m = 1$$

Se invece $\mathbf{x}$ è una configurazione degli ingressi per la quale $f(\mathbf{x}) = 0$, $\langle \mathbf{x} \rangle \notin [i_r, j_r]$ per ogni $r = 1, 2, \ldots, m$. In tal caso avremo:

$$D^n_{\geq i_r}(\mathbf{x}) + D^n_{\leq j_r}(\mathbf{x}) = 1 \qquad \text{per ogni } r = 1, 2, \ldots, m.$$

e quindi:

$$\sum_{r=1}^{m} \left( D^n_{\geq i_r}(\mathbf{x}) + D^n_{\leq j_r}(\mathbf{x}) \right) - m = m - m = 0$$

Possiamo allora concludere che valgono le seguenti uguaglianze:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{r=1}^{m} \left( D^n_{\geq i_r}(\mathbf{x}) + D^n_{\leq j_r}(\mathbf{x}) \right) - m = \\ &= step\left( \sum_{r=1}^{m} \left( D^n_{\geq i_r}(\mathbf{x}) + D^n_{\leq j_r}(\mathbf{x}) \right) - (m+1) \right) = \\ &= T^{2m}_{\geq m+1}\left( D^n_{\geq i_1}(\mathbf{x}), D^n_{\leq j_1}(\mathbf{x}), \ldots, D^n_{\geq i_m}(\mathbf{x}), D^n_{\leq j_m}(\mathbf{x}) \right) \end{aligned}$$

Si osservi che, per ogni configurazione di ingresso $\mathbf{x} \in \{0,1\}^n$ fissata, il numero di argomenti di $T^{2m}_{\geq m+1}$ che valgono 1 è sempre $m$ o $m+1$.

Per il Teorema 1.7 abbiamo che $T^{2m}_{\geq m+1} \in \widehat{LT}_1$, mentre si vede in modo analogo che ciascun $D^n_{\geq k}$ e ciascun $D^n_{\leq k}$ appartengono a $LT_1$. Esiste pertanto un circuito di dimensione $2m+1$ e profondità 2 che calcola $f$; inoltre, in tale circuito il neurone d'uscita ha pesi polinomiali, così che $f \in \widetilde{LT}_2$. Per il Teorema 2.1, $f \in \widehat{LT}_2$. ∎

Questo teorema è molto più forte dell'affermazione fatta precedentemente, per la quale una funzione avente un numero polinomiale di 1 o di 0 appartiene a $\widehat{LT}_2$; qui si afferma che il numero di 1 o di 0 può anche essere esponenziale, basta che il numero di *variazioni* sia polinomiale. Disgraziatamente, neanche questo criterio è sufficiente a trovare una funzione separatrice per le classi $\widehat{LT}_2$ e $LT_2$, o per $\widehat{LT}_2$ e $\widehat{LT}_3$: nella funzione $\texttt{PARITY}_n$, che appartiene a $\widehat{LT}_2$, le configurazioni per le quali la funzione vale 1 sono frapposte a quelle per cui la funzione vale 0, dando luogo a un numero esponenziale di variazioni.

## 2.5 Abbassare la profondità di un circuito

Siamo ora in grado di illustrare una semplice tecnica che può risultare utile — in alcuni casi — per abbassare la profondità di un circuito a soglia che calcola una funzione data.

Nella dimostrazione del Teorema 2.7, abbiamo evidenziato che il numero degli argomenti della funzione $T^{2k}_{\geq k+1}$ (calcolata dal threshold gate d'uscita del circuito) che valgono 1 è sempre $k$ o $k+1$, ed è $k+1$ in tutte e sole le configurazioni di ingresso del circuito per le quali la funzione da esso calcolata vale 1. Questa notevole proprietà ci consente di esprimere la funzione $f$ nel modo seguente:

$$f(\mathbf{x}) = \sum_{j=1}^{k} \left( T^n_{\geq s_j}(\mathbf{x}) + T^n_{\leq s_j}(\mathbf{x}) \right) - k$$

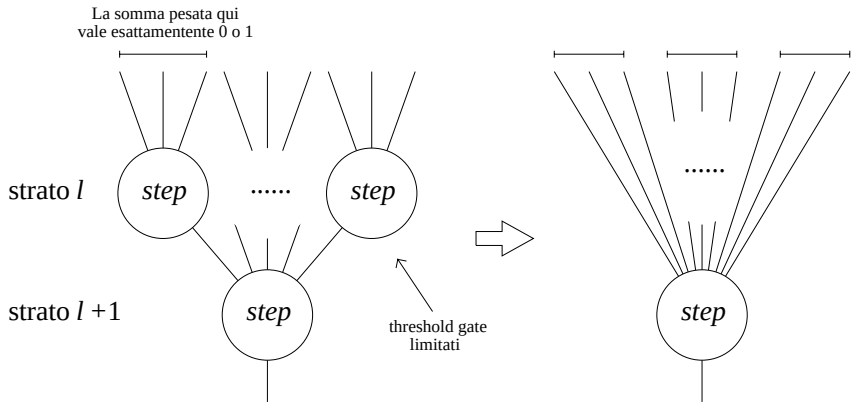



Figura 2.2: *Trucco che consente di abbassare di* 1 *la profondità di un circuito che contiene uno strato di threshold gate* limitati.

Vediamo allora che il threshold gate d'uscita del circuito applica inutilmente la propria funzione di trasferimento *step*, in quanto per trovare il valore di $f$ sarebbe sufficiente calcolare la sommatoria di cui sopra, la quale di per sé vale già 0 o 1. Quando si verifica questa situazione, chiamiamo *limitato* il threshold gate coinvolto.

La tecnica che consente di abbassare la profondità di alcuni circuiti a soglia si basa sul fatto che una combinazione lineare di combinazioni lineari è ancora una combinazione lineare, ed è la seguente:

> *se in un dato circuito a soglia tutti i neuroni dello strato $l$ sono threshold gate limitati, è possibile eliminare l'intero strato mandando le uscite dei neuroni dello strato $l-1$* direttamente *in ingresso a quelli dello strato $l+1$.*

Questa tecnica, che in letteratura si chiama "trucco dei threshold gate limitati", è illustrata schematicamente nella Figura 2.2. I pesi dei *nuovi* ingressi ai neuroni dello strato $l$ (che sono i neuroni del *vecchio* strato $l+1$) saranno uguali ai pesi dei *vecchi* ingressi dello strato $l+1$ *moltiplicati* per i corrispondenti pesi dei *vecchi* ingressi dello strato $l$; in maniera analoga vanno aggiustate anche le soglie dei neuroni del *nuovo* strato $l$.

Come si può notare dalla Figura 2.2, può capitare che il fan-in dei neuroni del nuovo strato $l$ aumenti notevolmente, soprattutto se il fan-out di qualcuno dei neuroni dello strato $l$ del circuito di partenza è maggiore di 1. È tuttavia facile rendersi conto che, applicando il trucco dei threshold gate limitati a un circuito avente un numero polinomiale di collegamenti (e quindi, per quanto visto all'inizio del capitolo, di dimensione polinomiale) il circuito risultante avrà anch'esso un numero al più polinomiale di collegamenti.

Naturalmente, l'unico caso in cui il trucco non può essere applicato è quello in cui il threshold gate limitato sia il neurone d'uscita del circuito.

Alla luce di tutto ciò, l'utilità pratica del Teorema 2.7 risulta essere duplice in quanto, oltre a mostrare che ogni funzione simmetrica appartiene a $\widehat{LT}_2$, ci dice che il neurone d'uscita del circuito che la calcola è un threshold gate limitato.

Per comprendere meglio la portata di questo risultato, definiamo i *gate simmetrici* come dei dispositivi che calcolano funzioni simmetriche dei propri ingressi, e consideriamo un circuito $C_{symm}$ di dimensione polinomiale e profondità $d$ formato interamente da gate simmetrici[5]. Il Teorema 2.7, applicato ad ogni singolo gate del circuito, ci consente di dire che la funzione $f$ calcolata da $C_{symm}$ appartiene a $\widehat{LT}_{2d}$. L'osservazione precedente ci permette poi di affermare che, nel circuito a soglia di profondità $2d$ così ottenuto, tutti gli strati pari sono formati da threshold gate limitati; siamo allora in grado di applicare il trucco dei threshold gate limitati $d-1$ volte (poiché sul neurone d'uscita non lo si può applicare) pervenendo a un circuito a soglia di profondità $d+1$ (e dimensione e peso polinomiali).

Come il lettore avrà già notato, i Teoremi 2.9 e 2.10 affermano che le funzioni simmetriche non sono le sole che possono essere calcolate da un circuito di dimensione polinomiale e profondità 2 il cui threshold gate d'uscita è *limitato*; è anche in questo senso che essi estendono il Teorema 2.7. Anche per le funzioni che soddisfano le ipotesi degli enunciati dei Teoremi 2.9 e 2.10, quindi, è possibile definire dei gate che le calcolano, e costruire con tali gate dei circuiti di dimensione polinomiale e profondità $d$. In maniera del tutto analoga a quanto fatto con i gate simmetrici, anche in questi casi è possibile applicare il trucco dei threshold gate limitati, e dimostrare che le funzioni calcolate da tali circuiti appartengono alla classe $\widehat{LT}_{d+1}$.

## 2.6 Separazione delle classi $\widehat{LT}_2$ e $\widehat{LT}_3$

Riprendiamo il discorso sulla separazione delle classi della gerarchia (2.1) interrotto precedentemente. La separazione di $\widehat{LT}_2$ da $LT_2$ è stata dimostrata da Goldmann, Håstad e Razborov [4], definendo nuove classi di funzioni Booleane e utilizzando alcuni risultati della Complessità di Comunicazione, una tecnica sofisticata di analisi delle funzioni Booleane che non verrà presa in considerazione in questi appunti. La separazione tra le classi $\widehat{LT}_2$ ed $\widehat{LT}_3$ è stata dimostrata da Hajnal, Maass, Pudlák, Szegedy e Turán [7], utilizzando come funzione separatrice una generalizzazione della funzione `PARITY` chiamata `INNER PRODUCT`

[5]Si noti che stiamo sottointendendo che tutti i pesi associati ai collegamenti tra due gate simmetrici siano uguali a 1. Il ragionamento si applica anche ai circuiti nei quali ai collegamenti tra i gate simmetrici siano associati dei pesi interi maggiori di 1, purché polinomiali; in tal caso, infatti, possiamo sempre costruire un circuito equivalente in cui tutti i pesi sono uguali a 1, duplicando i collegamenti e i sottocircuiti del circuito di partenza, mantenendo polinomiale la dimensione del circuito risultante.

MOD 2 (prodotto interno modulo 2), abbreviata con IP [6]:

$$\mathtt{IP}_n(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

Recentemente, le tecniche derivate dalla Complessità di Comunicazione (in particolare dal cosiddetto Discriminator Lemma) sono state estese da Goldmann [6], che è riuscito così a migliorare il risultato di Hajnal, Maass, Pudlák, Szegedy e Turán, fornendo una semplice dimostrazione del fatto che $\mathtt{IP} \notin LT_2$.

Osserviamo anzitutto che la funzione $\mathtt{IP}_n$ elude tutti i metodi visti finora per mostrare che una funzione appartiene a $\widehat{LT}_2$: in particolare, $\mathtt{IP}_n$ non è simmetrica, e non ha né un numero polinomiale di mintermini né un numero polinomiale di zeri, né un numero polinomiale di variazioni, dato che si riduce a $\mathtt{PARITY}_n$ se imponiamo $x_i = y_i$ per ogni $i = 1, 2, \ldots, n$. Calcoliamo comunque il numero di mintermini di $\mathtt{IP}_n$, dato che sarà utile in seguito; una definizione alternativa di $\mathtt{IP}_n$ è la seguente:

$$\begin{aligned}
&\mathtt{IP}_1(x_1, y_1) = x_1 \wedge y_1 \\
&\mathtt{IP}_n(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \\
&\qquad \mathtt{IP}_{n-1}(x_1, x_2, \ldots, x_{n-1}, y_1, y_2, \ldots, y_{n-1}) \oplus (x_n \wedge y_n)
\end{aligned}$$

Detto allora $a_n$ il numero di mintermini di $\mathtt{IP}_n$, si può scrivere la seguente successione ricorrente:

$$\begin{aligned}
&a_1 = 1 \\
&a_n = 3 \cdot a_{n-1} + 4^{n-1} - a_{n-1} = 2 \cdot a_{n-1} + 4^{n-1}
\end{aligned}$$

la cui soluzione, come è facile verificare, è:

$$a_n = \frac{1}{2}(4^n - 2^n) = 2^{2n-1} - 2^{n-1} \tag{2.3}$$

Il numero di configurazioni di ingresso in corrispondenza delle quali $\mathtt{IP}_n = 0$ sarà allora:

$$4^n - a_n = 2^{2n} - \frac{1}{2} \cdot 2^{2n} + 2^{n-1} = 2^{2n-1} + 2^{n-1} \tag{2.4}$$

Hajnal, Maass, Pudlák, Szegedy e Turán hanno dimostrato che se si vuole calcolare la funzione $\mathtt{IP}_n$ con un circuito a soglia di profondità 2 e peso polinomiale

[6] Il nome deriva dal fatto che, considerati i due vettori a $n$ componenti $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ e $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \{0,1\}^n$, $\mathtt{IP}_n$ può essere riscritto come:

$$\begin{aligned}
\mathtt{IP}_n(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) &= x_1 y_1 \oplus x_2 y_2 \oplus \ldots \oplus x_n y_n = \\
&= (x_1 y_1 + x_2 y_2 + \ldots + x_n y_n) \text{ MOD } 2 = \\
&= (\mathbf{x}^T \mathbf{y}) \text{ MOD } 2
\end{aligned}$$

dove $\mathbf{x}^T \mathbf{y}$ indica l'usuale prodotto interno fra vettori di $\Re^n$.

la dimensione deve necessariamente essere esponenziale; da ciò segue chiaramente che $\mathtt{IP} \notin \widehat{LT}_2$. Esponiamo qui la dimostrazione completa di questo fatto; poiché tale dimostrazione fa uso della nozione di *ε-discriminatore*, diamo anzitutto la seguente definizione.

**Definizione 2.12.** Siano $A, B \subseteq \{0,1\}^n$ due insiemi disgiunti, sia $C$ un circuito a soglia a $n$ ingressi che calcola la funzione $f_C \in BIN_n$, e sia $\varepsilon > 0$ un numero reale fissato. Indicando con $P_A$ e $P_B$ rispettivamente le distribuzioni di probabilità *uniformi* definite sugli insiemi $A$ e $B$, diciamo che $C$ *è un ε-discriminatore per* $A$ *e* $B$ se:

$$\Big| P_A[f_C(\mathbf{x}) = 1] - P_B[f_C(\mathbf{x}) = 1] \Big| \geq \varepsilon \qquad \forall\, \mathbf{x} \in \{0,1\}^n$$

■

La definizione precedente può essere estesa alle funzioni Booleane, considerando come insiemi $A$ e $B$ le antiimmagini di 0 e 1.

**Definizione 2.13.** Se $f(x_1, x_2, \ldots, x_n) \in BIN_n$, $C$ *è un ε-discriminatore per* $f$ se è un $\varepsilon$-discriminatore per gli insiemi:

$$A = \{\mathbf{x} \in \{0,1\}^n \mid f(\mathbf{x}) = 0\}$$
$$B = \{\mathbf{x} \in \{0,1\}^n \mid f(\mathbf{x}) = 1\}$$

■

Il prossimo lemma riduce il problema di dimostrare un lower bound sulla dimensione dei circuiti di profondità $d$ a quello di dimostrare che non esistono $\varepsilon$-discriminatori di profondità $d-1$ per valori grandi di $\varepsilon$ (ovvero — si veda l'enunciato del lemma — per valori piccoli (polinomiali) di $\alpha$). Si noti che il lemma non necessita realmente di distribuzioni di probabilità *uniformi* sugli insiemi $A$ e $B$, e vale quindi per distribuzioni di probabilità arbitrarie.

**Lemma 2.11.** *Sia* $f \in \widehat{LT}_d$*, e sia* $C$ *un circuito di dimensione e peso polinomiali (e profondità* $d$*) che la calcola. Siano poi* $C_1, C_2, \ldots, C_m$ *i sottocircuiti di profondità* $d-1$ *ottenuti da* $C$ *rimuovendo il threshold gate d'uscita, e siano* $\alpha_1, \alpha_2, \ldots, \alpha_m$ *e* $k$ *rispettivamente i pesi e la soglia di quest'ultimo. Posto allora* $\alpha = \sum_{i=1}^{m} |\alpha_i|$ *e presi due insiemi disgiunti* $A, B \subseteq \{0,1\}^n$ *tali che:*

$$f(\mathbf{x}) = 1 \qquad \forall\, \mathbf{x} \in A$$
$$f(\mathbf{x}) = 0 \qquad \forall\, \mathbf{x} \in B$$

*esiste un indice* $i \in \{1, 2, \ldots, m\}$ *tale che il sottocircuito* $C_i$ *è un* $(1/\alpha)$*-discriminatore per* $A$ *e* $B$*.*

*Dimostrazione.* Si noti che $A$ e $B$ sono sottoinsiemi delle antiimmagini di $f$. Dette $f_1, f_2, \ldots, f_m \in \widehat{LT}_{d-1}$ le funzioni calcolate dai sottocircuiti $C_1, C_2, \ldots, C_m$, il circuito $C$ calcola la funzione $f$ nel modo seguente:

$$f(\mathbf{x}) = 1 \iff \sum_{i=1}^{m} \alpha_i f_i(\mathbf{x}) \geq k$$

Definiamo la variabile casuale $f_i^A(\mathbf{x})$ (rispettivamente $f_i^B(\mathbf{x})$) come il valore di $f_i(\mathbf{x})$ quando $\mathbf{x}$ è distribuito con probabilità uniforme su $A$ (rispettivamente su $B$). Supponendo $\alpha_1, \alpha_2, \ldots, \alpha_m$ e $k$ interi, avremo:

$$\sum_{i=1}^{m} \alpha_i f_i^A(\mathbf{x}) \geq k$$
$$\sum_{i=1}^{m} \alpha_i f_i^B(\mathbf{x}) \leq k-1 \quad \text{da cui} \quad -\sum_{i=1}^{m} \alpha_i f_i^B(\mathbf{x}) \geq 1-k$$

Consideriamo il valore atteso di queste due quantità; abbiamo:

$$E\left[\sum_{i=1}^{m} \alpha_i f_i^A(\mathbf{x})\right] = \sum_{i=1}^{m} \alpha_i E\Big[f_i^A(\mathbf{x})\Big] \geq k$$
$$E\left[-\sum_{i=1}^{m} \alpha_i f_i^B(\mathbf{x})\right] = -\sum_{i=1}^{m} \alpha_i E\Big[f_i^B(\mathbf{x})\Big] \geq 1-k$$

Sommando membro a membro, otteniamo:

$$\sum_{i=1}^{m} \alpha_i \Big( E\Big[f_i^A(\mathbf{x})\Big] - E\Big[f_i^B(\mathbf{x})\Big] \Big) \geq 1$$

Il primo membro della disuguaglianza può essere maggiorato come segue:

$$\begin{aligned}
\sum_{i=1}^{m} \alpha_i \Big( E\Big[f_i^A(\mathbf{x})\Big] - E\Big[f_i^B(\mathbf{x})\Big] \Big) &= \sum_{i=1}^{m} \alpha_i \cdot E\Big[f_i^A(\mathbf{x}) - f_i^B(\mathbf{x})\Big] \leq \\
&\leq \sum_{i=1}^{m} \alpha_i \cdot \max_{1 \leq i \leq m} \Big| f_i^A(\mathbf{x}) - f_i^B(\mathbf{x}) \Big| \leq \\
&\leq \max_{1 \leq i \leq m} \Big| f_i^A(\mathbf{x}) - f_i^B(\mathbf{x}) \Big| \cdot \sum_{i=1}^{m} |\alpha_i| = \\
&= \alpha \cdot \max_{1 \leq i \leq m} \Big| f_i^A(\mathbf{x}) - f_i^B(\mathbf{x}) \Big| = \\
&= \alpha \cdot \max_{1 \leq i \leq m} \Big| P_A[f_i(\mathbf{x}) = 1] - P_B[f_i(\mathbf{x}) = 1] \Big|
\end{aligned}$$

e quindi otteniamo:

$$\max_{1\leq i\leq m}\Big|P_A[f_i(\mathbf{x})=1]-P_B[f_i(\mathbf{x})=1]\Big|\geq\frac{1}{\alpha}$$

Esiste allora un indice $i\in\{1,2,\ldots,m\}$ tale che:

$$\Big|P_A[f_i(\mathbf{x})=1]-P_B[f_i(\mathbf{x})=1]\Big|\geq\frac{1}{\alpha}\qquad\forall\,\mathbf{x}\in\{0,1\}^n$$

ovvero $C_i$ è un $(1/\alpha)$-discriminatore per gli insiemi $A$ e $B$. ∎

Il seguente lemma è un caso speciale di un lemma di Lindsey riguardante le matrici di Hadamard[7].

**Lemma 2.12.** *Siano $X,Y\subseteq\{0,1\}^n$; definiti i due insiemi:*

$$A=\{(\mathbf{x},\mathbf{y})\in X\times Y\mid \mathtt{IP}_n(\mathbf{x},\mathbf{y})=1\}$$
$$B=\{(\mathbf{x},\mathbf{y})\in X\times Y\mid \mathtt{IP}_n(\mathbf{x},\mathbf{y})=0\}$$

*vale:*

$$\Big||A|-|B|\Big|\leq\sqrt{|X|\cdot|Y|\cdot 2^n}$$

∎

Siamo ora in grado di dimostrare il seguente teorema.

[7]Le *matrici di Hadamard*, dal nome di Jacques Hadamard (1865–1963), sono le matrici $n\times n$ con le seguenti proprietà:

- ogni elemento è 1 oppure $-1$;
- il prodotto scalare di due righe distinte vale zero.

Le matrici di Hadamard compaiono in certi problemi di Geometria e di Teoria dei Numeri, e sono state applicate di recente per la costruzione di codici cifrati per le comunicazioni spaziali.

Menzioniamo qui due proprietà che possono dare una prima idea di come sono fatte le matrici di Hadamard. La prima è che ogni riga, considerata come vettore di $\Re^n$, ha norma euclidea pari a $\sqrt{n}$; questa osservazione discende immediatamente dal fatto che gli elementi della matrice appartengono a $\{1,-1\}$.

La seconda proprietà è la seguente: *se $A$ è una matrice di Hadamard $n\times n$, con $n>2$, allora $n$ è un multiplo di 4.* La dimostrazione di questa proprietà è basata su due semplicissimi lemmi riguardanti i vettori di $\Re^n$, applicati alle righe delle matrici di Hadamard.

**Lemma.** *Se $X,Y,Z$ sono vettori ortogonali di $\Re^n$, allora:*

$$(X+Y)\cdot(X+Z)=\|X\|^2$$

∎

**Lemma.** *Dati tre vettori $X,Y,Z\in\Re^n$, se ciascuna delle componenti $x_i,y_i,z_i$ è 1 oppure $-1$, allora il prodotto $(x_i+y_i)(x_i+z_i)$ vale 0 oppure 4.* ∎

Si noti che, a dispetto della loro apparente semplicità, le matrici di Hadamard presentano ancora molti problemi insoluti, il più importante dei quali è quello di determinare tutti i valori di $n$ per i quali esiste una matrice di Hadamard di ordine $n$.

**Teorema 2.13.** *Sia $\varepsilon > 0$ un numero reale, e sia $p(n)$ un polinomio. Sia inoltre $C$ un circuito di profondità 2 e peso $\leq p(n)$ che calcola la funzione $\mathtt{IP}_n(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$. Se $n$ è sufficientemente grande, la dimensione di $C$ è almeno $2^{(1/2-\varepsilon)n}$.*

*Dimostrazione.* Sia $C$ un circuito che verifica le ipotesi dell'enunciato del teorema, e siano $C_1, C_2, \ldots, C_m$ i sottocircuiti ottenuti rimuovendo da $C$ il threshold gate d'uscita, come nell'enunciato del Lemma 2.11.

Poiché $C$ è di profondità 2, ciascun $C_i$ ($1 \leq i \leq m$) sarà un singolo threshold gate. Detti $\beta_1, \beta_2, \ldots, \beta_n, \gamma_1, \gamma_2, \ldots, \gamma_n$ ed $l$ rispettivamente i pesi e la soglia di $C_i$, questi emetterà un 1 in uscita se e solo se:

$$\sum_{j=1}^{n} \beta_j x_j + \sum_{j=1}^{n} \gamma_j y_j \geq l \tag{2.5}$$

Siano:

$$X_u = \left\{ \mathbf{x} \in \{0,1\}^n \;\middle|\; \sum_{j=1}^{n} \beta_j x_j = u \right\}$$

$$Y_u = \left\{ \mathbf{y} \in \{0,1\}^n \;\middle|\; \sum_{j=1}^{n} \gamma_j y_j \geq l - u \right\}$$

Dalla definizione di $X_u$, dato che per ogni $j = 1, 2, \ldots, n$ si ha che $x_j \in \{0,1\}$, si vede subito che $|u| \leq np(n)$. Inoltre, è facile rendersi conto che la disuguaglianza (2.5) viene verificata da tutte e sole le coppie $(\mathbf{x}, \mathbf{y})$ tali che $\mathbf{x} \in X_u$ e $\mathbf{y} \in Y_u$, così che possiamo definire l'insieme $V$ di coppie per le quali $C_i(\mathbf{x}, \mathbf{y}) = 1$ nel modo seguente:

$$V = \{(\mathbf{x}, \mathbf{y}) \mid C_i(\mathbf{x}, \mathbf{y}) = 1\} = \bigcup_{u=-np(n)}^{np(n)} X_u \times Y_u$$

Possiamo a questo punto applicare il Lemma 2.12 agli insiemi:

$$A_u = \{(\mathbf{x}, \mathbf{y}) \in X_u \times Y_u \mid \mathtt{IP}_n(\mathbf{x}, \mathbf{y}) = 1\}$$
$$B_u = \{(\mathbf{x}, \mathbf{y}) \in X_u \times Y_u \mid \mathtt{IP}_n(\mathbf{x}, \mathbf{y}) = 0\}$$

ottenendo:

$$\Big| |A_u| - |B_u| \Big| \leq \sqrt{|X_u| \cdot |Y_u| \cdot 2^n} \tag{2.6}$$

Definendo poi gli insiemi:

$$A = \{(\mathbf{x}, \mathbf{y}) \in V \mid \mathtt{IP}_n(\mathbf{x}, \mathbf{y}) = 1\} = \bigcup_{u=-np(n)}^{np(n)} A_u$$

$$B = \{(\mathbf{x}, \mathbf{y}) \in V \mid \mathtt{IP}_n(\mathbf{x}, \mathbf{y}) = 0\} = \bigcup_{u=-np(n)}^{np(n)} B_u$$

possiamo scrivere:

$$\Big||A| - |B|\Big| = \left|\left|\bigcup_{u=-np(n)}^{np(n)} A_u\right| - \left|\bigcup_{u=-np(n)}^{np(n)} B_u\right|\right|$$

Essendo gli insiemi $X_u$ disgiunti al variare di $u$, anche gli insiemi $A_u$ e $B_u$ saranno disgiunti, così che possiamo scrivere:

$$\begin{aligned}\Big||A| - |B|\Big| &= \left|\sum_{u=-np(n)}^{np(n)} |A_u| - \sum_{u=-np(n)}^{np(n)} |B_u|\right| = \\ &= \left|\sum_{u=-np(n)}^{np(n)} \Big(|A_u| - |B_u|\Big)\right| \leq \sum_{u=-np(n)}^{np(n)} \Big||A_u| - |B_u|\Big|\end{aligned}$$

dove la maggiorazione discende dalla disuguaglianza triangolare. Applicando la disuguaglianza (2.6) ad ogni termine della sommatoria otteniamo:

$$\Big||A| - |B|\Big| \leq \sum_{u=-np(n)}^{np(n)} \sqrt{|X_u| \cdot |Y_u| \cdot 2^n}$$

Essendo chiaramente $|X_u| \leq 2^n$ e $|Y_u| \leq 2^n$, abbiamo che:

$$\begin{aligned}\Big||A| - |B|\Big| &\leq \sum_{u=-np(n)}^{np(n)} \sqrt{2^n \cdot 2^n \cdot 2^n} = \sum_{u=-np(n)}^{np(n)} 2^{3n/2} = \\ &= \big[2np(n) + 1\big] \cdot 2^{3n/2} \leq (2n+1)p(n) \cdot 2^{3n/2}\end{aligned}$$

Definiti gli insiemi:

$$\begin{aligned}IP_1 &= \{(\mathbf{x}, \mathbf{y}) \mid \mathtt{IP}_n(\mathbf{x}, \mathbf{y}) = 1\} \\ IP_0 &= \{(\mathbf{x}, \mathbf{y}) \mid \mathtt{IP}_n(\mathbf{x}, \mathbf{y}) = 0\}\end{aligned}$$

abbiamo visto (equazioni (2.3) e (2.4)) che:

$$\begin{aligned}|IP_1| &= 2^{2n-1} - 2^{n-1} \\ |IP_0| &= 2^{2n-1} + 2^{n-1}\end{aligned}$$

Posto $\alpha = \sum_{i=1}^{m} |\alpha_i|$, se il circuito $C_i$ è un $(1/\alpha)$-discriminatore per la funzione $\mathtt{IP}_n$ allora $C_i$ è un $(1/\alpha)$-discriminatore per gli insiemi $IP_1$ e $IP_0$. Indicate con $P_{IP_1}$ e $P_{IP_0}$ le distribuzioni di probabilità uniformi definite rispettivamente su $IP_1$ e $IP_0$, questo significa che:

$$\Big|P_{IP_1}[C_i(\mathbf{x}, \mathbf{y}) = 1] - P_{IP_0}[C_i(\mathbf{x}, \mathbf{y}) = 1]\Big| \geq \frac{1}{\alpha}$$

Per $n$ sufficientemente grande, $|IP_1|$ e $|IP_0|$ sono asintotici a $2^{2n}$:

$$|IP_1| \sim |IP_0| \sim 2^{2n}$$

e quindi abbiamo:

$$\begin{aligned}\frac{1}{\alpha} &\leq \left| P_{IP_1}[C_i(\mathbf{x},\mathbf{y}) = 1] - P_{IP_0}[C_i(\mathbf{x},\mathbf{y}) = 1] \right| = \\ &= \left| \frac{|A|}{|IP_1|} - \frac{|B|}{|IP_0|} \right| \sim \frac{1}{2^{2n}} \cdot \left| |A| - |B| \right| \leq \frac{(2n+1)p(n) \cdot 2^{3n/2}}{2^{2n}} = \\ &= (2n+1)p(n) \cdot 2^{-n/2} \leq 2^{-\frac{(1-\varepsilon)}{2}n} = 2^{-(1/2-\varepsilon/2)n}\end{aligned}$$

per $n$ sufficientemente grande e per $\varepsilon > 0$. Questo significa che deve essere:

$$\alpha \geq 2^{(1/2-\varepsilon/2)n}$$

Essendo per ipotesi $|\alpha_i| \leq p(n)$ per ogni $i = 1, 2, \ldots, m$, dalla definizione di $\alpha$ segue che $\alpha \leq m \cdot p(n)$; possiamo quindi concludere che:

$$m \geq \frac{\alpha}{p(n)} \geq \frac{1}{p(n)} \cdot 2^{(1/2-\varepsilon/2)n} \geq 2^{(1/2-\varepsilon)n}$$

■

Avendo dimostrato che ogni circuito di profondità 2 e peso polinomiale deve necessariamente avere dimensione esponenziale per poter calcolare la funzione IP, si può affermare, come già detto, che $\mathtt{IP} \notin \widehat{LT}_2$. D'altra parte, per mostrare che $\mathtt{IP} \in \widehat{LT}_3$ è sufficiente calcolare ciascun $x_i \wedge y_i$ con uno strato di $n$ threshold gate, e mandare le uscite di questo strato in ingresso al circuito che calcola $\mathtt{PARITY}_n$ (di profondità 2) per ottenere un circuito che calcola $\mathtt{IP}_n$ in profondità 3 (e peso e dimensione polinomiali). Possiamo pertanto concludere che:

$$\widehat{LT}_2 \subset \widehat{LT}_3$$

## 2.7 Separazioni successive

Tutte le separazioni fra le classi della gerarchia (2.1) per $d \geq 3$ sono, a tutt'oggi, aperte. Il motivo di ciò è fondamentalmente il fatto che, come viene generalmente riconosciuto, i circuiti a soglia di bassa profondità sono dispositivi di calcolo molto potenti. Questa affermazione può essere verificata osservando qui di seguito qualche risultato riguardante alcune funzioni notevoli. Per le dimostrazioni relative alle funzioni aritmetiche, l'articolo di Siu e Roychowdhury [16] costituisce un ottimo punto di partenza.

Diamo anzitutto la seguente definizione.

**Definizione 2.14.** Dato un vettore $\mathbf{x} \in \{0,1\}^n$, indichiamo con $\langle \mathbf{x} \rangle$ il numero intero $\sum_{i=1}^{n} x_i 2^{i-1}$. In pratica, questo significa che $\mathbf{x}$ è la *rappresentazione binaria* di $\langle \mathbf{x} \rangle$, con $x_n$ bit più significativo. In modo improprio, diremo anche che $\langle \mathbf{x} \rangle$ è un numero intero a $n$ bit. ∎

Possiamo ora definire le seguenti funzioni.

**Definizione 2.15.** Dati due interi $\langle \mathbf{x} \rangle$ e $\langle \mathbf{y} \rangle$ a $n$ bit, definiamo la funzione $\texttt{ADDIZIONE}(\mathbf{x}, \mathbf{y})$ a $2n$ ingressi e $n+1$ uscite nel modo seguente:

$$\texttt{ADDIZIONE}(\mathbf{x}, \mathbf{y}) = \mathbf{z} \qquad \text{con } \langle \mathbf{z} \rangle = \langle \mathbf{x} \rangle + \langle \mathbf{y} \rangle$$

Essendo ADDIZIONE una funzione multi-uscita, ottenibile accostando $n+1$ funzioni Booleane, nel seguito ci riferiremo impropriamente ad $\texttt{ADDIZIONE}(\mathbf{x}, \mathbf{y})$ — e a tutte le funzioni qui di seguito definite — come alla funzione che calcola $z_i$, cioè l'$i$-esimo bit di $\langle \mathbf{x} \rangle + \langle \mathbf{y} \rangle$. ∎

**Definizione 2.16.** Dati $n$ interi $\langle \mathbf{x}_1 \rangle, \langle \mathbf{x}_2 \rangle, \ldots, \langle \mathbf{x}_n \rangle$ a $n$ bit, definiamo la funzione $\texttt{ADDIZIONE} - \texttt{MULTIPLA}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ a $n^2$ ingressi e $n + \log n$ uscite nel modo seguente:

$$\texttt{ADDIZIONE} - \texttt{MULTIPLA}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \mathbf{z} \qquad \text{con } \langle \mathbf{z} \rangle = \sum_{i=1}^{n} \langle \mathbf{x}_i \rangle$$

In letteratura, la funzione ADDIZIONE-MULTIPLA è nota anche come *somma iterata*. ∎

In modo del tutto analogo a quanto fatto per le funzioni ADDIZIONE e ADDIZIONE-MULTIPLA è possibile definire le funzioni MOLTIPLICAZIONE e MOLTIPLICAZIONE-MULTIPLA (quest'ultima è nota anche come *prodotto iterato*), dove la prima è una funzione a $2n$ ingressi e $2n$ uscite, mentre la seconda ha $n^2$ ingressi e $n^2$ uscite.

**Definizione 2.17.** Dato un intero $\langle \mathbf{x} \rangle$ a $n$ bit, definiamo la funzione $\texttt{QUADRATO}(\mathbf{x})$ a $n$ ingressi e $2n$ uscite nel modo seguente:

$$\texttt{QUADRATO}(\mathbf{x}) = \mathbf{z} \qquad \text{con } \langle \mathbf{z} \rangle = \langle \mathbf{x} \rangle^2$$

∎

**Definizione 2.18.** Dato un intero $\langle \mathbf{x} \rangle \geq 0$ a $n$ bit, definiamo la funzione $\texttt{ESPONENZIAZIONE}(\mathbf{x})$ a $n$ ingressi e $n^2$ uscite nel modo seguente:

$$\texttt{ESPONENZIAZIONE}(\mathbf{x}) = \mathbf{z} \qquad \text{con } \langle \mathbf{z} \rangle = \langle \mathbf{x} \rangle^n$$

∎

Per quanto riguarda la divisione, si supponga di voler calcolare il quoziente di due numeri interi $\langle \mathbf{x} \rangle$ e $\langle \mathbf{y} \rangle$ a $n$ bit. Ci si rende presto conto che alcuni quozienti hanno una rappresentazione binaria che richiede infinite cifre binarie; avendo per ipotesi ogni circuito a soglia un'unica uscita, dovremmo accostare infiniti circuiti per calcolare il quoziente in maniera esatta. Poiché ciò è inaccettabile dal punto di vista pratico, mentre dal punto di vista teorico inficia i nostri sforzi per trovare circuiti di dimensione polinomiale, ci limiteremo a calcolare i bit più significativi del quoziente. Siamo pertanto interessati al calcolo del quoziente *troncato*, definito come segue.

**Definizione 2.19.** Siano $\langle \mathbf{x} \rangle$ e $\langle \mathbf{y} \rangle$ due interi a $n$ bit, con $\langle \mathbf{y} \rangle \geq 1$. Sia $\langle \mathbf{x} \rangle / \langle \mathbf{y} \rangle = \sum_{i=-\infty}^{n-1} z_i 2^i$ il quoziente di $\langle \mathbf{x} \rangle$ diviso per $\langle \mathbf{y} \rangle$. Definiamo la funzione $\texttt{DIVISIONE}_k(\mathbf{x}, \mathbf{y})$ come il valore di $\langle \mathbf{x} \rangle / \langle \mathbf{y} \rangle$ troncato all'$(n+k)$-esimo bit, cioè:

$$\texttt{DIVISIONE}_k(\mathbf{x}, \mathbf{y}) = \sum_{i=-k}^{n-1} z_i 2^i$$

Come caso particolare, abbiamo che $\texttt{DIVISIONE}_0(\mathbf{x}, \mathbf{y})$ è $\lfloor \langle \mathbf{x} \rangle / \langle \mathbf{y} \rangle \rfloor$, il più grande intero minore o uguale a $\langle \mathbf{x} \rangle / \langle \mathbf{y} \rangle$. ■

Oltre alle funzioni aritmetiche, consideriamo le seguenti funzioni multi-uscita, che sono tra le più studiate nel confronto fra le capacità computazionali dei diversi tipi di circuiti.

**Definizione 2.20.** Dati due interi $\langle \mathbf{x} \rangle$ e $\langle \mathbf{y} \rangle$ a $n$ bit, definiamo la funzione $\texttt{CONFRONTO}(\mathbf{x}, \mathbf{y})$ a $2n$ ingressi e 1 uscita nel modo seguente:

$$\texttt{CONFRONTO}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{se } \langle \mathbf{x} \rangle \geq \langle \mathbf{y} \rangle \\ 0 & \text{se } \langle \mathbf{x} \rangle < \langle \mathbf{y} \rangle \end{cases}$$

■

**Definizione 2.21.** Dati $n$ interi $\langle \mathbf{x}_1 \rangle, \langle \mathbf{x}_2 \rangle, \ldots, \langle \mathbf{x}_n \rangle$ a $n$ bit, definiamo la funzione $\texttt{MASSIMO}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ a $n^2$ ingressi ed $n$ uscite nel modo seguente:

$$\texttt{MASSIMO}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \mathbf{z} \qquad \text{con } \langle \mathbf{z} \rangle = \max_{1 \leq i \leq n} \langle \mathbf{x}_i \rangle$$

■

**Definizione 2.22.** Dati $n$ interi $\langle \mathbf{x}_1 \rangle, \langle \mathbf{x}_2 \rangle, \ldots, \langle \mathbf{x}_n \rangle$ a $n$ bit, definiamo la funzione $\texttt{ORDINAMENTO}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ a $n^2$ ingressi ed $n^2$ uscite nel modo seguente:

$$\texttt{ORDINAMENTO}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = (\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n})$$
$$\text{con } \{i_1, i_2, \ldots, i_n\} = \{1, 2, \ldots, n\} \text{ e } \langle \mathbf{x}_{i_j} \rangle \leq \langle \mathbf{x}_{i_{j+1}} \rangle \quad \forall j = 1, 2, \ldots, n$$

■

La seguente tabella illustra le profondità necessarie (lower bound) e sufficienti (upper bound) per calcolare le funzioni appena definite con circuiti a soglia di dimensione e peso polinomiali. Si noti che gli upper bound indicati sono *uniformi*, nel senso indicato a pagina 30.

| Funzione | Lower Bound | Upper Bound uniforme |
|---|---|---|
| ADDIZIONE | 2 | 2 |
| ADDIZIONE MULTIPLA | 2 | 2 |
| MOLTIPLICAZIONE | 3 | 3 |
| MOLTIPLICAZIONE MULTIPLA | 3 | 4 |
| QUADRATO | 3 | 3 |
| ESPONENZIAZIONE | 2 | 3 |
| DIVISIONE | 3 | 3 |
| CONFRONTO | 2 | 2 |
| MASSIMO | 2 | 3 |
| ORDINAMENTO | 3 | 3 |

Si noti che la tabella si riferisce ai circuiti di peso polinomiale; se accettiamo di utilizzare anche pesi esponenziali, siamo in grado di migliorare il risultato riguardante la funzione CONFRONTO.

**Teorema 2.14.** CONFRONTO $\in LT_1$.

*Dimostrazione.* Dati due numeri $\langle \mathbf{x} \rangle$ e $\langle \mathbf{y} \rangle$ di $n$ bit ciascuno, per definizione si ha che:

$$\begin{aligned}
\texttt{CONFRONTO}(\mathbf{x},\mathbf{y}) = 1 &\iff \langle \mathbf{x} \rangle \geq \langle \mathbf{y} \rangle \\
&\iff \sum_{i=1}^{n} x_i 2^{i-1} \geq \sum_{i=1}^{n} y_i 2^{i-1} \\
&\iff \sum_{i=1}^{n} x_i 2^{i-1} - \sum_{i=1}^{n} y_i 2^{i-1} \geq 0 \\
&\iff \sum_{i=1}^{n} \left[ x_i 2^{i-1} + y_i (-2^{i-1}) \right] \geq 0
\end{aligned}$$

Preso allora un threshold gate $T$ a $2n$ ingressi $x_1, x_2, \ldots, x_n$ e $y_1, y_2, \ldots, y_n$, con pesi rispettivamente uguali a $2^0, 2^1, \ldots, 2^{n-1}$ e a $-2^0, -2^1, \ldots, -2^{n-1}$, e soglia uguale a 0, è immediato verificare che $T$ calcola la funzione CONFRONTO. ∎

# Capitolo 3

# Approssimazione di funzioni Booleane

Nel capitolo precedente abbiamo visto come calcolare alcune funzioni Booleane utilizzando circuiti a soglia di profondità costante, e abbiamo anche visto come un semplice trucco — applicabile ai neuroni che calcolano funzioni simmetriche, oppure funzioni che dipendono dalla quantità $\sum_{i=1}^{n} w_i x_i$ (e che verificano le ipotesi del Teorema 2.9), oppure a funzioni che valgono 1 all'interno di un numero al più polinomiale di intervalli — consenta in alcuni casi di abbassare la profondità del circuito che calcola una data funzione Booleana.

In questo capitolo introduciamo il concetto di *approssimazione di una funzione Booleana*: dopo avere visto cosa significa approssimare una funzione, vedremo alcune proprietà delle approssimazioni che ci consentono di costruire dei circuiti a soglia per calcolare le funzioni approssimate; infine, vedremo come la suddetta tecnica che consente di abbassare la profondità dei circuiti possa essere estesa anche a questo dominio.

Come si avrà modo di appurare in seguito, le definizioni e i teoremi di questo capitolo formano una solida base su cui poggiano le tecniche più sofisticate di analisi delle funzioni Booleane — come le tecniche spettrali, esaminate nel prossimo capitolo — e i tentativi di confronto dei circuiti a soglia con modelli di calcolo più tradizionali.

## 3.1 Definizioni

Si consideri un threshold gate con ingressi $(x_1, x_2, \ldots, x_n) \in \{-1, 1\}^n$; come abbiamo visto nel Capitolo 1, la funzione calcolata da tale threshold gate è:

$$f(x_1, x_2, \ldots, x_n) = sign\left(\sum_{i=1}^{n} w_i x_i - w_0\right) = \begin{cases} +1 & \text{se } \sum_{i=1}^{n} w_i x_i - w_0 \geq 0 \\ -1 & \text{se } \sum_{i=1}^{n} w_i x_i - w_0 < 0 \end{cases}$$

Poiché la funzione di trasferimento del threshold gate prende in considerazione solo il segno della combinazione lineare degli ingressi $\sum_{i=1}^{n} w_i x_i - w_0$, senza

considerarne il valore, si potrà in generale ottenere la funzione $f$ in più di un modo, con diversi valori dei pesi e della soglia. In particolare, se $w'_0, w'_1, w'_2, \ldots, w'_n$ sono numeri interi o razionali per i quali, al variare di $(x_1, x_2, \ldots, x_n)$ in $\{-1, 1\}^n$ si ha:

$$\left| f(x_1, x_2, \ldots, x_n) - \left( \sum_{i=1}^{n} w'_i x_i - w'_0 \right) \right| < 1$$

allora il segno di $f(x_1, x_2, \ldots, x_n)$ e della combinazione lineare indicata è lo stesso per ogni configurazione delle variabili di ingresso, e quindi:

$$f(x_1, x_2, \ldots, x_n) = sign\left( \sum_{i=1}^{n} w_i x_i - w_0 \right) = sign\left( \sum_{i=1}^{n} w'_i x_i - w'_0 \right)$$

Naturalmente, il discorso appena fatto non ha alcuna utilità pratica se applicato alle funzioni calcolate dai singoli threshold gate, ma la situazione cambia se lo applichiamo al threshold gate d'uscita di un circuito a soglia, facendo variare, oltre ai pesi, anche gli *ingressi* (ovvero le funzioni calcolate dai sottocircuiti) della combinazione lineare. Consideriamo quindi un circuito $C_n$ a $n$ ingressi, di dimensione e peso polinomiale e di profondità $d$; chiamiamo $T$ il threshold gate d'uscita, e indichiamo con $f$ la funzione Booleana calcolata dal circuito. Possiamo allora scrivere:

$$f(\mathbf{x}) = sign\left( \sum_{j=1}^{N} \alpha_j f_j(\mathbf{x}) \right)$$

con $\mathbf{x} \in \{-1, 1\}^n$, dove $f_1, f_2, \ldots, f_N$ sono le funzioni calcolate dai sottocircuiti le cui uscite vanno in ingresso a $T$, $N$ è il numero (al più polinomiale rispetto ad $n$) di tali sottocircuiti, e $\alpha_1, \alpha_2, \ldots, \alpha_N$ sono i pesi del threshold gate $T$. Analogamente a quanto avviene nel caso di un neurone singolo, se esistono un intero $M$ limitato da un polinomio, dei numeri interi $\beta_1, \beta_2, \ldots, \beta_M$ al più polinomiali e delle funzioni $g_1, g_2, \ldots, g_M \in \Re^{\{-1,1\}^n}$ tali che:

$$\left| f(\mathbf{x}) - \sum_{i=1}^{M} \beta_i g_i(\mathbf{x}) \right| < 1 \qquad \forall\, \mathbf{x} \in \{-1, 1\}^n \tag{3.1}$$

allora possiamo scrivere che:

$$f(\mathbf{x}) = sign\left( \sum_{j=1}^{N} \alpha_j f_j(\mathbf{x}) \right) = sign\left( \sum_{i=1}^{M} \beta_i g_i(\mathbf{x}) \right)$$

Si osservi che, nell'equazione (3.1), possiamo sostituire la costante maggiorante 1 con un parametro $\varepsilon$ minore di 1, ottenendo così il fulcro sul quale poggia la seguente definizione.

**Definizione 3.1.** Sia $\varepsilon < 1$ un valore reale fissato. Diciamo che una funzione Booleana $f \in BIN_n$, definita nel dominio $\{-1, 1\}$, è *$\varepsilon$-approssimata* se esistono degli interi $\alpha_1, \alpha_2, \ldots, \alpha_N$ al più polinomiali e delle funzioni $t_1, t_2, \ldots, t_N \in \Re^{\{-1,1\}^n}$ tali che:

$$\left| f(\mathbf{x}) - \sum_{i=1}^{N} \alpha_i t_i(\mathbf{x}) \right| < \varepsilon \qquad \forall\, \mathbf{x} \in \{-1, 1\}^n$$

con $N$ intero al più polinomiale. ∎

Si noti che abbiamo finora supposto che le funzioni Booleane considerate siano definite sul dominio $\{-1, 1\}$. Ciò è stato fatto esclusivamente per una questione di comodità di notazione: le definizioni fin qui enunciate possono essere facilmente adattate a funzioni Booleane definite sul dominio $\{0, 1\}$[1].

Restando nel dominio $\{-1, 1\}$, possiamo estendere la nozione di $\varepsilon$-approssimazione richiedendo che il valore di $\varepsilon$ diminuisca al crescere di $n$, rendendo così l'approssimazione sempre più precisa al tendere di $n$ all'infinito. Questo significa che, al posto di un valore fissato per $\varepsilon$, considereremo una *successione* $\{\varepsilon_n\}_{n \in \boldsymbol{N}}$ di valori reali; per ogni funzione $f_n(x_1, x_2, \ldots, x_n)$ di una data famiglia $\boldsymbol{\mathcal{F}}$, costruiremo pertanto una $\varepsilon_n$-approssimazione nel senso della Definizione 3.1. Poiché la condizione che $\varepsilon_n$ tenda a zero come una funzione esponenziale (ad esempio $2^{-n}$) sembra troppo restrittiva, ci limitiamo a chiedere che $\varepsilon_n$ tenda a zero più velocemente di qualsiasi polinomio prefissato. Per formalizzare quanto appena detto, diamo le seguenti definizioni.

**Definizione 3.2.** Una *funzione multi-uscita* $f(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\big)$ è una collezione di funzioni Booleane $f_1, f_2, \ldots, f_m \in BIN_n$.

Per comodità, scriveremo impropriamente che $f \in BIN_n$, intendendo con questo che ciascuna delle sue componenti $f_1, f_2, \ldots, f_m$ appartiene a $BIN_n$.

Se vale inoltre $m \leq p(n)$ per un opportuno polinomio $p(n)$, diremo che $f$ è una funzione multi-uscita *polinomiale*. ∎

**Definizione 3.3.** Sia $\mathcal{G}$ un insieme di funzioni multi-uscita. Una funzione $f \in \Re^n$ viene detta $\mathcal{G}$-approssimabile se per ogni $k > 0$ esiste una funzione multi-uscita polinomiale $(g_1, g_2, \ldots, g_{p(n)}) \in \mathcal{G}$ tale che:

$$f(\mathbf{x}) = \sum_{j=1}^{p(n)} a_j g_j(\mathbf{x}) \pm O(n^{-k})$$

dove $a_1, a_2, \ldots, a_{p(n)}$ sono numeri reali tali che $\max\limits_{1 \leq j \leq p(n)} |a_j|$ è limitato da un polinomio $q(n)$. Si noti che $p(n)$ e $q(n)$ potranno dipendere in generale anche da $k$. ∎

[1] Ad esempio, nella Definizione 3.1 è sufficiente imporre $\varepsilon < \frac{1}{2}$.

Otteniamo il caso più importante di approssimabilità quando scegliamo $\mathcal{G} = \widehat{LT}_d$; investighiamo allora la $\widehat{LT}_d$-approssimabilità che, per convenienza di notazione, indichiamo semplicemente con *d-approssimabilità*. Dato che i circuiti a soglia fin qui considerati possono calcolare solo funzioni Booleane, consideriamo l'importante sottoclasse di funzioni $d$-approssimabili definita qui di seguito.

**Definizione 3.4.** Indichiamo con $APP_d$ la classe di tutte le funzioni *Booleane* $d$-approssimabili. Poniamo inoltre:

$$APP = \bigcup_{d=1}^{+\infty} APP_d$$

■

## 3.2 Alcune proprietà della classe $APP_d$

Notiamo anzitutto che possiamo interpretare la Definizione 3.3 come segue: ogniqualvolta una funzione Booleana $f$ è $d$-approssimabile, esistono un numero $p(n)$ al più polinomiale di circuiti di profondità $d$ e dimensione e peso polinomiali tali che $f$ può essere ben approssimata da una combinazione lineare delle funzioni da essi calcolate. Alla luce di questa interpretazione, il risultato espresso dal prossimo teorema diventa quasi una conseguenza immediata della Definizione 3.3.

**Teorema 3.1.** *Per ogni intero $d \geq 1$, $APP_d \subseteq \widehat{LT}_{d+1}$.*

*Dimostrazione.* Se $f$ è una funzione Booleana $d$-approssimabile, per ogni $k > 0$ esistono una funzione multi-uscita $(g_1, g_2, \ldots, g_N) \in \widehat{LT}_d$ e $N$ numeri reali $a_1, a_2, \ldots, a_N$ tali che:

$$f(\mathbf{x}) = \sum_{i=1}^{N} a_i g_i(\mathbf{x}) \pm O(n^{-k}) \tag{3.2}$$

con $N$ e $\max\limits_{1 \leq i \leq N} |a_i|$ polinomiali.

Possiamo sostituire i coefficienti reali $a_i$ con numeri razionali nel modo seguente: scegliamo come denominatore comune $N \cdot n^k$, e approssimiamo ciascun $a_i$ con un numero del tipo $\frac{z_i}{N \cdot n^k}$, con $z_i \in \boldsymbol{Z}$. Dato che ciò può essere sempre fatto in modo tale che il valore assoluto dell'errore commesso sia limitato da $\frac{1}{N \cdot n^k}$ per ogni $a_i$, la sostituzione di tutti gli $a_i$ con le approssimazioni razionali nella sommatoria $\sum_{i=1}^{N} a_i g_i(\mathbf{x})$ porta a commettere un errore che è al più $O(n^{-k})$, che viene quindi assorbito nel termine $O(n^{-k})$ già presente. Otteniamo allora la seguente approssimazione di $f$, perfettamente legittima secondo la Definizione 3.3:

$$f(\mathbf{x}) = \frac{1}{N \cdot n^k} \cdot \sum_{i=1}^{N} z_i g_i(\mathbf{x}) \pm O(n^{-k})$$

nella quale tutti i coefficienti della combinazione lineare sono razionali, e dove $N \cdot n^k$ e tutti i $|z_i|$ sono polinomiali. Come si è già osservato a proposito dell'equazione (3.1), tale approssimazione ci consente di affermare che, per $n$ sufficientemente grande, vale:

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N} z_i g_i(\mathbf{x})\right) \qquad \forall\, \mathbf{x} \in \{-1,1\}^n$$

Questa rappresentazione di $f$ ci consente di costruire immediatamente un circuito di profondità $d+1$ — e dimensione e peso polinomiali — che la calcola, a partire dagli $N$ circuiti di profondità $d$ che calcolano le funzioni $g_1, g_2, \ldots, g_N$; pertanto, si può concludere che $f \in \widehat{LT}_{d+1}$. ∎

Si noti che nella Definizione 3.3 non viene richiesto che i coefficienti $a_1, a_2, \ldots, a_{p(n)}$ della combinazione lineare utilizzata per approssimare $f$ siano numeri razionali. La ragione per cui ciò non è necessario diventa chiara nel corso della dimostrazione del Teorema 3.1, in cui viene mostrato un metodo per sostituire i coefficienti con delle approssimazioni razionali. Un metodo alternativo per liberarsi dei coefficienti reali, sostituendoli con valori interi, è quello di costruire il circuito per calcolare $f$ a partire direttamente dalla rappresentazione (3.2), e di applicare successivamente il Teorema 1.1 al neurone d'uscita. Questa osservazione sembrerebbe banalizzare la dimostrazione del Teorema 3.1, rendendola praticamente inutile; in realtà, il principale pregio della dimostrazione è quello di mostrare che la sostituzione dei coefficienti $a_1, a_2, \ldots, a_N$ con valori interi può essere fatta in modo tale che l'approssimazione di $f$ risultante verifichi ancora la Definizione 3.3.

Uno dei motivi per cui le funzioni Booleane $d$-approssimabili sono interessanti è dato dal seguente teorema, che mostra come la tecnica illustrata nella Figura 2.2 per abbassare la profondità di alcuni circuiti possa essere estesa anche al caso in cui la funzione da calcolare dipenda dalla somma di un numero al più polinomiale di funzioni $d$-approssimabili.

**Teorema 3.2.** *Sia $(t_1, t_2, \ldots, t_{r(n)})$ una funzione multi-uscita polinomiale e $d$-approssimabile. Se una funzione Booleana $f$ può essere calcolata come:*

$$f(\mathbf{x}) = sign\big(t_1(\mathbf{x}) + t_2(\mathbf{x}) + \ldots + t_{r(n)}(\mathbf{x})\big)$$

*allora $f \in \widehat{LT}_{d+1}$.*

*Dimostrazione.* Essendo $r(n)$ un polinomio, esiste un intero $k^*$ tale che $r(n) = O(n^{k^*})$. Per ogni $i = 1, 2, \ldots, r(n)$, la funzione $t_i$ è $d$-approssimabile: esistono pertanto, in corrispondenza di $k^*$, $r(n)$ funzioni multi-uscita polinomiali $(g_{i,1}, g_{i,2}, \ldots, g_{i,q(n)}) \in \widehat{LT}_d$ ed $r(n) \cdot q(n)$ numeri polinomiali $a_{i,1}, a_{i,2}, \ldots, a_{i,q(n)}$

— che per il Teorema 3.1 possiamo supporre tutti *razionali* con lo stesso denominatore — tali che:

$$t_i(\mathbf{x}) = \sum_{j=1}^{q(n)} a_{i,j} g_{i,j}(\mathbf{x}) + \varepsilon_i(\mathbf{x})$$

con $|\varepsilon_i(\mathbf{x})| = O(n^{-k^*})$ per ogni $\mathbf{x} \in \{-1,1\}^n$. Se poniamo:

$$app_i(\mathbf{x}) = \sum_{j=1}^{q(n)} a_{i,j} g_{i,j}(\mathbf{x})$$

abbiamo che vale:

$$t_1(\mathbf{x}) + \ldots + t_{r(n)}(\mathbf{x}) - \big[app_1(\mathbf{x}) + \ldots + app_{r(n)}(\mathbf{x})\big] = \varepsilon(\mathbf{x})$$

con:

$$|\varepsilon(\mathbf{x})| = \left| \sum_{i=1}^{r(n)} \varepsilon_i(\mathbf{x}) \right| = o(1) \qquad \forall\, \mathbf{x} \in \{-1,1\}^n$$

Quindi, per $n$ sufficientemente grande, possiamo esprimere $f$ nel seguente modo:

$$\begin{aligned} f(\mathbf{x}) &= sign\big(t_1(\mathbf{x}) + t_2(\mathbf{x}) + \ldots + t_{r(n)}(\mathbf{x})\big) = \\ &= sign\big(app_1(\mathbf{x}) + app_2(\mathbf{x}) + \ldots + app_{r(n)}(\mathbf{x})\big) \end{aligned}$$

A questo punto, si può notare che l'argomento della funzione *sign* può essere scritto come segue:

$$\begin{aligned} app_1(\mathbf{x}) + app_2(\mathbf{x}) + \ldots + app_{r(n)}(\mathbf{x}) = \\ = \sum_{i=1}^{r(n)} app_i(\mathbf{x}) = \sum_{i=1}^{r(n)} \sum_{j=1}^{q(n)} a_{i,j} g_{i,j}(\mathbf{x}) = \sum_{j=1}^{q(n)} a_j^* g_j(\mathbf{x}) \end{aligned} \tag{3.3}$$

dove l'ultima uguaglianza deriva dal fatto che una combinazione lineare (e quindi anche una somma) di combinazioni lineari è essa stessa una combinazione lineare; i coefficienti $a_j^*$ di quest'ultima sono numeri razionali (polinomiali) con un denominatore comune. Il segno delle quantità indicate nell'equazione 3.3 non cambia se le moltiplichiamo per tale denominatore, in modo da ottenere dei coefficienti *interi* per la combinazione lineare.

Abbiamo allora espresso $f$ come il segno della combinazione lineare — a coefficienti polinomiali — di un numero al più polinomiale di funzioni appartenenti a $\widehat{LT}_d$: possiamo pertanto concludere che $f \in \widehat{LT}_{d+1}$. ∎

È immediato riconoscere che il teorema appena dimostrato può essere esteso al caso in cui $f$, anziché essere esprimibile come il segno della somma di un numero polinomiale di funzioni $d$-approssimabili, sia rappresentabile come il segno di una

combinazione lineare a pesi polinomiali di un numero polinomiale di funzioni $d$-approssimabili:

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{r(n)} \alpha_i t_i(\mathbf{x})\right)$$

È infatti sufficiente riscrivere per esteso la combinazione lineare per ricadere nelle ipotesi del Teorema 3.2.

Si noti che, nell'enunciato del Teorema 3.2, non viene richiesto che $(t_1, t_2, \ldots, t_{r(n)}) \in APP_d$, bensì viene richiesto semplicemente che $(t_1, t_2, \ldots, t_{r(n)})$ sia $d$-approssimabile; si noti inoltre che, nella dimostrazione, il circuito cui si perviene per calcolare $f$ ha profondità pari a $d+1$, laddove il circuito "banale" ottenibile a partire dal Teorema 3.1 e dal fatto che le funzioni $t_1, t_2, \ldots, t_{r(n)}$ sono $d$-approssimabili ha profondità $d+2$. La tecnica che consente di abbasssare di 1 la profondità del circuito è quella indicata nell'equazione (3.3), nella quale riscriviamo una somma di combinazioni lineari sotto forma di semplice combinazione lineare; come è già stato detto, questa tecnica costituisce in un certo senso un'estensione di quella introdotta dai Teoremi 2.7, 2.9 e 2.10 — e illustrata schematicamente nella Figura 2.2 — al caso in cui la funzione dipenda da una combinazione lineare di funzioni $d$-approssimabili.

Abbiamo parlato finora di funzioni $d$-approssimabili senza preoccuparci di vedere se tra di esse esistono delle funzioni utili. Nel prossimo capitolo vedremo che il Teorema 4.6, il risultato fondamentale su cui si basa la tecnica di analisi spettrale di una funzione Booleana, può essere parafrasato dicendo che ogni funzione Booleana che abbia $L_1$-norma polinomiale è 1-approssimabile, ed appartiene quindi alla classe $\widehat{LT}_2$.

Un'osservazione banale — ma non priva di risvolti interessanti — è poi quella che ogni funzione $f \in \widehat{LT}_d$ è $d$-approssimabile: l'approssimazione (in questo caso esatta) è costituita da $f$ stessa, con coefficiente pari a 1. Questo significa che il risultato del Teorema 3.1 non è sempre ottimale, in quanto può capitare che una funzione $f \in APP_d$ appartenga alla classe $\widehat{LT}_d$.

Possiamo unire l'osservazione precedente con l'enunciato del Teorema 3.1: vediamo allora che le classi $\widehat{LT}_d$ e $APP_d$ formano una gerarchia che ricorda molto da vicino quella dell'equazione (2.1):

$$\widehat{LT}_d \subseteq APP_d \subseteq \widehat{LT}_{d+1} \qquad \forall d \geq 1 \tag{3.4}$$

Analogamente a quanto è stato detto riguardo la gerarchia (2.1), il seguente teorema è conseguenza immediata della gerarchia (3.4).

**Teorema 3.3.** $APP = TC^0$.

*Dimostrazione.* La dimostrazione è analoga a quella del Teorema 2.6. ∎

Tenendo conto del teorema precedente, è quasi superfluo dire che la separazione delle classi coinvolte nella gerarchia (3.4) è strettamente correlata alla separazione delle classi della gerarchia (2.1), ed è altrettanto difficile.

Altre funzioni $d$-approssimabili sono ottenibili dal seguente teorema, che molto spesso si rivela utile nella costruzione dei circuiti.

**Teorema 3.4.** *$APP_{APP_d} \subseteq APP_d$ (ovvero ogni funzione Booleana $APP_d$-approssimabile è anche $d$-approssimabile).*

*Dimostrazione* (Schema). Sia $f \in APP_{APP_d}$; poiché $f$ è $APP_d$-approssimabile, per ogni $k > 0$ esistono una funzione multi-uscita polinomiale $(g_1, g_2, \ldots, g_{p(n)}) \in APP_d$ e dei numeri reali polinomiali $a_1, a_2, \ldots, a_{p(n)}$ tali che:

$$f(\mathbf{x}) = \sum_{j=1}^{p(n)} a_j g_j(\mathbf{x}) \pm O(n^{-k})$$

Per ogni $j = 1, 2, \ldots, p(n)$, la funzione $g_j$ è $d$-approssimabile: procedendo come nel Teorema 3.2, possiamo quindi approssimare ciascuna funzione $g_j$ con una combinazione lineare — a coefficienti polinomiali — di un numero al più polinomiale di funzioni che appartengono a $\widehat{LT}_d$. Il valore assoluto dell'errore commesso considerando tali approssimazioni è al più $O(n^{-k})$, come si può vedere nella dimostrazione del Teorema 3.2. Riscrivendo la combinazione lineare di combinazioni lineari sotto forma di semplice combinazione lineare, otteniamo un'approssimazione di $f$ costituita da una combinazione lineare — a coefficienti polinomiali — di un numero al più polinomiale di funzioni appartenenti a $\widehat{LT}_d$, quindi $f$ è $d$-approssimabile. ∎

Un esempio molto particolare di applicazione del Teorema 3.4 è costituito dal prossimo teorema. Si osservi che, in questo caso, risulta conveniente lavorare con ingressi e uscite a valori in $\{0, 1\}$.

**Teorema 3.5.** *Sia $(g_1, g_2, \ldots, g_{r(n)})$ una funzione multi-uscita $d$-approssimabile. Se $f \in BIN_n$ è una funzione Booleana tale che:*

$$f(\mathbf{x}) = \bigvee_{i=1}^{r(n)} g_i(\mathbf{x})$$

*ed inoltre, per ogni $\mathbf{x} \in \{0,1\}^n$, esiste al più un indice $i \in \{1, 2, \ldots, r(n)\}$ tale che $g_i(\mathbf{x}) = 1$, allora anche $f$ è $d$-approssimabile (e appartiene quindi ad $APP_d$).*

*Dimostrazione.* È sufficiente considerare la seguente approssimazione di $f$ (che, in questo caso, è una approssimazione *esatta*):

$$f(\mathbf{x}) = \sum_{i=1}^{r(n)} g_i(\mathbf{x})$$

e applicare il Teorema 3.4. ∎

Si noti che il teorema precedente *non* afferma che la disgiunzione logica di funzioni Booleane $d$-approssimabili è *sempre* $d$-approssimabile: il caso qui considerato è molto particolare in quanto, per ogni possibile configurazione di ingresso, *al più una* delle funzioni della disgiunzione logica assume valore 1.

## 3.3 Chiusura delle classi $APP_d$ rispetto alle funzioni di grado costante

Quando si progettano circuiti a soglia di profondità costante, può capitare che si uniscano le uscite di un numero costante di circuiti che risolvono dei sottoproblemi, ottenendo un circuito in cui uno strato di neuroni ha fan-in costante. In questo paragrafo vediamo come si possa anche in questo caso abbassare di 1 la profondità del circuito ottenuto, utilizzando un altro semplice trucco. Vedremo inoltre che alcune importanti proprietà dei sottocircuiti vengono preservate, sollevando quindi la questione della conservazione delle medesime proprietà quando il fan-in non è più costante ma è, ad esempio, polinomiale.

Cominciamo allora col definire cosa si intende per chiusura rispetto alle funzioni di grado costante.

**Definizione 3.5.** Un insieme $\mathcal{G}$ di funzioni multi-uscita viene detto *chiuso rispetto alle funzioni di grado costante* se, per ogni intero $c \geq 1$ fissato, ogni funzione multi-uscita $(g_1, g_2, \ldots, g_c) \in \mathcal{G}$ e ogni funzione Booleana $h \in BIN_c$ si ha che la funzione Booleana:

$$f(\mathbf{x}) = h\big(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_c(\mathbf{x})\big)$$

appartiene a $\mathcal{G}$. ■

La locuzione *funzione di grado costante* — riferita alla funzione $h \in BIN_c$ della definizione precedente — è dovuta al fatto che, come si vedrà nel prossimo capitolo, $h$ può essere rappresentata tramite un polinomio formato da un numero costante di monomi, ciascuno di grado costante.

Sorge allora spontanea la domanda: esiste almeno una classe $\mathcal{G}$ di funzioni multi-uscita che sia chiusa rispetto alle funzioni Booleane di grado costante? La risposta è naturalmente positiva, e la classe che prendiamo in considerazione per dimostrarlo è la seguente.

**Definizione 3.6.** Indichiamo con $SYMM_n$ l'insieme di tutte le funzioni multi-uscita polinomiali $(g_1, g_2, \ldots, g_{p(n)})$ a $n$ variabili che sono calcolabili da un circuito di profondità 1 formato da $p(n)$ gate simmetrici. Poniamo inoltre:

$$SYMM = \bigcup_{n=1}^{+\infty} SYMM_n$$

■

Per dimostrare il prossimo teorema, è conveniente lavorare con ingressi e uscite nel dominio $\{0,1\}$.

**Teorema 3.6.** *$SYMM$ è chiuso rispetto alle funzioni di grado costante.*

*Dimostrazione.* Fissato un intero $c \geq 1$, si considerino una funzione multi-uscita $(g_1, g_2, \ldots, g_c) \in SYMM$ e una funzione Booleana $h \in BIN_c$. Senza perdere in generalità, si può supporre che $g_1, g_2, \ldots, g_c \in SYMM_n$; sia inoltre:

$$f(\mathbf{x}) = h\big(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_c(\mathbf{x})\big)$$

Il circuito che calcola $f$ è allora costituito da $c$ gate simmetrici a $n$ ingressi, le cui uscite vanno in ingresso a un dispositivo a $c$ ingressi che calcola $h$. Si osservi che non ci interessa assolutamente come è fatto il dispositivo che calcola la funzione $h \in BIN_c$; dato che il nostro scopo è quello di eliminare tale dispositivo, e dato che siamo in grado di farlo senza considerare il suo funzionamento interno, possiamo pensare ad esso come ad una scatola nera dotata di $c$ ingressi e un'uscita.

Mostriamo come, con l'aiuto di una semplice codifica, possiamo esprimere la funzione $f$ come una funzione simmetrica. A tale scopo, chiamiamo $G_1, G_2, \ldots, G_c$ i gate simmetrici del circuito, e indichiamo con $W_i$ l'insieme dei collegamenti di ingresso del gate $G_i$. Sia inoltre:

$$N = \max_{1 \leq i \leq c} |W_i|$$

il fan-in massimo tra tutti i gate simmetrici del circuito. Si noti che, essendo ogni $|W_i|$ polinomiale, anche $N$ sarà al più polinomiale.

Per ogni gate $G_i$, si prendano $(N+1)^{i-1}$ copie di ciascuno dei suoi collegamenti di ingresso in $W_i$: avremo così 1 copia dei collegamenti di ingresso di $G_1$, $N+1$ copie dei collegamenti di $G_2$, $(N+1)^2$ copie dei collegamenti di $G_3$, ..., $(N+1)^{c-1}$ copie dei collegamenti di $G_c$; si noti che ciò corrisponde ad una codifica $(N+1)$-aria degli ingressi del circuito. Indichiamo con $W$ l'insieme risultante di collegamenti.

A questo punto si osservi che se conosciamo il numero $w$ di collegamenti dell'insieme $W$ che trasportano un 1 per una determinata configurazione d'ingresso $\mathbf{x}$, possiamo facilmente ricostruire il numero di collegamenti di ciascun insieme $W_i$ che trasportano un 1 in corrispondenza di $\mathbf{x}$; basta esprimere il numero intero $w$ in base $N+1$:

$$w = w_c \cdot (N+1)^{c-1} + \ldots + w_2 \cdot (N+1) + w_1$$

per ottenere, per ogni $i = 1, 2, \ldots, c$, il numero $w_i$ di collegamenti di $W_i$ richiesti. Questo significa che le uscite di tutti i gate $G_i$ — e pertanto anche il valore di $f$ — sono univocamente determinati da $w$, così che $f$ può essere vista come una funzione simmetrica in $W$.

Resta solamente da stimare $|W|$. A questo proposito abbiamo:

$$|W| = \sum_{i=1}^{c} |W_i| \cdot (N+1)^{i-1} \leq \sum_{i=1}^{c} N \cdot (N+1)^{i-1} = (N+1)^c - 1$$

Dato che $N$ è al più polinomiale e $c$ è costante, anche $|W|$ sarà al più polinomiale. Questo significa che $f$ può essere calcolata da un circuito di profondità 1 formato da un numero al più polinomiale di gate simmetrici, ovvero $f \in SYMM$. ∎

Trucchi analoghi a quello utilizzato per dimostrare il precedente teorema, consistenti in particolari codifiche, possono essere trovati anche per classi di funzioni diverse da $SYMM$.

Esponiamo ora il principale risultato di questo paragrafo; poiché la dimostrazione fa uso della rappresentazione polinomiale della funzione $h \in BIN_c$, si raccomanda la lettura del primo paragrafo del prossimo capitolo, dedicato alla trasformata di Fourier per le funzioni Booleane. Si osservi che, per dimostrare il prossimo teorema, risulta conveniente supporre che le funzioni Booleane siano definite nel dominio $\{1, -1\}$.

**Teorema 3.7.** *Per ogni intero $d \geq 1$, $APP_d$ è chiusa rispetto alle funzioni di grado costante.*

*Dimostrazione.* Siano date una funzione multi-uscita $(t_1, t_2, \ldots, t_c) \in APP_d$ e una funzione Booleana $h \in BIN_c$. Vogliamo dimostrare che la funzione Booleana:

$$f(\mathbf{x}) = h\big(t_1(\mathbf{x}), t_2(\mathbf{x}), \ldots, t_c(\mathbf{x})\big)$$

è $d$-approssimabile, ovvero che appartiene alla classe $APP_d$.

Per ogni $i = 1, 2, \ldots, c$ la funzione $t_i$ è $d$-approssimabile: esistono pertanto, per ogni $k > 0$, delle funzioni multi-uscita polinomiali $(g_{i,1}, g_{i,2}, \ldots, g_{i,p(n)}) \in \widehat{LT}_d$ e dei numeri reali polinomiali $a_{i,1}, a_{i,2}, \ldots, a_{i,p(n)}$ tali che:

$$t_i(\mathbf{x}) = \sum_{j=1}^{p(n)} a_{i,j} g_{i,j}(\mathbf{x}) + \varepsilon_i(\mathbf{x}) \tag{3.5}$$

con $|\varepsilon_i(\mathbf{x})| = O(n^{-k})$ per ogni $\mathbf{x} \in \{1, -1\}^n$.

Consideriamo ora la rappresentazione polinomiale di $h$: essendo $h$ una funzione a $c$ ingressi, la lunghezza di ciascun monomio è al massimo $c$. Pertanto, possiamo scrivere:

$$f(\mathbf{x}) = \sum_{S \subseteq \{1,\ldots,c\}} a_S \cdot Y_S(\mathbf{x})$$

con:

$$Y_S(\mathbf{x}) = \prod_{i \in S} t_i(\mathbf{x}).$$

Vediamo come si può approssimare $Y_S$. Per semplicità, ci limitiamo a considerare solo il caso in cui $|S| = 2$, dato che gli altri casi sono simili. Abbiamo:

$$t_1(\mathbf{x}) \cdot t_2(\mathbf{x}) = \left( \sum_{i=1}^{p(n)} a_{1,i} g_{1,i}(\mathbf{x}) + \varepsilon_1(\mathbf{x}) \right) \cdot \left( \sum_{j=1}^{p(n)} a_{2,j} g_{2,j}(\mathbf{x}) + \varepsilon_2(\mathbf{x}) \right)$$

Se scegliamo la seguente quantità come approssimazione per $t_1(\mathbf{x}) \cdot t_2(\mathbf{x})$:

$$\sum_{i=1}^{p(n)} \sum_{j=1}^{p(n)} a_{1,i} a_{2,j} g_{1,i}(\mathbf{x}) g_{2,j}(\mathbf{x}) \tag{3.6}$$

troviamo che (per $n$ sufficientemente grande) il valore assoluto dell'errore commesso è maggiorato da:

$$2 \cdot |\varepsilon_1(\mathbf{x})| + 2 \cdot |\varepsilon_2(\mathbf{x})| + |\varepsilon_1(\mathbf{x}) \cdot \varepsilon_2(\mathbf{x})|$$

Infatti, come si può vedere dall'equazione (3.5), il valore della combinazione lineare $\sum_j a_{i,j} g_{i,j}(\mathbf{x})$ non si distacca mai da quello di $t_i(\mathbf{x})$ per più di un'unità; dato che $t_i(\mathbf{x})$ assume solo i valori 1 e $-1$, questo significa che:

$$\left| \sum_{j=1}^{p(n)} a_{i,j} g_{i,j}(\mathbf{x}) \right| < 2 \qquad \forall \mathbf{x} \in \{1, -1\}^n$$

da cui segue la citata maggiorazione.

Si osservi poi che la quantità $2 \cdot |\varepsilon_1(\mathbf{x})| + 2 \cdot |\varepsilon_2(\mathbf{x})| + |\varepsilon_1(\mathbf{x}) \cdot \varepsilon_2(\mathbf{x})|$ è a sua volta $O(n^{-k})$; questo significa che $t_1(\mathbf{x}) \cdot t_2(\mathbf{x})$ è approssimata dalla funzione multi-uscita polinomiale $\{g_{1,i} \cdot g_{2,j} \mid 1 \leq i, j \leq p(n)\}$ nel senso della Definizione 3.3.

Sebbene tale funzione multi-uscita non appartenga necessariamente a $\widehat{LT}_d$, siamo tuttavia in grado di dimostrare che è $d$-approssimabile. Infatti, poiché ogni funzione $g_{1,i}(\mathbf{x}) \cdot g_{2,j}(\mathbf{x})$ consiste nel prodotto di un numero *costante* di funzioni, possiamo applicare il trucco utilizzato nella dimostrazione del Teorema 3.6 per mostrare che può essere computata da un gate simmetrico con pesi polinomiali i cui ingressi sono le uscite di circuiti di profondità $d - 1$. Di conseguenza, ciascuna funzione $g_{1,i}(\mathbf{x}) \cdot g_{2,j}(\mathbf{x})$ può essere vista come una combinazione lineare (con coefficienti polinomiali) di funzioni appartenenti a $\widehat{LT}_d$; inserendo tali combinazioni lineari nell'equazione (3.6), troviamo che $t_1(\mathbf{x}) \cdot t_2(\mathbf{x})$ è $d$-approssimabile. Per il Teorema 3.4, questo significa che anche $Y_S(\mathbf{x})$ è $d$-approssimabile.

Poiché $f$ è la combinazione lineare (a coefficienti costanti) di un numero costante di funzioni $Y_S(\mathbf{x})$, possiamo concludere che anche $f$ è $d$-approssimabile. ∎

Una conseguenza immediata del teorema appena dimostrato è data dal seguente teorema.

**Teorema 3.8.** *Per ogni funzione $h \in BIN_c$ di grado costante, e per ogni funzione multi-uscita $(t_1, t_2, \ldots, t_c) \in \widehat{LT}_d$, la funzione Booleana:*

$$f(\mathbf{x}) = h\big(t_1(\mathbf{x}), t_2(\mathbf{x}), \ldots, t_c(\mathbf{x})\big)$$

*è $d$-approssimabile.*

*Dimostrazione.* Per ogni $i = 1, 2, \ldots, c$ abbiamo che $t_i \in \widehat{LT}_d$, quindi $t_i$ è $d$-approssimabile. Per il Teorema 3.7, anche $f$ è $d$-approssimabile. ∎

A questo punto, viene spontaneo chiedersi se il risultato espresso dal Teorema 3.7 possa essere esteso al caso in cui il fan-in della funzione $h$ sia polinomiale anziché costante. Esaminando la dimostrazione del teorema risulta subito evidente che un'estensione diretta non è possibile: infatti, se $|S|$ fosse polinomiale, l'approssimazione delle funzioni $Y_S(\mathbf{x})$ implicherebbe la dimostrazione del fatto che funzioni del tipo:

$$g_1(\mathbf{x}) \cdot g_2(\mathbf{x}) \cdot \ldots \cdot g_{p(n)}(\mathbf{x})$$

per un opportuno polinomio $p(n)$, e con $g_i \in \widehat{LT}_d$ per ogni $i = 1, 2, \ldots, p(n)$, sono $d$-approssimabili. Mentre ciò è sicuramente possibile in presenza di un numero costante di fattori — applicando il trucco di codifica utilizzato nel Teorema 3.6 — con un numero polinomiale di fattori il suddetto trucco non è più applicabile, poiché si perviene a circuiti di dimensione esponenziale.

D'altra parte, se esistesse un intero $d^* \geq 1$ per il quale la classe $APP_{d^*}$ è chiusa rispetto alle funzioni Booleane di grado polinomiale, allora l'intera classe $TC^0$ collasserebbe sulla classe $APP_{d^*}$, come viene mostrato dai seguenti teoremi.

**Teorema 3.9.** *Sia $d^* \geq 1$ un intero per cui la classe $APP_{d^*}$ è chiusa rispetto alle funzioni Booleane di grado polinomiale. Allora $APP_{d^*} = \widehat{LT}_{d^*+1}$.*

*Dimostrazione.* Sia $f \in \widehat{LT}_{d^*+1}$; esiste allora un circuito a soglia $C$ di profondità $d^* + 1$, e dimensione e peso polinomiali, che calcola $f$.

Siano $C_1, C_2, \ldots, C_m$ i sottocircuiti di $C$ ottenuti rimuovendo da quest'ultimo il threshold gate $T$ d'uscita; siano inoltre $f_1, f_2, \ldots, f_m \in \widehat{LT}_{d^*}$ ed $h \in \widehat{LT}_1$ rispettivamente le funzioni Booleane calcolate da $C_1, C_2, \ldots, C_m$ e da $T$. Possiamo allora scrivere la seguente uguaglianza:

$$f(\mathbf{x}) = h\big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\big) \tag{3.7}$$

Poiché $\widehat{LT}_{d^*} \subseteq APP_{d^*}$, abbiamo che $f_1, f_2, \ldots, f_m \in APP_{d^*}$; d'altra parte, $h \in BIN_m$ con $m \leq p(n)$ per un opportuno polinomio $p(n)$. Essendo $APP_{d^*}$ chiuso rispetto alle funzioni Booleane di grado polinomiale, dall'uguaglianza (3.7) possiamo concludere che $f \in APP_{d^*}$, ovvero che $\widehat{LT}_{d^*+1} \subseteq APP_{d^*}$. Per il Teorema 3.1, $APP_{d^*} \subseteq \widehat{LT}_{d^*+1}$, quindi $APP_{d^*} = \widehat{LT}_{d^*+1}$. ∎

Il teorema appena dimostrato può anche essere interpretato nella maniera seguente: se esiste un intero $d^*$ per il quale la classe $APP_{d^*}$ è chiusa rispetto alle funzioni Booleane di grado polinomiale, allora anche la classe $\widehat{LT}_{d^*+1}$ soddisfa la medesima proprietà di chiusura (poiché coincide con $APP_{d^*}$). Pertanto, supporre che esista un intero $d^* \geq 1$ in corrispondenza del quale la classe $APP_{d^*}$ è chiusa rispetto alle funzioni Booleane di grado polinomiale è equivalente a supporre che esista un intero $\overline{d} \geq 2$ per il quale la classe $\widehat{LT}_{\overline{d}}$ è chiusa rispetto alle funzioni di grado polinomiale: è sufficiente porre $\overline{d} = d^* + 1$[2].

Possiamo ora enunciare il seguente teorema, la cui dimostrazione ricalca in maniera quasi esatta quella del Teorema 3.9.

**Teorema 3.10.** *Sia $\overline{d} \geq 2$ un intero per cui la classe $\widehat{LT}_{\overline{d}}$ è chiusa rispetto alle funzioni Booleane di grado polinomiale. Allora $\widehat{LT}_{\overline{d}} = \widehat{LT}_{\overline{d}+1}$.* ∎

Ecco infine il teorema che ci consente di mostrare che, nell'ipotesi che la classe $APP_{d^*}$ sia chiusa rispetto alle funzioni Booleane di grado polinomiale, la classe $TC^0$ collassa nella classe $APP_{d^*}$:

**Teorema 3.11.** *Sia $\overline{d} \geq 2$ un intero per cui vale l'uguaglianza $\widehat{LT}_{\overline{d}} = \widehat{LT}_{\overline{d}+1}$; allora, per ogni intero $n \geq 1$ vale l'uguaglianza $\widehat{LT}_{\overline{d}} = \widehat{LT}_{\overline{d}+n}$.*

*Dimostrazione.* Per induzione su $n$. Per $n = 1$, abbiamo l'uguaglianza $\widehat{LT}_{\overline{d}} = \widehat{LT}_{\overline{d}+1}$, che è vera per ipotesi.

Si supponga quindi che valga l'uguaglianza $\widehat{LT}_{\overline{d}+n} = \widehat{LT}_{\overline{d}}$, e mostriamo che in tal caso vale anche l'uguaglianza $\widehat{LT}_{\overline{d}+(n+1)} = \widehat{LT}_{\overline{d}}$.

Sia $f \in \widehat{LT}_{\overline{d}+(n+1)}$; esiste allora un circuito $C$ di profondità $\overline{d} + (n+1)$, e dimensione e peso polinomiali, che calcola $f$. Operando come nel Teorema 3.9, sia $T$ il threshold gate d'uscita di $C$, e siano $C_1, C_2, \ldots, C_m$ i sottocircuiti di $C$ ottenuti rimuovendo $T$ da $C$. Siano poi $f_1, f_2, \ldots, f_m \in \widehat{LT}_{\overline{d}+n}$ le funzioni calcolate da $C_1, C_2, \ldots, C_m$, e sia $h \in \widehat{LT}_1$ la funzione calcolata da $T$. Per ipotesi induttiva, $f_1, f_2, \ldots, f_m \in \widehat{LT}_{\overline{d}}$, e quindi $f \in \widehat{LT}_{\overline{d}+1} = \widehat{LT}_{\overline{d}}$, da cui si può concludere che $\widehat{LT}_{\overline{d}+(n+1)} \subseteq \widehat{LT}_{\overline{d}}$. Chiaramente vale anche $\widehat{LT}_{\overline{d}} \subseteq \widehat{LT}_{\overline{d}+(n+1)}$, e quindi resta dimostrata l'uguaglianza $\widehat{LT}_{\overline{d}} = \widehat{LT}_{\overline{d}+(n+1)}$. ∎

Potendo scrivere:

$$TC^0 = \bigcup_{d=1}^{+\infty} \widehat{LT}_d = \bigcup_{d=1}^{\overline{d}} \widehat{LT}_d \cup \bigcup_{d=\overline{d}+1}^{+\infty} \widehat{LT}_d = \bigcup_{d=1}^{\overline{d}} \widehat{LT}_d \cup \bigcup_{n=1}^{+\infty} \widehat{LT}_{\overline{d}+n}$$

[2]Si noti che possiamo escludere a priori che la classe $\widehat{LT}_1$ sia chiusa rispetto alle funzioni Booleane di grado polinomiale. Per il Teorema 1.10, infatti, $\widehat{LT}_1$ non è chiusa neppure rispetto alle funzioni di grado costante.

abbiamo che la prima unione genera l'insieme $\widehat{LT}_{\overline{d}}$, dato che $\widehat{LT}_d \subseteq \widehat{LT}_{\overline{d}}$ per ogni intero $d \leq \overline{d}$, mentre la seconda unione genera l'insieme $\widehat{LT}_{\overline{d}}$ a causa del Teorema 3.11. Otteniamo allora che:

$$TC^0 = \widehat{LT}_{\overline{d}} \cup \widehat{LT}_{\overline{d}} = \widehat{LT}_{\overline{d}}$$

ovvero, come è stato detto, l'intera classe $TC^0$ *collassa* sulla classe $\widehat{LT}_{\overline{d}}$.

Ricordando infine che si era posto:

$$\overline{d} = d^* + 1$$

per il Teorema 3.9 abbiamo che:

$$TC^0 = \widehat{LT}_{d^*+1} = APP_{d^*}$$

ovvero, equivalentemente, l'intera classe $TC^0$ collassa sulla classe $APP_{d^*}$.

Riassumendo, abbiamo dimostrato che vale il seguente teorema.

**Teorema 3.12.** *Sia $d^* \geq 1$ un intero per cui la classe $APP_{d^*}$ è chiusa rispetto alle funzioni Booleane di grado polinomiale. Allora $TC^0 = APP_{d^*}$.* ∎

È facile verificare che, per avere il collasso dell'intera classe $TC^0$ sulla classe $APP_{d^*}$ non è necessario che $APP_{d^*}$ sia chiusa rispetto a *tutte* le funzioni Booleane di grado polinomiale. Esaminando infatti le dimostrazioni dei Teoremi 3.9 e 3.11, si nota subito che è sufficiente che la funzione $h$ appartenga alla classe $\widehat{LT}_1$, e che quindi il teorema vale anche se $APP_{d^*}$ è chiusa rispetto ad ogni funzione Booleana di grado polinomiale *appartenente ad* $\widehat{LT}_1$.

Viceversa, si può osservare che se esiste un intero $d^* \geq 1$ per cui vale $TC^0 = APP_{d^*}$ allora $APP_{d^*}$ è necessariamente chiusa rispetto alle funzioni Booleane di grado polinomiale appartenenti ad $\widehat{LT}_1$. Infatti, se così non fosse, basterebbe prendere un numero polinomiale di funzioni $f_1, f_2, \ldots, f_{p(n)} \in APP_{d^*}$ e una funzione $h \in \widehat{LT}_1$; per il Teorema 3.1 avremmo che $f_1, f_2, \ldots, f_{p(n)} \in \widehat{LT}_{d^*+1}$, e quindi:

$$f(\mathbf{x}) = h\big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{p(n)}(\mathbf{x})\big)$$

apparterrebbe alla classe $\widehat{LT}_{d^*+2}$ ma non alla classe $APP_{d^*}$, contro l'ipotesi che sia $TC^0 = APP_{d^*}$. Possiamo pertanto concludere dicendo che vale il seguente teorema.

**Teorema 3.13.** *Condizione necessaria e sufficiente affinché sia $TC^0 = APP_{d^*}$ per un intero $d^* \geq 1$ fissato è che la classe $APP_{d^*}$ sia chiusa rispetto alle funzioni Booleane di grado polinomiale appartenenti alla classe $\widehat{LT}_1$.* ∎

# Capitolo 4

# Tecniche spettrali

In questo capitolo introduciamo alcune tecniche analitiche che si sono rivelate degli strumenti molto potenti per determinare se una data funzione Booleana appartiene alla classe $\widehat{LT}_2$.

Nonostante le tecniche spettrali siano da considerare sicuramente sofisticate (e conviene pertanto applicarle soltanto quando le tecniche elementari esposte nel Capitolo 2 falliscono), nondimeno hanno una semplice interpretazione geometrica, la quale costituisce un potente supporto all'intuizione. Infatti, oltre a suggerire nuovi risultati, l'interpretazione geometrica consente molto spesso di generalizzare quelli esistenti, o perlomeno di semplificare alcune dimostrazioni. Tra i risultati ottenuti con l'uso di queste tecniche, spicca sicuramente il limite inferiore sul numero di funzioni in ingresso necessarie ad un threshold gate affinché possa calcolare una data funzione Booleana.

## 4.1 Trasformata di Fourier per le funzioni Booleane

La trasformata di Fourier $n$-dimensionale sull'insieme $\{0,1\}^n$ è un'applicazione dell'insieme $\Re^{\{0,1\}^n}$ delle funzioni reali a $n$ variabili Booleane in se stesso.

Prima di considerare la trasformata, diamo la seguente definizione.

**Definizione 4.1.** Chiamiamo *funzione parità* $\varphi_\alpha$, per $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$, la funzione da $\{0,1\}^n$ a $\{1,-1\}$ così definita:

$$\varphi_\alpha(\mathbf{x}) = (-1)^{\sum_{i=1}^{n} \alpha_i x_i}$$

■

Come si può notare, fissato un valore di $\alpha \in \{0,1\}^n$, la funzione $\varphi_\alpha$ non è altro che l'applicazione della funzione `PARITY` al sottoinsieme di variabili Booleane $\{x_i \mid i \in \{1,2,\ldots,n\} \text{ e } \alpha_i = 1\}$, con la sola differenza che $\varphi_\alpha$ assume valori in $\{1,-1\}$ anziché in $\{0,1\}$. Questo significa che $\varphi_\alpha$ può essere calcolata da un circuito di dimensione polinomiale, peso polinomiale e profondità

2 (ossia appartiene a $\widehat{LT}_2$), ottenuto a partire da quello che calcola la funzione `PARITY` corrispondente modificando semplicemente la funzione di trasferimento del neurone d'uscita in modo che emetta i valori $\{1, -1\}$ anziché $\{0, 1\}$.

Si osservi che, al variare di $\alpha$ su $\{0,1\}^n$, vengono selezionati tutti i possibili sottoinsiemi di variabili da $\{x_1, x_2, \ldots, x_n\}$ sui quali calcolare la parità (ovvero $\alpha$ funge da *maschera* o *filtro* per l'insieme $\{x_1, x_2, \ldots, x_n\}$); avremo quindi un totale di $2^n$ funzioni $\varphi_\alpha$ distinte.

Vediamo ora come *ogni* funzione Booleana $f \in BIN_n$ può essere espressa tramite un'opportuna combinazione lineare delle $2^n$ funzioni $\varphi_\alpha$ a $n$ argomenti. A tale scopo, introduciamo anzitutto la seguente definizione.

**Definizione 4.2.** Siano date due funzioni $f, g \in \Re^{\{0,1\}^n}$; il loro *prodotto interno* $\langle f, g \rangle$ è definito come:

$$\langle f, g \rangle = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) \cdot g(\mathbf{x})$$

■

Come si può notare, se consideriamo $f$ e $g$ come due vettori a $2^n$ componenti in $\{0, 1\}$, $\langle f, g \rangle$ è l'usuale prodotto scalare fra vettori di $\Re^{2^n}$, normalizzato dal fattore $\frac{1}{2^n}$.

A questo punto, possiamo calcolare il prodotto interno tra funzioni di parità, distinguendo il caso del prodotto fra $\varphi_\alpha$ e se stessa da quello del prodotto fra due funzioni $\varphi_{\alpha_1}$ e $\varphi_{\alpha_2}$ distinte.

A tale scopo, fissiamo $\alpha \in \{0,1\}^n$ e calcoliamo $\langle \varphi_\alpha, \varphi_\alpha \rangle$:

$$\begin{aligned} \langle \varphi_\alpha, \varphi_\alpha \rangle &= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\sum_{i=1}^n \alpha_i x_i} \cdot (-1)^{\sum_{i=1}^n \alpha_i x_i} = \\ &= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{2 \cdot \sum_{i=1}^n \alpha_i x_i} = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} 1 = \frac{1}{2^n} \cdot 2^n = 1 \end{aligned} \tag{4.1}$$

Prendiamo ora $\alpha_1, \alpha_2 \in \{0,1\}^n$, con $\alpha_1 \neq \alpha_2$, e calcoliamo $\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle$:

$$\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x})$$

Concentriamo l'attenzione sul prodotto $\varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x})$:

$$\varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x}) = (-1)^{\sum_{i=1}^n \alpha_{1,i} x_i} \cdot (-1)^{\sum_{i=1}^n \alpha_{2,i} x_i} = (-1)^{\sum_{i=1}^n (\alpha_{1,i} + \alpha_{2,i}) x_i}$$

La quantità $\alpha_{1,i} + \alpha_{2,i}$ può assumere solamente i valori 0, 1 e 2; definiti allora i seguenti insiemi di indici:

$$\begin{aligned} \lambda_0 &= \{i : 1 \leq i \leq n \text{ e } \alpha_{1,i} + \alpha_{2,i} = 0\} \\ \lambda_1 &= \{i : 1 \leq i \leq n \text{ e } \alpha_{1,i} + \alpha_{2,i} = 1\} \\ \lambda_2 &= \{i : 1 \leq i \leq n \text{ e } \alpha_{1,i} + \alpha_{2,i} = 2\} \end{aligned}$$

possiamo riscrivere $\varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x})$ come:

$$\begin{aligned}\varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x}) &= (-1)^{\sum_{i\in\lambda_0} 0} \cdot (-1)^{\sum_{i\in\lambda_1} x_i} \cdot (-1)^{2\cdot\sum_{i\in\lambda_2} x_i} = \\ &= (-1)^0 \cdot (-1)^{\sum_{i\in\lambda_1} x_i} \cdot [(-1)^2]^{\sum_{i\in\lambda_2} x_i} = \\ &= 1 \cdot (-1)^{\sum_{i\in\lambda_1} x_i} \cdot 1^{\sum_{i\in\lambda_2} x_i} = (-1)^{\sum_{i\in\lambda_1} x_i}\end{aligned}$$

Dato che, considerando $\alpha_{1,i}$ e $\alpha_{2,i}$ come parametri Booleani, si ha che $i \in \lambda_1$ se e solo se $\alpha_{1,i} \oplus \alpha_{2,i} = 1$, possiamo scrivere:

$$\varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x}) = (-1)^{\sum_{i=1}^{n}(\alpha_{1,i}\oplus\alpha_{2,i})x_i}$$

Allora, dati i due vettori $\alpha_1, \alpha_2 \in \{0,1\}^n$, possiamo indicare con $\alpha_1 \oplus \alpha_2$ il vettore in $\{0,1\}^n$ ottenuto facendo l'`XOR` (`OR` esclusivo) bit a bit tra $\alpha_1$ e $\alpha_2$, e possiamo quindi scrivere in forma compatta:

$$\varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x}) = \varphi_{\alpha_1\oplus\alpha_2}(\mathbf{x})$$

Posto $\alpha = \alpha_1 \oplus \alpha_2$, il vettore $\alpha$ gode di una simpatica proprietà:

$$\alpha = \mathbf{0}_n \iff \alpha_1 = \alpha_2$$

come ci si può rendere conto esaminando la tavola di verità dell'`XOR` in riferimento a ciascuna componente $\alpha_i$. Questa proprietà è molto utile in quanto semplifica notevolmente i calcoli. Infatti, sia $\alpha$ un vettore in $\{0,1\}^n$; abbiamo:

$$\begin{aligned}\langle\varphi_\alpha, \varphi_\alpha\rangle &= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} \varphi_\alpha(\mathbf{x}) \cdot \varphi_\alpha(\mathbf{x}) = \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} \varphi_{\alpha\oplus\alpha}(\mathbf{x}) = \\ &= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} \varphi_{\mathbf{0}_n}(\mathbf{x}) = \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{\sum_{i=1}^{n} 0\cdot x_i} = \\ &= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} 1 = \frac{1}{2^n}\cdot 2^n = 1\end{aligned}$$

riottenendo così il risultato ricavato nell'equazione (4.1). Siano ora $\alpha_1$ e $\alpha_2$ due vettori *distinti* in $\{0,1\}^n$; poniamo $\alpha = \alpha_1 \oplus \alpha_2$ e calcoliamo:

$$\begin{aligned}\langle\varphi_{\alpha_1}, \varphi_{\alpha_2}\rangle &= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} \varphi_{\alpha_1}(\mathbf{x}) \cdot \varphi_{\alpha_2}(\mathbf{x}) = \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} \varphi_{\alpha_1\oplus\alpha_2}(\mathbf{x}) = \\ &= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} \varphi_\alpha(\mathbf{x})\end{aligned}$$

con $\alpha \neq \mathbf{0}_n$. Definito l'insieme di indici:

$$\lambda = \{i : \alpha_i = 1\}$$

possiamo riscrivere $\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle$ come:

$$\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\sum_{i=1}^{n} \alpha_i x_i} = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\sum_{i \in \lambda} x_i}$$

Pertanto, posto $k = |\lambda|$, i $2^n$ termini della sommatoria più esterna si suddividono in $2^k$ classi di equivalenza di $2^{n-k}$ elementi ciascuna, rispetto alla relazione di equivalenza così definita tra coppie di vettori $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$:

$$\mathbf{x} \sim \mathbf{y} \iff x_i = y_i \quad \forall\, i \in \lambda$$

Ciascuno dei $2^{n-k}$ elementi di una data classe di equivalenza porta al medesimo valore della quantità $(-1)^{\sum_{i \in \lambda} x_i}$, poiché tali elementi differiscono unicamente nei valori di $x_i$ per i quali $i \notin \lambda$; per valutare $\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle$ è sufficiente allora considerare un solo rappresentante di ciascuna delle $2^k$ classi di equivalenza. Inoltre, come si è già detto, la quantità $(-1)^{\sum_{i \in \lambda} x_i}$ non è altro che la funzione $\texttt{PARITY}_k$ con valori di uscita in $\{1, -1\}$ anziché in $\{0, 1\}$, e vale pertanto 1 in $2^{k-1}$ configurazioni e $-1$ nelle restanti $2^{k-1}$ configurazioni. Allora, se $\alpha_1 \neq \alpha_2$ si ottiene:

$$\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\sum_{i \in \lambda} x_i} = 0$$

Questo significa che le funzioni $\varphi_{\alpha_1}$ e $\varphi_{\alpha_2}$ sono *ortogonali*.

Abbiamo così tutti gli elementi per dimostrare il seguente teorema.

**Teorema 4.1.** *La famiglia di funzioni* $\{\varphi_\alpha\}_{\alpha \in \{0,1\}^n}$ *costituisce una base ortonormale per lo spazio vettoriale* $\Re^{\{0,1\}^n}$.

*Dimostrazione.* La famiglia $\{\varphi_\alpha\}_{\alpha \in \{0,1\}^n}$ è formata da $2^n$ funzioni; abbiamo inoltre visto che, prese due funzioni qualsiasi $\varphi_{\alpha_1}$ e $\varphi_{\alpha_2}$ della famiglia, si ha:

$$\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle = \begin{cases} 1 & \text{se } \alpha_1 = \alpha_2 \\ 0 & \text{se } \alpha_1 \neq \alpha_2 \end{cases}$$

La proprietà $\langle \varphi_{\alpha_1}, \varphi_{\alpha_2} \rangle = 0$ per $\alpha_1 \neq \alpha_2$ significa che le funzioni della famiglia sono a due a due ortogonali; dato che $\varphi_\alpha \in \Re^{\{0,1\}^n}$ per ogni $\alpha \in \{0,1\}^n$, e lo spazio vettoriale $\Re^{\{0,1\}^n}$ ha dimensione $2^n$, possiamo concludere che la famiglia di funzioni $\{\varphi_\alpha\}_{\alpha \in \{0,1\}^n}$ costituisce una base ortogonale per $\Re^{\{0,1\}^n}$. Il fatto poi che valga $\langle \varphi_\alpha, \varphi_\alpha \rangle = 1$ per ogni $\alpha \in \{0,1\}^n$ significa che la base è pure ortonormale rispetto alla norma:

$$\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} f^2(\mathbf{x})}$$

■

**Definizione 4.3.** La base ortonormale costituita dalla famiglia di funzioni $\{\varphi_\alpha\}_{\alpha\in\{0,1\}^n}$ si chiama comunemente *base di Hadamard* o *base di Walsh*.

Per comodità, indicheremo con $W_n$ la base di Walsh dello spazio vettoriale $\Re^{\{0,1\}^n}$. ∎

Conseguenza immediata della definizione precedente è che *ogni* funzione $f \in \Re^{\{0,1\}^n}$ può essere espressa mediante una combinazione lineare delle funzioni $\varphi_\alpha \in W_n$:

$$f(\mathbf{x}) = \sum_{\alpha\in\{0,1\}^n} \widehat{f}(\alpha)\varphi_\alpha(\mathbf{x}) \tag{4.2}$$

dove i coefficienti $\widehat{f}(\alpha)$ sono numeri reali, comunemente chiamati *coefficienti di Fourier*. L'insieme dei coefficienti di Fourier di una data funzione $f$:

$$\mathcal{F}_n(f) = \{\widehat{f}(\alpha) \mid \alpha \in \{0,1\}^n\}$$

viene comunemente chiamato *spettro* della funzione, mentre la funzione:

$$\widehat{f} : \{0,1\}^n \to \Re$$

che ad ogni $\alpha \in \{0,1\}^n$ associa il corrispondente coefficiente $\widehat{f}(\alpha)$ viene comunemente chiamata la *trasformata di Fourier di* $f$. Con lo stesso nome, poi, si indica solitamente anche la funzione:

$$\mathcal{F}_n : \Re^{\{0,1\}^n} \to \Re^{\{0,1\}^n}$$

che ad ogni funzione $f \in \Re^{\{0,1\}^n}$ associa il suo spettro $\mathcal{F}_n$, visto come vettore di $\Re^{2^n}$. È in questa accezione che va interpretata la frase, che apre questo paragrafo, secondo cui la trasformata di Fourier $n$-dimensionale su $\{0,1\}^n$ è un'applicazione dell'insieme $\Re^{\{0,1\}^n}$ in se stesso.

Si pone a questo punto un problema pratico: data una funzione $f \in \Re^{\{0,1\}^n}$, come si calcolano i valori $\widehat{f}(\alpha)$? Sia $\overline{\alpha} \in \{0,1\}^n$ un valore fissato per $\alpha$; ricordando che la rappresentazione spettrale di $f$ è (equazione (4.2)):

$$f(\mathbf{x}) = \sum_{\alpha\in\{0,1\}^n} \widehat{f}(\alpha)\varphi_\alpha(\mathbf{x})$$

calcoliamo il prodotto $\langle f, \varphi_{\overline{\alpha}} \rangle$:

$$\begin{aligned}
\langle f, \varphi_{\overline{\alpha}} \rangle &= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) \cdot \varphi_{\overline{\alpha}}(\mathbf{x}) = \\
&= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \left[ \sum_{\alpha \in \{0,1\}^n} \widehat{f}(\alpha) \varphi_\alpha(\mathbf{x}) \right] \varphi_{\overline{\alpha}}(\mathbf{x}) = \\
&= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \sum_{\alpha \in \{0,1\}^n} \widehat{f}(\alpha) \varphi_\alpha(\mathbf{x}) \varphi_{\overline{\alpha}}(\mathbf{x}) = \\
&= \sum_{\alpha \in \{0,1\}^n} \widehat{f}(\alpha) \cdot \left[ \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \varphi_\alpha(\mathbf{x}) \cdot \varphi_{\overline{\alpha}}(\mathbf{x}) \right] = \\
&= \sum_{\alpha \in \{0,1\}^n} \widehat{f}(\alpha) \cdot \langle \varphi_\alpha, \varphi_{\overline{\alpha}} \rangle = \\
&= \widehat{f}(\overline{\alpha}) \cdot \langle \varphi_{\overline{\alpha}}, \varphi_{\overline{\alpha}} \rangle + \sum_{\alpha \neq \overline{\alpha}} \widehat{f}(\alpha) \cdot \langle \varphi_\alpha, \varphi_{\overline{\alpha}} \rangle
\end{aligned}$$

Per l'ortogonalità delle $\varphi_\alpha$, ogni termine della sommatoria vale 0; inoltre $\langle \varphi_{\overline{\alpha}}, \varphi_{\overline{\alpha}} \rangle = 1$, così che:

$$\widehat{f}(\overline{\alpha}) = \langle f, \varphi_{\overline{\alpha}} \rangle = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) \cdot \varphi_{\overline{\alpha}}(\mathbf{x})$$

Ossia, per ottenere il coefficiente di Fourier $\widehat{f}(\overline{\alpha})$ di una funzione $f \in \Re^{\{0,1\}^n}$ per un valore $\overline{\alpha} \in \{0,1\}^n$ fissato è sufficiente calcolare il prodotto interno tra $f$ e $\varphi_{\overline{\alpha}}$.

Siamo ora in grado di dimostrare due utili proprietà della trasformata di Fourier. La prima è la *linearità*, ossia:

$$\widehat{(f+g)}(\alpha) = \widehat{f}(\alpha) + \widehat{g}(\alpha) \qquad \forall f, g \in \Re^{\{0,1\}^n} \text{ e } \forall \alpha \in \{0,1\}^n$$

Infatti, per ogni coppia di funzioni $f, g \in \Re^{\{0,1\}^n}$ fissate e per ogni $\alpha \in \{0,1\}^n$ fissato, abbiamo:

$$\begin{aligned}
\widehat{(f+g)}(\alpha) = \langle f+g, \varphi_\alpha \rangle &= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \big[ f(\mathbf{x}) + g(\mathbf{x}) \big] \cdot \varphi_\alpha(\mathbf{x}) = \\
&= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} \big[ f(\mathbf{x}) \varphi_\alpha(\mathbf{x}) + g(\mathbf{x}) \varphi_\alpha(\mathbf{x}) \big] = \\
&= \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) \cdot \varphi_\alpha(\mathbf{x}) + \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} g(\mathbf{x}) \cdot \varphi_\alpha(\mathbf{x}) = \\
&= \langle f, \varphi_\alpha \rangle + \langle g, \varphi_\alpha \rangle = \widehat{f}(\alpha) + \widehat{g}(\alpha)
\end{aligned}$$

La seconda proprietà che dimostriamo è l'*identità di Parseval*:

$$\sum_{\mathbf{x}\in\{0,1\}^n} f^2(\mathbf{x}) = 2^n \cdot \sum_{\alpha\in\{0,1\}^n} \widehat{f}^{\,2}(\alpha)$$

Anch'essa deriva dall'ortogonalità delle funzioni della base di Walsh; la dimostrazione dell'identità segue semplicemente da alcuni calcoli algebrici:

$$\begin{aligned}
\frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} f^2(\mathbf{x}) &= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} f(\mathbf{x})\cdot f(\mathbf{x}) = \\
&= \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n}\left[\left(\sum_{\alpha\in\{0,1\}^n}\widehat{f}(\alpha)\varphi_\alpha(\mathbf{x})\right)\left(\sum_{\beta\in\{0,1\}^n}\widehat{f}(\beta)\varphi_\beta(\mathbf{x})\right)\right] = \\
&= \sum_{\alpha\in\{0,1\}^n}\sum_{\beta\in\{0,1\}^n}\widehat{f}(\alpha)\widehat{f}(\beta)\cdot\frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n}\varphi_\alpha(\mathbf{x})\cdot\varphi_\beta(\mathbf{x}) = \\
&= \sum_{\alpha\in\{0,1\}^n}\sum_{\beta\in\{0,1\}^n}\widehat{f}(\alpha)\widehat{f}(\beta)\cdot\langle\varphi_\alpha,\varphi_\beta\rangle
\end{aligned}$$

Poiché per $\alpha \neq \beta$ si ha $\langle\varphi_\alpha,\varphi_\beta\rangle = 0$, nella doppia sommatoria restano solamente i termini per i quali $\alpha = \beta$:

$$\frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n} f^2(\mathbf{x}) = \sum_{\alpha\in\{0,1\}^n}\widehat{f}(\alpha)\widehat{f}(\alpha)\cdot\langle\varphi_\alpha,\varphi_\alpha\rangle = \sum_{\alpha\in\{0,1\}^n}\widehat{f}^{\,2}(\alpha)$$

Come abbiamo più volte ripetuto nel corso dell'esposizione, a volte conviene considerare funzioni i cui ingressi assumono valori in $\{1,-1\}$ anziché in $\{0,1\}$. Più precisamente, data una funzione $f \in \Re^{\{0,1\}^n}$ che dipende dalle variabili $x_1, x_2, \ldots, x_n$ a valori in $\{0,1\}$, possiamo sempre esprimerla sotto forma di una funzione $F \in \Re^{\{1,-1\}^n}$ che dipende dalle variabili $X_i = (-1)^{x_i}$ a valori in $\{1,-1\}$. In tal caso, le funzioni $\varphi_\alpha$ della base di Walsh possono essere scritte nel modo seguente:

$$\varphi_\alpha(X_1, X_2, \ldots, X_n) = (-1)^{\sum_{i=1}^n \alpha_i x_i} = \prod_{i=1}^n (-1)^{x_i\alpha_i} = \prod_{i=1}^n X_i^{\alpha_i}$$

e vengono solitamente indicate col simbolo $X^\alpha$; la trasformata di Fourier di una funzione $f \in \Re^{\{1,-1\}^n}$ assume allora la seguente forma:

$$f(X_1, X_2, \ldots, X_n) = \sum_{\alpha\in\{0,1\}^n}\widehat{f}(\alpha)\cdot X^\alpha$$

e viene chiamata anche *rappresentazione polinomiale* di $f$, mentre ciascun $X^\alpha$ viene detto *monomio* della rappresentazione. Si noti che $X^\alpha$ è semplicemente una diversa notazione per la funzione $\varphi_\alpha$, utilizzata per indicare che si lavora con

variabili a valori in $\{1, -1\}$ anziché in $\{0, 1\}$: la trasformata e le sue proprietà restano invariate.

Chiudiamo questo paragrafo con un semplice esempio di rappresentazione polinomiale. Sia $f(x_1, x_2) = x_1 \oplus x_2$ la funzione `OR`-esclusivo (`XOR`) delle due variabili $x_1$ e $x_2$, che supponiamo assumano i loro valori in $\{1, -1\}$. È facile convincersi che la rappresentazione polinomiale di $f$ è:

$$f(x_1, x_2) = x_1 x_2$$

ovvero il prodotto algebrico tra i valori delle variabili Booleane coincide con l'operazione Booleana `XOR` se interpretiamo i valori Booleani $\{\texttt{falso}, \texttt{vero}\}$ come $\{1, -1\}$. Dalla rappresentazione polinomiale possiamo ricavare immediatamente lo spettro $\mathcal{F}_2(f)$ della funzione $f$:

$$\mathcal{F}_2(f) = \Big\{ \widehat{f}\big((0,0)\big) = 0,\ \ \widehat{f}\big((0,1)\big) = 0,\ \ \widehat{f}\big((1,0)\big) = 0,\ \ \widehat{f}\big((1,1)\big) = 1 \Big\}$$

Infine, dall'esempio appena presentato è facile rendersi conto che la rappresentazione polinomiale di *ogni* funzione Booleana $f$ è *lineare* in ciascuna delle variabili da cui $f$ dipende, poiché $x^2 = 1$ per $x \in \{1, -1\}$.

## 4.2 Il Teorema di Rappresentazione

Come abbiamo precedentemente detto, se si vuole calcolare il coefficiente di Fourier $\widehat{f}(\alpha)$ di una data funzione Booleana $f$ per un valore $\alpha \in \{0, 1\}^n$ fissato, è sufficiente calcolare il prodotto interno $\langle f, \varphi_\alpha \rangle$; lo spettro di $f$ è allora ottenibile iterando il calcolo al variare di $\alpha$.

Un metodo alternativo, che consente di calcolare l'intero spettro $\mathcal{F}_n(f)$ di una data funzione Booleana $f$, è costituito dal Teorema di Rappresentazione, tratto da un articolo di Bruck [2]. Il concetto principale utilizzato nella dimostrazione del teorema è quello di *matrice di Sylvester*, che è una particolare matrice di Hadamard (si veda la nota a pagina 46) definita ricorsivamente nel modo seguente:

$$H_1 = [1]$$
$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$H_{2^{n+1}} = \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix}$$

Ad esempio:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Le matrici di Sylvester sono facilmente invertibili: è facile infatti verificare che sono simmetriche, per cui $H_{2^n}^T = H_{2^n}$; inoltre, per l'ortogonalità delle righe e delle colonne, vale:

$$H_{2^n} \cdot H_{2^n}^T = 2^n \, I_{2^n}$$

Possiamo allora scrivere:

$$H_{2^n} \cdot H_{2^n}^{-1} = I_{2^n} = \frac{1}{2^n} H_{2^n} \cdot H_{2^n}^T$$

da cui, semplificando, si ottiene:

$$H_{2^n}^{-1} = \frac{1}{2^n} H_{2^n}^T = \frac{1}{2^n} H_{2^n}$$

Abbiamo quindi tutti gli strumenti per dimostrare il cosiddetto Teorema di Rappresentazione.

**Teorema 4.2.** *Sia $f(\mathbf{x})$ una funzione Booleana, con $\mathbf{x} \in \{1,-1\}^n$, e sia $\mathbf{v}_n$ il vettore $\big(f(1,1,\ldots,1), f(1,1,\ldots,-1), \ldots, f(-1,-1,\ldots,-1)\big)$, già utilizzato nella Definizione 2.11. Lo spettro $\mathcal{F}_n(f)$ di $f$ può essere calcolato come segue:*

$$\mathcal{F}_n(f) = \frac{1}{2^n} \, \mathbf{v}_n H_{2^n}$$

*Dimostrazione.* La dimostrazione è costruttiva ed è per induzione su $n$. L'idea è quella di calcolare $\mathcal{F}_n(f)$ risolvendo un sistema di equazioni lineari.

Cominciamo calcolando $\mathcal{F}_n(f)$ per $n = 1$; essendo $f$ funzione di una sola variabile, possiamo scrivere la rappresentazione polinomiale di $f$ (che, ricordiamo, è lineare nelle variabili da cui dipende) come:

$$f(x_1) = a_0 + a_1 x_1$$

Al variare di $x_1$ in $\{1,-1\}$ otteniamo il seguente sistema di equazioni lineari:

$$\begin{aligned} f(1) &= a_0 + a_1 \\ f(-1) &= a_0 - a_1 \end{aligned}$$

che può essere scritto in forma più compatta come:

$$\mathbf{v}_1 = \mathcal{F}_1(f) H_2 \tag{4.3}$$

Per l'invertibilità della matrice di Hadamard $H_2$ si ottiene pertanto:

$$\mathcal{F}_1(f) = \mathbf{v}_1 H_2^{-1} = \frac{1}{2} \, \mathbf{v}_1 H_2$$

Si supponga ora che l'uguaglianza (4.3) valga per $n$:

$$\mathbf{v}_n = \mathcal{F}_n(f) H_{2^n} \tag{4.4}$$

e mostriamo che vale anche per $n+1$; considerando i valori che può assumere la variabile $x_{n+1}$, e combinandoli in tutti i modi possibili con le configurazioni di valori assunte dalle variabili $x_1, x_2, \ldots, x_n$, otteniamo:

$$\mathbf{v}_{n+1} = \mathcal{F}_{n+1}(f) \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix} = \mathcal{F}_{n+1}(f) H_{2^{n+1}}$$

Avendo dimostrato che l'uguaglianza (4.4) vale per ogni intero $n$, è possibile sfruttare l'invertibilità delle matrici di Hadamard $H_{2^n}$ per ricavare lo spettro $\mathcal{F}_n(f)$:

$$\mathcal{F}_n(f) = \mathbf{v}_n H_{2^n}^{-1} = \frac{1}{2^n} \mathbf{v}_n H_{2^n}$$

■

Come esempio di applicazione del Teorema di Rappresentazione, si consideri la funzione $\texttt{AND}_2$:

$$f(x_1, x_2) = x_1 \wedge x_2$$

Allora abbiamo:

$$\mathbf{v}_2 = \big(f(1,1) = 1,\ f(1,-1) = 1,\ f(-1,1) = 1,\ f(-1,-1) = -1\big)$$

Per il Teorema 4.2, vale:

$$\frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ -0.5 \end{bmatrix}$$

da cui si ottiene:

$$f(x_1, x_2) = \frac{1}{2}(1 + x_1 + x_2 - x_1 x_2)$$

Utilizzare il Teorema 4.2 con le matrici di Sylvester per calcolare lo spettro di una funzione Booleana $f$ ha uno svantaggio che rende il metodo virtualmente impossibile da utilizzare in pratica: per calcolare $\mathcal{F}_n(f)$ occorre prima calcolare il valore di tutti gli elementi del vettore $\mathbf{v}_n$ — ovvero l'intera tavola di verità di $f$ — la cui dimensione cresce in maniera esponenziale all'aumentare di $n$. Poiché nella maggior parte dei casi che si presentano in pratica, la funzione di cui si vuole calcolare lo spettro risulta essere la disgiunzione (o la congiunzione) logica di sottofunzioni, oppure ancora la parità (`OR`-esclusivo) di sottofunzioni, mostriamo un metodo alternativo per il calcolo dello spettro. Tale metodo consiste semplicemente nel ricavare lo spettro di alcune funzioni di base — in questo caso delle funzioni `AND`, `OR` e `PARITY` — e di operare per sostituzione sulle sottofunzioni. Si osservi come questo metodo consenta, in linea di principio, di calcolare lo spettro di ogni funzione Booleana: è sufficiente scrivere la funzione in Forma Normale Disgiuntiva (DNF) o in Forma Normale Congiuntiva (CNF).

Purtroppo, anche tali rappresentazioni hanno in generale lunghezza esponenziale rispetto al numero di variabili di ingresso.

Enunciamo allora i teoremi che riguardano lo spettro delle funzioni di base AND, OR e PARITY.

**Teorema 4.3.** $\texttt{PARITY}(x_1, x_2, \ldots, x_n) = x_1 x_2 \cdots x_n$.

*Dimostrazione.* La quantità $x_1 x_2 \cdots x_n$ vale $-1$ se e solo se il numero di $-1$ tra gli $x_i$ è dispari. ∎

Il teorema precedente estende il risultato già trovato per $n = 2$. Come è già stato osservato in quell'occasione, posto $\mathcal{F}_n(\texttt{PARITY}) = \left\{\widehat{\texttt{PARITY}}(\alpha) \mid \alpha \in \{0,1\}^n\right\}$, avremo:

$$\widehat{\texttt{PARITY}}(\alpha) = \begin{cases} 1 & \text{se } \alpha = \mathbf{1}_n \\ 0 & \text{altrimenti} \end{cases}$$

**Teorema 4.4.** *Sia* $\mathcal{F}_n(\texttt{AND}) = \left\{\widehat{\texttt{AND}}(\alpha) \mid \alpha \in \{0,1\}^n\right\}$. *Allora:*

$$\widehat{\texttt{AND}}(\alpha) = \begin{cases} 1 - 2^{1-n} & \textit{se } \sum_{i=1}^n \alpha_i = 0 \\ 2^{1-n} & \textit{se } \sum_{i=1}^n \alpha_i \textit{ è dispari} \\ -2^{1-n} & \textit{se } \sum_{i=1}^n \alpha_i \neq 0 \textit{ è pari} \end{cases}$$

*Dimostrazione.* Si consideri il seguente polinomio:

$$g(x_1, x_2, \ldots, x_n) = \frac{1}{2^n} \prod_{i=1}^n (1 - x_i)$$

È facile rendersi conto che $g(x_1, x_2, \ldots, x_n) = 1$ se e solo se $(x_1, x_2, \ldots, x_n)$ è il vettore i cui elementi sono tutti uguali a $-1$, mentre in tutti gli altri casi $g(x_1, x_2, \ldots, x_n) = 0$. Da ciò segue che:

$$\texttt{AND}(x_1, x_2, \ldots, x_n) = 1 - 2 \cdot g(x_1, x_2, \ldots, x_n)$$

e il teorema risulta allora dimostrato considerando i coefficienti del polinomio $1 - 2 \cdot g(x_1, x_2, \ldots, x_n)$. ∎

**Teorema 4.5.** *Sia* $\mathcal{F}_n(\texttt{OR}) = \left\{\widehat{\texttt{OR}}(\alpha) \mid \alpha \in \{0,1\}^n\right\}$. *Allora:*

$$\widehat{\texttt{OR}}(\alpha) = \begin{cases} 2^{1-n} - 1 & \textit{se } \sum_{i=1}^n \alpha_i = 0 \\ 2^{1-n} & \textit{se } \sum_{i=1}^n \alpha_i \neq 0 \end{cases}$$

*Dimostrazione.* La dimostrazione segue immediatamente dal Teorema 4.4 e dal fatto che:

$$\texttt{OR}(\mathbf{x}) = -\texttt{AND}(-\mathbf{x})$$

∎

Mostriamo ora un semplice esempio di applicazione del metodo per sostituzione. Sia data la funzione Booleana:

$$f(x_1, x_2, x_3) = x_1 \wedge (x_2 \vee x_3)$$

Posto $y = (x_2 \vee x_3)$, abbiamo:

$$f(x_1, x_2, x_3) = \texttt{AND}(x_1, y) = \frac{1}{2}(1 + x_1 + y - x_1 y)$$
$$y = \texttt{OR}(x_2, x_3) = \frac{1}{2}(-1 + x_2 + x_3 + x_2 x_3)$$

Effettuando la sostituzione, otteniamo:

$$\begin{aligned} f(x_1,&x_2, x_3) = \\ &= \frac{1}{2}\left(1 + x_1 + \frac{1}{2}(-1 + x_2 + x_3 + x_2 x_3) - \frac{1}{2} x_1 (-1 + x_2 + x_3 + x_2 x_3)\right) = \\ &= \frac{1}{4}(1 + 3x_1 + x_2 + x_3 + x_2 x_3 - x_1 x_2 - x_1 x_3 - x_1 x_2 x_3) \end{aligned}$$

## 4.3 Rappresentazione polinomiale e tecniche spettrali

Mostriamo ora come la rappresentazione polinomiale può essere utilizzata per dimostrare che certe funzioni Booleane appartengono alla classe $\widehat{LT}_2$. Il risultato principale è costituito dal Teorema 4.6, tratto dal primo capitolo del libro di Roychowdhury, Siu e Orlitsky [14]. L'utilizzo di questo teorema e di quelli da esso derivati per stabilire l'appartenenza di una data funzione Booleana alla classe $\widehat{LT}_2$ costituisce una tecnica di analisi delle funzioni Booleane che va sotto il nome di *analisi spettrale*.

Diamo anzitutto la seguente definizione.

**Definizione 4.4.** Sia $f \in \Re^{2^n}$; definiamo la *norma spettrale* (o $L_1$-*norma*) di $f$ nel modo seguente:

$$\|f\|_1 = \sum_{\alpha \in \{0,1\}^n} |\widehat{f}(\alpha)|$$

■

Si noti che nella precedente definizione abbiamo richiesto che $f$ appartenga a $\Re^{2^n}$; questo perché la definizione di norma spettrale non dipende dai valori assunti dalle variabili di ingresso di $f$.

Prendiamo ora una funzione $f \in BIN_n$ e consideriamo la sua rappresentazione di Fourier:

$$f(\mathbf{x}) = \sum_{\alpha \in S} \widehat{f}(\alpha) \cdot \varphi_\alpha(\mathbf{x}) \tag{4.5}$$

dove con S abbiamo indicato l'insieme dei coefficienti di Fourier di $f$ non nulli:

$$S = \left\{\alpha \in \{0,1\}^n \mid \widehat{f}(\alpha) \neq 0\right\}$$

Possiamo allora fare due osservazioni. La prima è che, come è già stato detto, ciascuna delle funzioni $\varphi_\alpha$ non è altro che la funzione `PARITY` — a valori in $\{1,-1\}$ anziché in $\{0,1\}$ — delle variabili di ingresso da cui dipende; per il Teorema 2.8 abbiamo quindi che $\varphi_\alpha \in \widehat{LT}_2$ per ogni $\alpha \in S$. Inoltre, per il Teorema 2.7 è possibile calcolare ciascuna funzione $\varphi_\alpha$ tramite un circuito (di profondità 2) il cui threshold gate d'uscita è limitato. La seconda osservazione è che, essendo $f$ una funzione Booleana, la combinazione lineare (4.5) delle funzioni $\varphi_\alpha$ assumerà, per ogni configurazione di ingresso $\mathbf{x} \in \{0,1\}^n$, solamente i valori 0 e 1; vale pertanto la seguente uguaglianza:

$$f(\mathbf{x}) = step\left(\sum_{\alpha \in S} \widehat{f}(\alpha) \cdot \varphi_\alpha(\mathbf{x})\right) \qquad \forall\, \mathbf{x} \in \{0,1\}^n \tag{4.6}$$

Queste osservazioni ci consentono di costruire immediatamente un circuito a soglia $C$ di profondità 3 che calcola $f$: il circuito sarà formato da $|S|$ circuiti che calcolano le funzioni $\varphi_\alpha$ per ogni $\alpha \in S$, le cui uscite andranno in ingresso a un threshold gate (anch'esso limitato) a $|S|$ ingressi — ciascuno dei quali dotato del corrispondente peso $\widehat{f}(\alpha)$ — che calcola la combinazione lineare (4.5). Per la prima osservazione fatta, il secondo strato del circuito $C$ sarà formato interamente da threshold gate limitati; è pertanto possibile applicare l'omonima tecnica (si veda la Figura 2.2) per abbassare a 2 la profondità di $C$.

È immediato verificare che il circuito così ottenuto avrà in generale dimensione esponenziale, perché in generale $|S|$ crescerà in maniera esponenziale. Tuttavia, nei casi in cui $|S|$ cresce in maniera al più polinomiale, il circuito risultante dimostra che $f \in LT_2$; se poi anche i pesi interi del neurone d'uscita — ottenuti applicando su quest'ultimo il Teorema 1.1 a partire dai pesi reali $\widehat{f}(\alpha)$ — sono al più polinomiali, possiamo concludere che $f \in \widehat{LT}_2$.

Per quanto è stato detto nel Capitolo 3, affinché valga l'uguaglianza (4.6) non è necessario che la combinazione lineare delle funzioni $\varphi_\alpha$ valga esattamente 0 o 1 per ogni configurazione $\mathbf{x} \in \{0,1\}^n$ di ingresso. In particolare, la suddetta uguaglianza vale anche qualora la funzione $f$ sia $W_n$-approssimabile, nel senso della Definizione 3.3, dove con $W_n$ si intende la base di Walsh per lo spazio vettoriale $\Re^{2^n}$.

Se consideriamo le funzioni Booleane definite nel dominio $\{1,-1\}$ anziché in $\{0,1\}$, questo significa che l'uguaglianza (4.6) vale anche qualora $f$ (o, equivalentemente, la sua rappresentazione polinomiale) sia approssimata da un polinomio *sparso*, ovvero da un polinomio costituito da un numero al più polinomiale di monomi. Sarebbe pertanto di estrema utilità trovare una condizione sulla funzione $f$ che sia sufficiente ad assicurare tale approssimabilità.

Nel teorema che segue, viene dimostrato che una tale condizione sufficiente esiste: se la $L_1$-norma spettrale della funzione $f$ è al più polinomiale, esiste sicuramente un polinomio sparso che la approssima nel senso della Definizione 3.3.

**Teorema 4.6.** *Sia $f \in BIN_n$ una funzione tale che $\|f\|_1 \leq n^c$ per una costante $c$ fissata. Allora, per ogni $k > 0$, esiste un polinomio:*

$$F(\mathbf{x}) = \frac{1}{N} \sum_{\alpha \in S} w_\alpha \cdot X^\alpha$$

*con tutti i coefficienti $w_\alpha$ ed $N$ interi polinomiali, e in cui la cardinalità dell'insieme $S \subset \{0,1\}^n$ è polinomiale, tale che:*

$$\big|F(\mathbf{x}) - f(\mathbf{x})\big| \leq n^{-k} \qquad \forall\, \mathbf{x} \in \{1,-1\}^n$$

*Dimostrazione.* La dimostrazione è basata su un'argomentazione probabilistica; dimostriamo cioè che il polinomio sparso $F(\mathbf{x})$ esiste con probabilità maggiore di 0.

Consideriamo la rappresentazione polinomiale di $f$:

$$f(\mathbf{x}) = \sum_{\alpha \in \{0,1\}^n} \widehat{f}(\alpha) \cdot X^\alpha$$

e indichiamo con $L_1$ la norma spettrale di $f$:

$$L_1 = \|f\|_1 = \sum_{\alpha \in \{0,1\}^n} |\widehat{f}(\alpha)|$$

Sia $\lceil L_1 \rceil$ il più piccolo intero maggiore o uguale ad $L_1$; si noti che $\lceil L_1 \rceil < \|f\|_1 + 1$, e quindi $\lceil L_1 \rceil$ è al più polinomiale.

Posto:

$$p_\alpha = \frac{|\widehat{f}(\alpha)|}{\lceil L_1 \rceil} \qquad \forall\, \alpha \in \{0,1\}^n$$

definiamo, per $i = 1, 2, \ldots, N$ (dove $N$ viene determinato più avanti nel corso della dimostrazione), le variabili casuali $Z_i(\mathbf{x})$, tutte distribuite in modo identico e tutte indipendenti fra loro, nel modo seguente:

$$Z_i(\mathbf{x}) = \begin{cases} sign\big(\widehat{f}(\alpha)\big) \cdot X^\alpha & \text{con probabilità } p_\alpha, \quad \forall\, \alpha \in \{0,1\}^n \\ 0 & \text{con probabilità } 1 - \sum_{\alpha \in \{0,1\}^n} p_\alpha \end{cases}$$

Si osservi che il valore atteso di $Z_i(\mathbf{x})$ è:

$$E\big[Z_i(\mathbf{x})\big] = \sum_{\alpha \in \{0,1\}^n} p_\alpha \cdot sign\big(\widehat{f}(\alpha)\big) \cdot X^\alpha = \\ = \sum_{\alpha \in \{0,1\}^n} \frac{\widehat{f}(\alpha) \cdot X^\alpha}{\lceil L_1 \rceil} = \frac{f(\mathbf{x})}{\lceil L_1 \rceil}$$

mentre la varianza è:

$$Var\big[Z_i(\mathbf{x})\big] = E\big[Z_i^2(\mathbf{x})\big] - E^2\big[Z_i(\mathbf{x})\big] = \\ = \frac{\|f\|_1}{\lceil L_1 \rceil} - \frac{1}{\lceil L_1 \rceil^2} = \Omega(1)$$

Pertanto, secondo il Teorema Centrale della Statistica, per $n$ sufficientemente grande abbiamo:

$$\Pr\left[\left|\frac{1}{N}\sum_{i=1}^{N} Z_i(\mathbf{x}) - \frac{f(\mathbf{x})}{\lceil L_1 \rceil}\right| \geq \frac{\sqrt{n}}{\sqrt{N}}\right] = O(e^{-n})$$

Posto $N = \lceil L_1 \rceil^2 n^{2k+1}$, che è una quantità polinomiale, otteniamo:

$$\Pr\left[\left|\frac{1}{\lceil L_1 \rceil^2 n^{2k+1}}\sum_{i=1}^{N} Z_i(\mathbf{x}) - \frac{f(\mathbf{x})}{\lceil L_1 \rceil}\right| \geq \frac{\sqrt{n}}{\lceil L_1 \rceil n^{k+\frac{1}{2}}}\right] = \\ = \Pr\left[\left|\frac{1}{\lceil L_1 \rceil n^{2k+1}}\sum_{i=1}^{N} Z_i(\mathbf{x}) - f(\mathbf{x})\right| \geq n^{-k}\right] = \\ = \Pr\left[\left|\frac{\lceil L_1 \rceil}{N}\sum_{i=1}^{N} Z_i(\mathbf{x}) - f(\mathbf{x})\right| \geq n^{-k}\right] = O(e^{-n})$$

Se ora poniamo:

$$F(\mathbf{x}) = \frac{\lceil L_1 \rceil}{N}\sum_{i=1}^{N} Z_i(\mathbf{x})$$

otteniamo:

$$\Pr\Big[\forall\, \mathbf{x} \in \{1,-1\}^n \quad \big|F(\mathbf{x}) - f(\mathbf{x})\big| \leq n^{-k}\Big] = \\ = 1 - \Pr\Big[\exists\, \mathbf{x} \in \{1,-1\}^n \quad \big|F(\mathbf{x}) - f(\mathbf{x})\big| > n^{-k}\Big] \geq \\ \geq 1 - 2^n \cdot O(e^{-n}) = 1 - o(1)$$

Pertanto, per ogni $k > 0$ esiste almeno un polinomio $F(\mathbf{x})$ che approssima $f$ nel senso della Definizione 3.3. Possiamo riscrivere $F(\mathbf{x})$ come:

$$F(\mathbf{x}) = \frac{1}{N}\sum_{\alpha \in S} w_\alpha \cdot X^\alpha$$

dove $|S| \leq N$ è polinomiale, e $w_\alpha = \lceil L_1 \rceil \cdot sign\big(\widehat{f}(\alpha)\big)$ è un intero polinomiale per ogni $\alpha \in S$. ∎

Il Teorema 4.6 può essere parafrasato dicendo che ogni funzione Booleana avente norma spettrale polinomiale è 1-approssimabile, ovvero appartiene alla

classe $APP_1$ (si veda il Capitolo 3). Infatti, ogni monomio $X^\alpha$ del polinomio sparso $F(\mathbf{x})$ è una funzione Booleana simmetrica, e in quanto tale può essere scritto come una sommatoria di un numero polinomiale di funzioni che appartengono a $\widehat{LT}_1$ (si veda la dimostrazione del Teorema 2.7); riscrivendo in $F(\mathbf{x})$ la combinazione lineare delle sommatorie sotto forma di semplice combinazione lineare si ottiene che, per ogni $k > 0$, esistono una funzione multi-uscita polinomiale $(t_1, t_2, \ldots, t_s) \in \widehat{LT}_1$ e degli interi polinomiali $w_1, w_2, \ldots, w_s$ ed $N$ per i quali la combinazione lineare:

$$F(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{s} w_i t_i(\mathbf{x})$$

è tale che:

$$\big|F(\mathbf{x}) - f(\mathbf{x})\big| \leq n^{-k} \qquad \forall\, \mathbf{x} \in \{1, -1\}^n$$

Per la Definizione 3.3, questo significa che $f$ è 1-approssimabile.

## 4.4 La funzione `CONFRONTO`

Mostriamo ora un esempio di applicazione del Teorema 4.6. L'esempio viene reso interessante dal fatto che la norma spettrale della funzione interessata non viene calcolata esplicitamente con i metodi visti finora, bensì viene ricavata da una semplice relazione ricorsiva.

Si consideri la funzione `CONFRONTO`, introdotta nella Definizione 2.20. Per convenienza, si supponga che il dominio di definizione della funzione sia l'insieme $\{1, -1\}$, anziché $\{0, 1\}$ come nella citata definizione.

Se $C_n$ è la rappresentazione polinomiale corrispondente alla funzione `CONFRONTO`$(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$, è abbastanza facile rendersi conto che possiamo scrivere la seguente relazione ricorsiva:

$$\begin{cases} C_n = \frac{x_n - y_n}{2} + \frac{1 + x_n y_n}{2}\, C_{n-1} \\ C_1 = \frac{x_1 - y_1}{2} + \frac{1 + x_1 y_1}{2} \end{cases} \tag{4.7}$$

Infatti, al variare di $x_n$ e di $y_n$ in $\{1, -1\}$, abbiamo che le quantità coinvolte nell'equazione (4.7) assumono i valori riportati nella seguente tabella:

| $x_n$ | $y_n$ | $C_n$ | $\frac{x_n - y_n}{2}$ | $\frac{1 + x_n y_n}{2}$ |
|---|---|---|---|---|
| 1 | 1 | $C_{n-1}$ | 0 | 1 |
| 1 | −1 | 1 | 1 | 0 |
| −1 | 1 | −1 | −1 | 0 |
| −1 | −1 | $C_{n-1}$ | 0 | 1 |

Possiamo allora dimostrare il seguente teorema, che conferma quanto già riportato nella tabella in fondo al Capitolo 2.

**Teorema 4.7.** $\texttt{CONFRONTO} \in \widehat{LT}_2$.

*Dimostrazione.* Dall'equazione (4.7) si può vedere che:

$$\|\texttt{CONFRONTO}_1\|_1 = 2$$
$$\|\texttt{CONFRONTO}_n\|_1 = \|\texttt{CONFRONTO}_{n-1}\|_1 + 1$$

da cui segue immediatamente che $\|\texttt{CONFRONTO}_n\|_1 = n + 1$. Per il Teorema 4.6, $\texttt{CONFRONTO} \in \widehat{LT}_2$. ∎

Un discorso analogo può essere fatto per quanto riguarda la funzione Booleana che calcola il bit più significativo della funzione multi-uscita ADDIZIONE, introdotta nella Definizione 2.15. Inoltre, anche per le funzioni che calcolano gli altri $n$ bit della funzione multi-uscita ADDIZIONE è possibile applicare le tecniche dell'analisi spettrale, arrivando a determinare che $\texttt{ADDIZIONE} \in \widehat{LT}_2$, come riportato nella tabella in fondo al Capitolo 2.

## 4.5 Interpretazione geometrica

Finora abbiamo considerato la trasformata di Fourier unicamente come un metodo di rappresentazione polinomiale per le funzioni Booleane. Vediamo ora come una semplice interpretazione geometrica delle funzioni calcolate dai circuiti a soglia ci consenta di vedere la trasformata di Fourier come una trasformazione lineare fra spazi vettoriali.

Come viene mostrato in un articolo di Roychowdhury, Siu, Orlitsky e Kailath [13], l'interpretazione geometrica delle funzioni calcolate dai circuiti a soglia permette molto spesso di semplificare e/o estendere i risultati riguardanti la capacità computazionale degli stessi circuiti. Questo avviene, fondamentalmente, perché le tecniche derivate dall'Algebra Lineare portano a dimostrazioni più immediate di quelle ottenibili manipolando le rappresentazioni polinomiali delle funzioni coinvolte.

### 4.5.1 Rappresentazione vettoriale

Data una funzione Booleana $f \in BIN_n$, si consideri il vettore $\mathbf{v}_n \in \Re^{2^n}$ introdotto nella Definizione 2.11. Possiamo immaginare che ciascuna delle $2^n$ dimensioni di $\Re^{2^n}$ corrisponda ad una configurazione d'ingresso per $f$, e che la coordinata del vettore $\mathbf{v}_n$ in una data dimensione sia il valore di $f$ per la configurazione d'ingresso corrispondente. Detto ciò, possiamo dare la seguente definizione.

**Definizione 4.5.** Chiamiamo $\mathbf{v}_n$ la *rappresentazione vettoriale* di $f$. Commettendo un abuso di notazione, d'ora in poi indicheremo col simbolo $f$ sia la funzione Booleana che la sua rappresentazione vettoriale. ∎

Per capire il motivo per cui la rappresentazione vettoriale delle funzioni Booleane è utile per analizzare i circuiti a soglia, si consideri quanto segue. Sia $C$ un circuito a soglia ad $n$ ingressi, e sia $f \in BIN_n$ la funzione Booleana da esso calcolata. Chiamato $T$ il threshold gate d'uscita di $C$, si indichino con $f_1, f_2, \ldots, f_S \in BIN_n$ le funzioni calcolate dai sottocircuiti le cui uscite vanno in ingresso a $T$. Siano infine $w_1, w_2, \ldots, w_S$ i pesi associati agli ingressi di $T$; supponendo nulla la soglia di $T$ (come è stato detto nel Capitolo 1, è sempre possibile ricondursi a questa situazione, aggiungendo eventualmente un ingresso dotato di peso opportuno), in corrispondenza di ciascuna configurazione di ingresso $\mathbf{x} \in \{0,1\}^n$ vale la seguente uguaglianza:

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{S} w_i f_i(\mathbf{x})\right)$$

L'insieme di tali uguaglianze, ottenute facendo variare $\mathbf{x}$ in $\{0,1\}^n$, può essere scritto in maniera compatta facendo uso delle rappresentazioni vettoriali delle funzioni coinvolte:

$$f = sign\left(\sum_{i=1}^{S} w_i f_i\right) \tag{4.8}$$

avendo definito la funzione $sign : \Re^{2^n} \rightarrow \{-1,1\}^{2^n}$ come l'applicazione della funzione $sign : \Re \rightarrow \{-1,1\}$ (equazione (1.3)) ai singoli elementi del vettore $\sum_{i=1}^{S} w_i f_i$. Si noti che, per come abbiamo definito la funzione $sign : \Re \rightarrow \{-1,1\}$, si ha che $sign(0) = 1$; se ciò potesse creare confusione, si osservi che il Teorema 1.15 ci autorizza ad assumere che le componenti del vettore $\sum_{i=1}^{S} w_i f_i$ siano tutte diverse da 0. Con queste premesse, possiamo dare la seguente definizione.

**Definizione 4.6.** Una funzione $f \in BIN_n$ è una *funzione a soglia*, rispetto alle funzioni $f_1, f_2, \ldots, f_S \in BIN_n$, se esistono dei numeri interi $w_1, w_2, \ldots, w_S$ per i quali vale l'uguaglianza (4.8). ∎

Equivalentemente, $f$ è una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$ se esiste un threshold gate che, ricevute in ingresso le uscite dei circuiti che calcolano le funzioni $f_1, f_2, \ldots, f_S$, calcola $f$.

Analogamente a quanto è stato fatto nel Capitolo 1 a proposito delle funzioni threshold, risulta conveniente riscrivere l'equazione (4.8) in forma ancora più compatta:

$$f = sign(Y\mathbf{w})$$

dove:

$$Y = \begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$$

è la *matrice d'ingresso* $2^n \times S$ le cui colonne sono le funzioni di ingresso, e:

$$\mathbf{w} = (w_1, w_2, \ldots, w_S)^T \in \Re^S$$

è il vettore colonna dei pesi. Senza perdere in generalità, assumiamo che il rango delle colonne di $Y$ sia massimo.

Diamo ora la seguente definizione.

**Definizione 4.7.** Definiamo la relazione d'equivalenza $\sim \subseteq \Re^n \times \Re^n$ come segue; per ogni coppia $\mathbf{x}, \mathbf{y}$ di vettori in $\Re^n$:

$$\mathbf{x} \sim \mathbf{y} \iff sign\big((\mathbf{x})_i\big) = sign\big((\mathbf{y})_i\big) \qquad \forall\, i \in \{1, 2, \ldots, n\}$$

Chiamiamo *ortanti* di $\Re^n$ le classi di equivalenza della relazione $\sim$. Dato un ortante $\mathcal{O}$ di $\Re^n$, definiamo l'*interno* di $\mathcal{O}$ come il sottoinsieme di $\mathcal{O}$ costituito dai vettori le cui componenti sono tutte diverse da 0. ∎

Dal punto di vista geometrico, ciascuna funzione $f \in BIN_n$ — essendo un vettore in $\Re^{2^n}$ le cui componenti valgono 1 o $-1$ — individua (l'interno di) un ortante di $\Re^{2^n}$. In questa interpretazione, $f$ è una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$ se e solo se esiste una combinazione lineare $Y\mathbf{w} = \sum_{i=1}^{S} w_i f_i$ che giace nell'ortante individuato da $f$, ovvero tale che $Y\mathbf{w} \sim f$. Questa semplice interpretazione geometrica forma la base su cui poggiano molti risultati riguardanti le funzioni a soglia.

Risulta a questo punto conveniente dare la seguente definizione.

**Definizione 4.8.** La *correlazione* (normalizzata) di due funzioni $f, f' \in BIN_n$ è il numero reale:

$$C_{ff'} = \frac{1}{2^n} f^T f' = \langle f, f' \rangle$$

Le due funzioni sono dette *non correlate* o ortogonali se $C_{ff'} = 0$.

Il *vettore di correlazione* (normalizzato) di una funzione $f \in BIN_n$ con le funzioni $f_1, f_2, \ldots, f_S \in BIN_n$ è il vettore di $\Re^S$ definito come:

$$C_{fY} = \frac{1}{2^n} f^T Y = \big(C_{ff_1}, C_{ff_2}, \ldots, C_{ff_S}\big)^T$$

La funzione $f$ è *non correlata* o ortogonale alle funzioni $f_1, f_2, \ldots, f_S$ se $C_{fY} = \mathbf{0}_S$. ∎

Esiste una semplice relazione che lega la distanza di Hamming $d_H(f, f')$ (si veda la Definizione 1.2) tra due funzioni Booleane e la correlazione $C_{ff'}$ tra le stesse:

$$C_{ff'} = 1 - \frac{d_H(f, f')}{2^{n-1}} \tag{4.9}$$

Questo significa che la correlazione può essere interpretata anche come una misura della "somiglianza" tra le due funzioni.

Si osservi che se due vettori giacciono nello stesso ortante il loro prodotto interno è sicuramente non negativo. Se poi uno dei due vettori appartiene all'interno dell'ortante, e l'altro non è il vettore nullo, allora il loro prodotto interno è positivo. Queste semplici osservazioni ci consentono di dimostrare il seguente teorema.

**Teorema 4.8.** *Siano $f, f_1, f_2, \ldots, f_S \in BIN_n$. Se $f$ è ortogonale ad ogni funzione $f_1, f_2, \ldots, f_S$, allora $f$ non è una funzione a soglia rispetto alle stesse.*

*Dimostrazione.* Sia $Y = \begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$ la matrice di ingresso, e sia $\mathcal{O}$ l'ortante di $\Re^{2^n}$ individuato da $f$.

Se $f$ è ortogonale ad ogni funzione $f_1, f_2, \ldots, f_S$, allora $f$ è ortogonale ad ogni combinazione lineare $Y\mathbf{w}$ di tali funzioni (perchè non appartiene al sottospazio da esse generato), e quindi il prodotto interno calcolato tra $f$ e il vettore $Y\mathbf{w}$ è sempre nullo.

D'altra parte, per il Teorema 1.15, il vettore $\mathbf{w}$ può sempre essere scelto in modo tale che il vettore $Y\mathbf{w}$ non sia nullo; dato che $f$ appartiene all'interno di $\mathcal{O}$, se per assurdo il vettore $Y\mathbf{w}$ appartenesse ad $\mathcal{O}$ il prodotto interno tra $f$ ed $Y\mathbf{w}$ dovrebbe essere positivo. Pertanto, il vettore $Y\mathbf{w}$ non può appartenere ad $\mathcal{O}$, ovvero $f$ non può essere una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$. ∎

Siamo ora in grado di mostrare perché la rappresentazione vettoriale delle funzioni Booleane consente di generalizzare i risultati ottenuti con le tecniche spettrali. Preso un qualsiasi insieme di funzioni $f_1, f_2, \ldots, f_S \in BIN_n$, si supponga che il rango delle colonne della matrice $Y = \begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$ sia massimo. Dall'Algebra Lineare sappiamo che ogni funzione $f \in \Re^{2^n}$ può essere espressa come:

$$f = Y\beta + Z \tag{4.10}$$

dove $Z$ è un vettore in $\Re^{2^n}$ tale che $Z^T Y = \mathbf{0}_S$ (cioè $Z$ è ortogonale ad ogni funzione $f_1, f_2, \ldots, f_S$). Il vettore colonna $\beta = (\beta_1, \beta_2, \ldots, \beta_S)^T \in \Re^S$ viene comunemente chiamato lo *spettro generalizzato di $f$* rispetto alle funzioni $f_1, f_2, \ldots, f_S$; è facile verificare che $\beta$ può essere ottenuto nel modo seguente:

$$\beta = (Y^T Y)^{-1} Y^T f = 2^n (Y^T Y)^{-1} C_{fY}$$

dove $C_{fY}$ è il vettore di correlazione definito precedentemente. Dal punto di vista geometrico, $Y\beta$ rappresenta la proiezione ortogonale di $f$ sul sottospazio generato dai vettori di ingresso $f_1, f_2, \ldots, f_S$, mentre $Z$ è il vettore d'errore ortogonale a tale sottospazio. Si noti che se le funzioni $f_1, f_2, \ldots, f_S$ sono ortogonali allora vale $Y^T Y = 2^n I_S$, da cui si ottiene $2^n (Y^T Y)^{-1} = I_S$, e quindi $\beta = C_{fY}$; in particolare, se scegliamo le funzioni $f_1, f_2, \ldots, f_S$ all'interno della base di Walsh $W_n$ abbiamo:

$$\begin{aligned} \beta = \big(\beta_1, \beta_2, \ldots, \beta_S\big) &= \big(C_{ff_1}, C_{ff_2}, \ldots, C_{ff_S}\big) = \\ &= \big(\langle f, f_1\rangle, \langle f, f_2\rangle, \ldots, \langle f, f_S\rangle\big) \end{aligned}$$

ovvero le componenti $\beta_1, \beta_2, \ldots, \beta_S$ del vettore $\beta$ sono i corrispondenti coefficienti di Fourier. Se poi costruiamo la matrice di ingresso utilizzando *tutte* le funzioni

$\varphi_\alpha$ della base di Walsh $W_n$, il termine $Z$ dell'equazione (4.10) sarà il vettore nullo $\mathbf{0}_{2^n}$, e la decomposizione di $f$ assumerà la seguente forma:

$$f = Y\mathcal{F}_n(f)$$

dove $\mathcal{F}_n(f)$ rappresenta lo spettro (di Fourier) di $f$. Confrontando l'equazione appena scritta con l'equazione (4.4), ci si rende conto che le matrici di Sylvester $H_{2^n}$ sono le matrici le cui colonne rappresentano tutte le possibili funzioni $\varphi_\alpha$ della base di Walsh; come si è visto nel Teorema 4.2, le inverse di tali matrici consentono di calcolare lo spettro di Fourier di $f$, e rappresentano quindi le trasformate di Fourier, che possiamo pertanto considerare come delle trasformazioni lineari dello spazio vettoriale $\Re^{2^n}$ in se stesso.

Poiché i risultati ottenuti manipolando la rappresentazione polinomiale di una funzione Booleana $f$ corrispondono alla decomposizione (4.10) di $f$ in cui la matrice di ingresso $Y$ è formata da funzioni prese all'interno della base di Walsh $W_n$, accade molto spesso che di tali risultati si riesca a dare una dimostrazione geometrica, utilizzando le tecniche dell'Algebra Lineare, a partire dalla rappresentazione vettoriale delle funzioni coinvolte. In tal caso, se accade che la dimostrazione geometrica non richiede che le funzioni $f_1, f_2, \ldots, f_S$ della matrice di ingresso siano prese dalla base di Walsh, o addirittura se la dimostrazione non dipende dal fatto che le suddette funzioni siano ortogonali, è molto spesso possibile estendere il risultato a matrici di ingresso qualsiasi, utilizzando il corrispondente spettro generalizzato al posto dello spettro di Fourier. Un esempio in tale direzione viene riportato alla fine del capitolo, in cui si accenna all'estensione del Teorema 4.6.

### 4.5.2 Unicità della correlazione

Abbiamo visto che è possibile interpretare una funzione Booleana di $n$ variabili come un vettore a $2^n$ componenti. Date allora due funzioni $f, f' \in BIN_n$ qualsiasi, dalla Definizione 1.2 segue che la loro distanza di Hamming $d_H(f, f')$ può assumere qualsiasi valore *intero* compreso tra 0 e $2^n$; da questa semplice osservazione, e dalla relazione che intercorre tra la distanza di Hamming e la correlazione $C_{ff'}$ delle due funzioni (equazione (4.9)), segue che $C_{ff'}$ può assumere esattamente $2^n + 1$ valori *razionali* compresi tra $-1$ e 1; si noti che un valore di $C_{ff'}$ prossimo a 1 corrisponde ad un valore di $d_H(f, f')$ prossimo a 0 (le funzioni sono cioè molto simili), mentre un valore di $C_{ff'}$ prossimo a $-1$ corrisponde ad un valore di $d_H(f, f')$ prossimo a $2^n$ (le funzioni sono molto diverse tra loro).

Le considerazioni fatte a proposito del numero di valori distinti che può assumere la correlazione $C_{ff'}$ possono essere immediatamente estese al vettore di correlazione: date le funzioni $f_1, f_2, \ldots, f_S \in BIN_n$ di ingresso a un threshold gate, e indicata come al solito con $Y = \begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$ la matrice di ingresso, il vettore di correlazione $C_{fY} = \left(C_{ff_1}, C_{ff_2}, \ldots, C_{ff_S}\right)^T$ potrà assumere al più $(2^n + 1)^S$ differenti valori per ogni funzione Booleana $f$.

Poiché le funzioni Booleane ad $n$ ingressi sono in tutto $2^{2^n}$, molte di queste condivideranno necessariamente lo stesso vettore di correlazione rispetto alle funzioni $f_1, f_2, \ldots, f_S$. Nonostante ciò, il prossimo teorema dimostra che una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$ non condivide il proprio vettore di correlazione con nessun'altra funzione Booleana.

**Teorema 4.9.** *Sia $f \in BIN_n$ una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S \in BIN_n$. Indicata con $Y$ la matrice di ingresso $\big[f_1\ f_2\ \ldots\ f_S\big]$, per ogni funzione $g \in BIN_n$ diversa da $f$ vale:*

$$C_{fY} \neq C_{gY}$$

*Dimostrazione.* Se $f \in BIN_n$ è una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S \in BIN_n$, esiste un vettore $\mathbf{w} \in \Re^S$ tale che $f = sign(Y\mathbf{w})$.

Sia $g \in BIN_n$ una funzione diversa da $f$, e si consideri la funzione $(f - g) \in \Re^{2^n}$; come è facile verificare, per ogni $i = 1, 2, \ldots, 2^n$ vale:

$$(f-g)_i = \begin{cases} 0 & \text{se } (f)_i = (g)_i \\ 2 \cdot (f)_i & \text{se } (f)_i \neq (g)_i \end{cases}$$

Pertanto, in corrispondenza di ciascun indice $i \in \{1, 2, \ldots, 2^n\}$ per il quale $(f)_i \neq (g)_i$, valgono le seguenti uguaglianze:

$$sign\big((f-g)_i\big) = sign\big((f)_i\big) = sign\big((Y\mathbf{w})_i\big)$$

Essendo $f \neq g$, esiste sicuramente un indice $i \in \{1, 2, \ldots, 2^n\}$ tale che $(f)_i \neq (g)_i$; potendo supporre, per il Teorema 1.15, che sia $(Y\mathbf{w})_i \neq 0$ per ogni indice $i$, avremo allora che il prodotto interno tra il vettore $(f - g)$ e il vettore $Y\mathbf{w}$ è sicuramente positivo:

$$(f-g)^T \cdot (Y\mathbf{w}) = \sum_{i=1}^{2^n} (f-g)_i \cdot (Y\mathbf{w})_i > 0$$

da cui segue immediatamente che $f^T Y \neq g^T Y$, ovvero che $C_{fY} \neq C_{gY}$. ∎

La dimostrazione del teorema precedente, che per ovvie ragioni è noto col nome di Teorema di Unicità (del vettore di correlazione per le funzioni a soglia) ha una semplice interpretazione geometrica. Se $f$ è una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$, allora esiste un vettore di pesi $\mathbf{w}$ tale che la combinazione lineare $Y\mathbf{w}$ delle funzioni $f_1, f_2, \ldots, f_S$ giace nell'ortante di $\Re^{2^n}$ il cui interno è individuato da $f$. Ma, per ogni funzione $g \neq f$, anche il vettore non nullo $(f - g)$ giace nel medesimo ortante, e quindi il prodotto interno tra i vettori $(f-g)$ e $Y\mathbf{w}$ è sicuramente positivo. Da ciò segue immediatamente che $f^T Y \neq g^T Y$, ovvero che $C_{fY} \neq C_{gY}$.

Se si prendono le funzioni $f_1, f_2, \ldots, f_S$ tra le funzioni $\varphi_\alpha$ che compongono la base di Walsh $W_n$, allora, come è stato detto, le componenti del vettore di

correlazione $C_{fY}$ coincidono con i corrispondenti coefficienti di Fourier di $f$; pertanto, se $f$ è una funzione a soglia delle funzioni $\varphi_\alpha \in W_n$, il Teorema di Unicità implica che l'insieme dei coefficienti di Fourier di $f$ è unico.

Come semplice conseguenza del Teorema di Unicità deriviamo un limite superiore al numero di funzioni a soglia di un qualsiasi insieme di funzioni di ingresso.

**Teorema 4.10.** *Esistono al più* $(2^n+1)^S$ *funzioni a soglia rispetto a un qualsiasi insieme* $f_1, f_2, \ldots, f_S$ *di* $S$ *funzioni di ingresso.*

*Dimostrazione.* La dimostrazione segue immediatamente dal Teorema di Unicità del vettore di correlazione osservando che, per qualsiasi insieme di $S$ funzioni di ingresso, esistono al più $(2^n + 1)^S$ diversi vettori di correlazione. ∎

Viene a questo punto naturale chiedersi se per caso valga anche l'inverso del Teorema di Unicità, ovvero se il fatto che una data funzione Booleana $f$ non condivida il suo vettore di correlazione — calcolato come sempre rispetto ad alcune funzioni $f_1, f_2, \ldots, f_S$ prefissate — con nessun'altra funzione Booleana implichi che $f$ è una funzione a soglia rispetto a $f_1, f_2, \ldots, f_S$. La risposta è negativa, come viene dimostrato nel seguente teorema.

**Teorema 4.11.** *Siano* $f, f_1, f_2, \ldots, f_S \in BIN_n$. *Indicata con* $Y$ *la matrice di ingresso* $\begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$, *si supponga che valga* $C_{fY} \neq C_{gY}$ *per ogni funzione* $g \in BIN_n$ *diversa da* $f$. *Allora* $f$ *non è necessariamente una funzione a soglia rispetto alle funzioni* $f_1, f_2, \ldots, f_S$.

*Dimostrazione.* Dimostriamo il teorema attraverso un esempio. Sia $W_n$ la base di Walsh per lo spazio vettoriale $\Re^{2^n}$; si ponga $f = \varphi_{\mathbf{1}_n}$, e siano $f_1, f_2, \ldots, f_S$ le restanti funzioni $\varphi_\alpha$ per $\alpha \neq \mathbf{1}_n$.

Chiaramente $S = 2^n - 1$, ed $f$ è ortogonale a ciascuna delle funzioni $f_1, f_2, \ldots, f_S$, così che $C_{fY} = \frac{1}{2^n} f^T Y = \mathbf{0}_S$.

D'altra parte, essendo $f$ l'unica funzione di $\Re^{2^n}$ ortogonale ad ogni colonna di $Y$, è semplice verificare che $C_{gY} = \frac{1}{2^n} g^T Y \neq 0$, così che $C_{gY} \neq C_{fY}$ per ogni funzione Booleana $g$ diversa da $f$.

Il Teorema resta allora dimostrato osservando che, essendo $f$ ortogonale a tutte le funzioni $f_1, f_2, \ldots, f_S$, per il Teorema 4.8 $f$ non può essere una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$. ∎

### 4.5.3 Un limite inferiore al numero delle funzioni di ingresso

Mostriamo ora come lo spettro generalizzato di una funzione $f$ rispetto alle funzioni $f_1, f_2, \ldots, f_S$, introdotto nell'equazione (4.10), possa essere utilizzato per derivare un limite inferiore al numero di funzioni di ingresso richieste da un threshold gate per calcolare una data funzione.

A tale scopo, diamo anzitutto la seguente definizione.

**Definizione 4.9.** Si denoti con $\widetilde{\beta}$ il coefficiente *massimale* dello spettro generalizzato $\beta = (\beta_1, \beta_2, \ldots, \beta_S)$:

$$\widetilde{\beta} = \max_{1 \leq i \leq S} |\beta_i|$$

Se le funzioni $f_1, f_2, \ldots, f_S$ sono mutuamente ortogonali, abbiamo visto che $\beta = C_{fY}$; in tal caso denoteremo con $\widetilde{C} = \max_{1 \leq i \leq S} C_{ff_i}$, anziché con $\widetilde{\beta}$, il coefficiente massimale dello spettro generalizzato. ∎

Il legame che intercorre tra il coefficiente massimale dello spettro generalizzato di una funzione Booleana $f$ e il numero di funzioni di ingresso necessarie affinché un threshold gate possa calcolare $f$ è dato dal seguente teorema.

**Teorema 4.12.** *Se $f \in BIN_n$ è una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S \in BIN_n$, ed $f$ non appartiene allo spazio generato da $f_1, f_2, \ldots, f_S$, allora:*

$$\sum_{i=1}^{S} |\beta_i| \geq 1$$

*da cui:*

$$S \geq \frac{1}{\widetilde{\beta}}$$

*Dimostrazione.* Indicata con $Y$ la matrice di ingresso $\begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$, si consideri la decomposizione $f = Y\beta + Z$, con $Z^T Y = \mathbf{0}_S$, come nell'equazione (4.10). Poiché $f$ non appartiene allo spazio generato da $f_1, f_2, \ldots, f_S$, possiamo supporre che $Z$ sia diverso dal vettore nullo $\mathbf{0}_{2^n}$.

Si supponga per assurdo che sia $\sum_{i=1}^{S} |\beta_i| < 1$. In tal caso si avrebbe:

$$(Y\beta)_j = \sum_{i=1}^{S} (f_i)_j \beta_i \leq \left| \sum_{i=1}^{S} (f_i)_j \beta_i \right| \leq \sum_{i=1}^{S} \left|(f_i)_j\right| \cdot \left|\beta_i\right| = \sum_{i=1}^{S} |\beta_i| < 1$$

per ogni $j = 1, 2, \ldots, 2^n$, e quindi:

$$sign(Z) = sign(f - Y\beta) = f$$

Questo significa che $Z$ apparterrebbe all'ortante di $\Re^{2^n}$ il cui interno è individuato da $f$. Essendo $f$ una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S$, esiste un vettore $\mathbf{w} \in \Re^S$ tale che anche la combinazione lineare $Y\mathbf{w}$ giace nell'ortante il cui interno è individuato da $f$; potendo supporre, per il Teorema 1.15, che il vettore $Y\mathbf{w}$ non abbia alcuna componente nulla, esso apparterrà all'interno dell'ortante, così che il prodotto interno tra $Z$ e $Y\mathbf{w}$ sarà sicuramente positivo.

Si giunge così ad una contraddizione, poiché da una parte $Z$ è ortogonale a tutte le funzioni $f_1, f_2, \ldots, f_S$ (da cui $Z^T Y = 0$), mentre dall'altra giace nello stesso ortante della combinazione lineare $Y\mathbf{w}$ di $f_1, f_2, \ldots, f_S$ (da cui $Z^T Y\mathbf{w} > 0$).

Pertanto si può concludere che vale $\sum_{i=1}^{S} |\beta_i| \geq 1$, da cui si ottiene:

$$1 \leq \sum_{i=1}^{S} |\beta_i| \leq \sum_{i=1}^{S} \widetilde{\beta} = \widetilde{\beta}\, S$$

ovvero:

$$S \geq \frac{1}{\widetilde{\beta}}$$

■

Come caso particolare del teorema precedente abbiamo che:

**Teorema 4.13.** *Se $f \in BIN_n$ è una funzione a soglia rispetto alle funzioni $f_1, f_2, \ldots, f_S \in BIN_n$ mutuamente ortogonali, allora $S \geq \frac{1}{\widetilde{C}}$.* ■

In particolare, il numero di funzioni $\varphi_\alpha$ in ingresso ad un threshold gate necessarie per calcolare una funzione $f$ è almeno l'inverso del più grande coefficiente di Fourier di $f$. Ad esempio, se tutti i coefficienti di Fourier di $f$ sono esponenzialmente piccoli allora sono necessarie un numero esponenziale di funzioni $\varphi_\alpha$ per calcolare $f$.

I risultati precedenti si applicano ai threshold gate senza alcuna restrizione sui pesi. Nel caso in cui i pesi siano al più polinomiali — e che le funzioni di ingresso siano mutuamente ortogonali — è possibile ottenere un risultato più forte.

**Teorema 4.14.** *Sia $f \in BIN_n$, e siano $f_1, f_2, \ldots, f_S \in BIN_n$ delle funzioni mutuamente ortogonali tali che:*

$$f = sign\left(\sum_{i=1}^{S} w_i f_i\right)$$

*per opportuni pesi $w_1, w_2, \ldots, w_S$ interi. Allora, posto $\widetilde{w} = \max_{1 \leq i \leq S} w_i$ e $\widetilde{C} = \max_{1 \leq i \leq S} C_{f f_i}$, vale:*

$$S \geq \frac{1}{\widetilde{w}\,\widetilde{C}}$$

*Dimostrazione.* Indicata con $Y$ la matrice di ingresso $\begin{bmatrix} f_1 & f_2 & \ldots & f_S \end{bmatrix}$, e con $\mathbf{w}$ il vettore dei pesi $(w_1, w_2, \ldots, w_S) \in \boldsymbol{Z}^S$, per ipotesi vale:

$$f = sign(Y\mathbf{w})$$

Questo significa che le componenti corrispondenti dei due vettori $f$ e $Y\mathbf{w}$ hanno lo stesso segno. Inoltre, dato che tutti i pesi sono interi, $Y\mathbf{w}$ è un vettore a componenti intere, per cui $|(Y\mathbf{w})_i| \geq 1$; unendo le due osservazioni si può scrivere:

$$2^n \leq f^T(Y\mathbf{w}) = (f^T Y)\mathbf{w} = 2^n\, C_{fY}^T \mathbf{w} = 2^n \sum_{i=1}^{S} w_i\, C_{f f_i}$$

da cui si ottiene:

$$1 \leq \sum_{i=1}^{S} w_i \, C_{ff_i} \leq \left| \sum_{i=1}^{S} w_i \, C_{ff_i} \right| \leq \sum_{i=1}^{S} |w_i| \cdot |C_{ff_i}| \leq \sum_{i=1}^{S} \widetilde{w} \, \widetilde{C} = S \, \widetilde{C} \, \widetilde{w}$$

e infine:

$$S \geq \frac{1}{\widetilde{w} \, \widetilde{C}}$$

■

In particolare, per ogni insieme di funzioni di ingresso $f_1, f_2, \ldots, f_S$ mutuamente ortogonali, se le correlazioni tra una data funzione Booleana $f$ e le funzioni $f_1, f_2, \ldots, f_S$ sono esponenzialmente piccole allora è sicuramente necessario un numero esponenziale di funzioni di ingresso per poter calcolare $f$ come funzione a soglia — con pesi al più polinomiali — rispetto a $f_1, f_2, \ldots, f_S$.

### 4.5.4 $L_1$-norme spettrali generalizzate

Con il Teorema 4.6, abbiamo dimostrato che se la norma spettrale $\|f\|_1$ di una funzione Booleana $f$ è al più polinomiale, allora $f$ può essere espressa come funzione a soglia (con pesi polinomiali) di un numero al più polinomiale di monomi. Abbiamo inoltre visto che, in tal caso, è possibile applicare la tecnica dei threshold gate limitati, arrivando ad affermare che $f \in \widehat{LT}_2$; meglio ancora, abbiamo visto che è possibile scrivere ciascun monomio come somma di un numero al più polinomiale di funzioni appartenenti a $\widehat{LT}_1$ e che, inserendo direttamente tali somme all'interno della rappresentazione polinomiale di $f$, si arriva a dimostrare che $f \in APP_1$.

Presentiamo ora una generalizzazione del Teorema 4.6 in termini di coefficienti spettrali generalizzati. Come per il Teorema 4.6, supponiamo che le funzioni Booleane considerate siano definite nel dominio $\{1, -1\}$.

Sia $\mathcal{G} = \{g_1, g_2, \ldots, g_{2^n}\}$ una qualsiasi base (non necessariamente ortogonale) dello spazio vettoriale $\Re^{2^n}$; come è stato dimostrato da Siu, Roychowdhury e Kailath [17], esiste sempre almeno una base siffatta. Ogni funzione $f \in \Re^{2^n}$ (e quindi anche ogni funzione $f \in BIN_n$) può allora essere espressa in un unico modo come:

$$f(\mathbf{x}) = \sum_{i=1}^{2^n} \beta_i g_i(\mathbf{x})$$

dove $\beta_1, \beta_2, \ldots, \beta_{2^n}$ sono i coefficienti spettrali generalizzati (si veda l'equazione (4.10)).

Analogamente a quanto è stato fatto con i coefficienti di Fourier, possiamo dare la seguente definizione.

**Definizione 4.10.** Sia $f \in \Re^{2^n}$; definiamo la *$L_1$-norma spettrale generalizzata* di $f$ rispetto alla base $\mathcal{G}$ nel modo seguente:

$$\|f\|_1^{\mathcal{G}} = \sum_{i=1}^{2^n} |\beta_i|$$

■

Il teorema che segue è una generalizzazione immediata del risultato espresso dal Teorema 4.6. La dimostrazione è molto simile a quella esposta per il Teorema 4.6, e quindi non viene qui riportata.

**Teorema 4.15.** *Sia $f \in BIN_n$ una funzione tale che $\|f\|_1^{\mathcal{G}} \leq n^c$ per una costante $c$ fissata. Allora, per ogni $k > 0$, esiste una funzione:*

$$F(\mathbf{x}) = \frac{1}{N} \sum_{i \in S} w_i g_i(\mathbf{x})$$

*con tutti i coefficienti $w_i$ ed $N$ interi polinomiali, e in cui la cardinalità dell'insieme $S \subset \{1, 2, \ldots, 2^n\}$ è polinomiale, tale che:*

$$\left|F(\mathbf{x}) - f(\mathbf{x})\right| \leq n^{-k} \qquad \forall\, \mathbf{x} \in \{1, -1\}^n$$

■

Naturalmente, ha senso applicare il teorema appena enunciato solo per classi $\mathcal{G}$ di funzioni che non abbiano esse stesse $L_1$-norma (di Fourier) polinomiale. Agli effetti pratici ciò costituisce un problema, in quanto a tutt'oggi non si conoscono basi per lo spazio vettoriale $\Re^{2^n}$ costituite da funzioni che non appartengano a $\widehat{LT}_2$. Ne consegue che un possibile sviluppo delle tecniche spettrali applicate allo studio della complessità dei circuiti a soglia potrebbe essere quello di trovare una base per lo spazio vettoriale $\Re^{2^n}$ che sia costituita da funzioni separatrici per le classi $\widehat{LT}_2$ e $\widehat{LT}_3$. Tale base, utilizzata in concomitanza con il Teorema 4.14, potrebbe eventualmente consentire di determinare una funzione separatrice per le classi $\widehat{LT}_3$ e $\widehat{LT}_4$.

# Bibliografia


[1] Balcázar J.L., Díaz J., Gabarró J. *Structural Complexity.* Voll. I e II, Springer-Verlag, Berlin, 1988-1990.

[2] Bruck J. *Fourier Transforms and Threshold (Neural) Circuit Complexity.* Manoscritto, California Institute of Technology, Pasadena, CA, 1994, pp. 1–32.

[3] Furst M., Saxe J., Sipser M. *Parity, Circuits, and the Polynomial Hierarchy.* Math. Systems Theory, Vol. 17, 1984, pp. 13–27.

[4] Goldmann M., Håstad J., Razborov A. *Majority Gates vs. general weighted Threshold Gates.* Computational Complexity, Vol. 2, No. 4, 1992, pp. 277–300.

[5] Goldmann M., Karpinski M. *Simulating Threshold Circuits by Majority Circuits.* In Proc. of the 25th Annual ACM Symposium on the Theory of Computing (STOC), San Diego, CA, Maggio 1993, pp. 551–560.

[6] Goldmann M. *On the power of a Threshold Gate at the top.* Information Processing Letters, Vol. 63, No. 6, Settembre 1997, pp. 287–293.

[7] Hajnal A., Maass W., Pudlák P., Szegedy M., Turán G. *Threshold Circuits of Bounded Depth.* Journal of Computer and System Sciences, Vol. 46, No. 2, 1993, pp. 129–154.

[8] Håstad J.T. *Computational Limitations for Small-Depth Circuits.* ACM Doctoral Dissertation Award 1986. The MIT Press, Cambridge, MA, 1987.

[9] Håstad J. *On the size of weights for threshold gates.* SIAM J. Discrete Math., Vol. 7, 1994, pp. 484–492.

[10] Lewis H.R., Papadimitriou C.H. *Elements of the Theory of Computation.* Prentice-Hall International Editions, Englewood Cliffs, NJ, 1981.

[11] Muroga S. *Threshold Logic and its Applications.* Wiley-Interscience, New York, 1971.

[12] Papadimitriou C.H. *Computational Complexity.* Addison-Wesley, 1994.

[13] Roychowdhury V.P., Siu K.Y., Orlitsky A., Kailath T. *A Geometric Approach to Threshold Circuit Complexity.* In Proceeding of the Fourth Annual Workshop on Computational Learning Theory (COLT), Agosto 1991, pp. 97-111.

[14] Roychowdhury V.P., Siu K.Y., Orlitsky A. *Theoretical Advances in Neural Computation and Learning.* Kluwer Academic, Boston, MA, 1994.

[15] Siu K.Y., Bruck J. *On the Power of Threshold Circuits with Small Weights.* SIAM J. Discrete Math., No. 4, 1991, pp. 423–435.

[16] Siu K.Y., Roychowdhury V.P. *On Optimal Depth Threshold Circuits for Multiplication and Related Problems.* SIAM J. Discrete Math., Vol. 7, No. 2, 1994, pp. 284–292.

[17] Siu K.Y., Roychowdhury V.P., Kailath T. *Discrete Neural Computation: A Theorical Foundation.* Prentice Hall, Englewood Cliffs, NJ, 1994.