Universitá degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Computer Science (Classe L-31)

Reti HTM per il riconoscimento di pattern

Laureando
Simone Morettini
Matricola 095131

Relatore
Prof. Marco Piangerelli

A.A. 2017/2018

Indice

Elenco delle figure **iv**

Elenco degli acronimi e abbreviazioni **v**

Introduzione **1**

1 Teoria delle reti HTM **2**
1.1 Biologia 2
1.2 Confronto con le reti neurali 5
1.3 Funzionamento 6
1.3.1 SDR 7
1.3.2 Codifica 11
1.3.3 Spatial pooler 14
1.3.4 Temporal Memory 16
1.3.5 Interpretazione degli output 17
1.3.6 Struttura a più livelli 19

2 Applicazioni **21**
2.1 Rilevazione di anomalie 21
2.2 Predizioni di consumi energetici 27
2.3 Predizioni di attacchi epilettici 29

3 Conclusioni **33**

Bibliografia **34**

Siti consultati **36**

Elenco delle figure

1.1 Semplice diagramma che mostra la struttura gerarchica di una rete HTM 3
1.2 Apprendimento di un oggetto tramite un organo unisensoriale 4
1.3 Rappresentazione di un neurone piramidale 5
1.4 Confronto tra i modelli di neuroni 6
1.5 Tabella delle probabilità di falsi match in base a n e w 10
1.6 Tabella dei falsi match in base alla soglia 10
1.7 Rappresentazione di un encoder numerico 11
1.8 Rappresentazione di uno spatial pooler 14
1.9 Apprendimento di una sequenza temporale 17
1.10 Schematizzazione di tutti gli step per utilizzare una rete HTM in un problema generico 19
1.11 Modello di rete complesso utilizzato per il riconoscimento di oggetti attraverso tre sensori 20

2.1 Grafici di un intervallo dei dati considerati 22
2.2 Grafico degli ingressi, dell'anomaly score e dell'anomaly likelihood . . . 24
2.3 Tabella dei risultati sul dataset del campus universitario 24
2.4 Grafici di una sezione del dataset del traffico stradale 25
2.5 Grafico degli ingressi, dell'anomaly score e dell'anomaly likelihood . . . 26
2.6 Tabella dei risultati sul dataset riguardante il traffico 26
2.7 Grafico di un mese di dati del cliente numero 16 27
2.8 Confronto tra il valore reale e le predizioni ottenute dalla rete nella predizione dei consumi 28
2.9 Grafici degli indicatori utilizzati per valutare l'algoritmo di predizione dei consumi 28
2.10 Tabella dell'indice MAPE e RMSE calcolato su 10000 valori di alcuni clienti 29
2.11 Segnali pre-ictali e interictali 30
2.12 Esempi di STFT con finestra a trenta secondi 30
2.13 Risultati della rete HTM applicata a dei dati pre-ictali e interictali . . . 32

Elenco degli acronimi e abbreviazioni

AAE Average Absolute Error

ANN Artificial Neural Network

EEG Elettroencefalogramma

CNN Convolutional Neural Network

HTM Hierarchical Temporal Memory

LSTM Long Short-Term Memory

MAPE Mean Absolute Percent Error

MMPP Markovmodulated Poisson process

NRMSE Normalized Root-Mean-Square Error

NuPIC Numenta platform for Intelligent Computing

SDR Sparse Distributed Representation

SP Spatial Pooler

STFT Short Time Fourier Transform

TM Temporal Memory

Introduzione

L'obiettivo di questa tesi è quello di conoscere e sperimentare una tecnica di machine learning nata recentemente, le HTM, memorie gerarchiche temporali. Questa tecnologia è ancora in via di sviluppo ma presenta aspetti interessanti, sia nella sua progettazione, basata sulla neuroscienza, sia nel funzionamento. Ad un primo sguardo sembrano molto simili alle reti neurali ma pur avendo alcune caratteristiche in comune, differiscono su tanti altri fronti. La tecnologia è poco utilizzata e ricopre ancora un bacino di utenti di nicchia. Dopo un profondo studio, dei test in diversi ambiti sono stati realizzati per verificarne le potenzialità nella predizione e nella rilevazione di anomalie.

La tesi è divisa in due parti: una prima parte, più teorica, in cui si descrive il funzionamento delle HTM; una seconda parte in cui si propongono alcune applicazioni pratiche.

Nel capitolo I è proposta la teoria delle HTM. La prima sezione riguarda la biologia da cui questo modello è ispirato, la seconda propone un confronto con le reti neurali e nella terza viene spiegato il funzionamento attraverso l'analisi delle varie parti: la comunicazione dei dati, la codifica, gli algoritmi alla base delle HTM, cioè Spatial Pooler e Temporal Memory, l'interpretazione degli output e infine sono proposte delle strutture più complesse realizzabili unendo delle HTM.

Nel capitolo II vengono proposte delle applicazioni realizzate utilizzando le HTM. Nella prima sezione si propongono due problemi riguardo la rilevazione di anomalie, nella seconda un problema legato alla predizione di consumi energetici e nell'ultima ci si occupa della predizione di attacchi epilettici.

Nel capitolo III sono riportate le conclusioni del mio progetto tratte analizzando i risultati ottenuti nelle applicazioni proposte.

Capitolo 1

Teoria delle reti HTM

La sigla HTM (Hierarchical Temporal Memory) ha un duplice utilizzo, viene usata sia per descrivere tutta la teoria che riguarda il funzionamento della neocorteccia cerebrale, cioè come funziona e come si relaziona con il resto delle strutture che formano il cervello umano, per creare l'intelligenza, e sia per indicare le tecnologie utilizzate nelle macchine che seguono i principi neocorticali. Questa teoria compare per la prima volta nel libro "On Intelligence" con il nome di Memory Prediction Framework [1]. Successivamente essa viene rielaborata e sviluppata ulteriormente dalla Numenta inc.[20] fondata da Jeff Hawkins, scrittore del libro citato.

L'intelligenza, secondo l'autore del libro, rappresenta la capacità di prevedere il futuro per analogia con il passato, di memorizzare pattern, di riconoscerli e di compiere predizioni. L'idea avuta da Jeff Hawkins e dai suoi collaboratori è di ispirarsi alla struttura biologica del cervello per costruire modelli tecnologici da applicare a qualsiasi tipologia di dati. Questo approccio consente di applicare il modello di intelligenza studiato a campi da cui la biologia non riesce ad estrapolare informazioni. Per esempio è possibile utilizzare le HTM su dati GPS che codificati nel giusto modo rendono possibile prevedere percorsi, trovare pattern nei movimenti e rilevare anomalie. Stessa cosa vale per il radar o dati più complessi per cui gli esseri viventi non hanno dei sensi che riescono a codificare e dare in pasto alla neocorteccia.

L'obiettivo di Numenta non è quello di ricreare tutto il cervello ma quello di studiarlo a fondo, comprendere come funzionano i processi che rendono possibile l'intelligenza e emularli per creare un'intelligenza artificiale [2].

1.1 Biologia

Il sistema nervoso come lo conosciamo oggi è il risultato del processo evolutivo: durante questo processo sono stati aggiunti strati con delle funzioni via via più complesse. All'inizio esso era composto solo dalla spina dorsale, come negli attuali bruchi, poi si sono aggiunti livelli come il tronco encefalico, i gangli basali e molte altre parti. Ciascun livello creatosi nell'evoluzione va ad aggiungersi a quelli precedenti apportando una complessità sempre maggiore. La struttura che si viene a creare è di tipo gerarchica dato che gli elementi più recenti comunicano, sia in input che in output con quelli più vecchi. Questa organizzazione gerarchica delle diverse regioni si verifica anche a livello fisico dato che le regioni formatesi più recentemente circondano quasi completamente le regioni createsi meno recentemente. Nel livello più esterno si trova la *neocorteccia* che è la parte creatasi per ultimo nel cervello umano e ne costituisce circa il 75% del

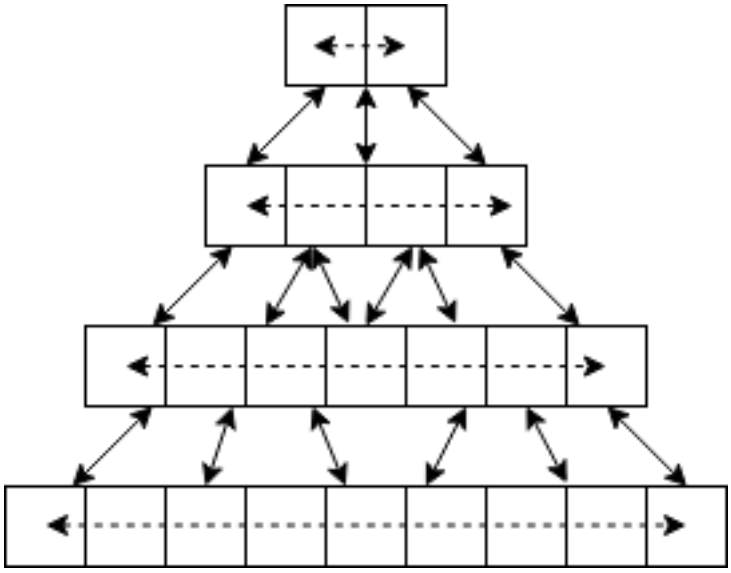

Figura 1.1: Semplice diagramma che mostra la struttura gerarchica di una rete HTM e come le regioni comunichino fra loro sia verticalmente che orizzontalmente.

volume. In essa ha origine l'unica implementazione di intelligenza che ad oggi tutti aderiscono a intendere come tale. La neocorteccia è una specificità dei mammiferi, è apparsa all'incirca 200 milioni di anni fa e la separazione tra la neocorteccia delle scimmie e quella umana risale a circa 25 milioni di anni fa. Quest'ultima differisce per complessità e dimensione raggiungendo i valori attuali tra 800000 e 200000 anni fa.

Essa è un tessuto sottile dello spessore di $3 \div 4$ mm e con una superficie di $2500cm^2$. Situata al di sotto del cranio, avvolge le altre regioni del cervello. Per rientrare in questo spazio ristretto essa è accartocciata su sé stessa. È composta da neuroni nell'ordine di 28×10^9 ed è proprio questo numero molto maggiore rispetto ai cervelli degli alti mammiferi che rende l'uomo una specie intelligente.

La neocorteccia è suddivisa in regioni adiacenti non visibilmente separate ma distinguibili perché non connesse fra loro direttamente, ma tramite delle fibre nervose che connettono due regioni passando per lo strato sottostante, la materia bianca. Il numero di regioni e le loro connessioni sono definite nei geni e differiscono tra specie differenti. Questo sistema di connessioni e regioni è stato mappato per alcune specie [3] e da ciò si è scoperto che si creano delle complesse strutture gerarchiche. Perciò ciascuna regione elabora dei dati e manda i risultati a una o più regioni, questi vengono elaborati e inoltrati più volte. Una semplice schematizzazione della struttura gerarchica della neocorteccia è riportata in figura 1.1 .

La particolarità della neocorteccia è il suo alto grado di omogeneità, se si vanno a studiare le diverse regioni non si notano differenze, al massimo capita di trovare un diverso numero di livelli o di cellule ma la struttura è la stessa malgrado si occupino di elaborare dati totalmente diversi come potrebbero essere informazioni provenienti dal senso tattile o dalla vista. La potenza di questo design è che una volta sviluppata la neocorteccia l'evoluzione non ha dovuto far altro che replicare questa semplice struttura per aumentare sensitivamente le capacità. Ciò spiega come sia stato possibile che l'evoluzione del cervello degli umani sia stata così veloce.

Gli input che riceve la neocorteccia sono di due tipologie. Il primo consiste dai

dati prodotti dai sensi. Questi non arrivano direttamente dagli organi sensoriali ma sono ricevuti dalla vecchia parte del cervello che li riceve dai sensi tramite i nervi. Il secondo ingresso consiste dai comandi eseguiti dalle parti del cervello sottostante. Per comprendere il secondo input prendiamo come esempio il riconoscimento di un oggetto attraverso il tatto. La neocorteccia è capace di comprendere l'oggetto che si sta toccando in base a una sequenza di sensazioni, per esempio se percepisce per quattro volte uno spigolo potrebbe comprendere che sta toccando un oggetto di forma quadrata ma potrebbe anche capitare che lo spigolo che riceve sia sempre lo stesso perciò potrebbe trattarsi di tutt'altro oggetto. Dato ciò è importante conoscere i comandi motori eseguiti per dare contesto agli input sensoriali. Essi permettono inoltre alla corteccia di distinguere se i cambiamenti ricevuti dipendano dall'ambiente o dal proprio movimento. L'output invece è sempre diretto al vecchio cervello che genererà i comportamenti appropriati.

Nella figura 1.2 è riportato un esempio dettagliato di come, attraverso una sequenza temporale di dati, una rete HTM è in grado di riconoscere un oggetto.

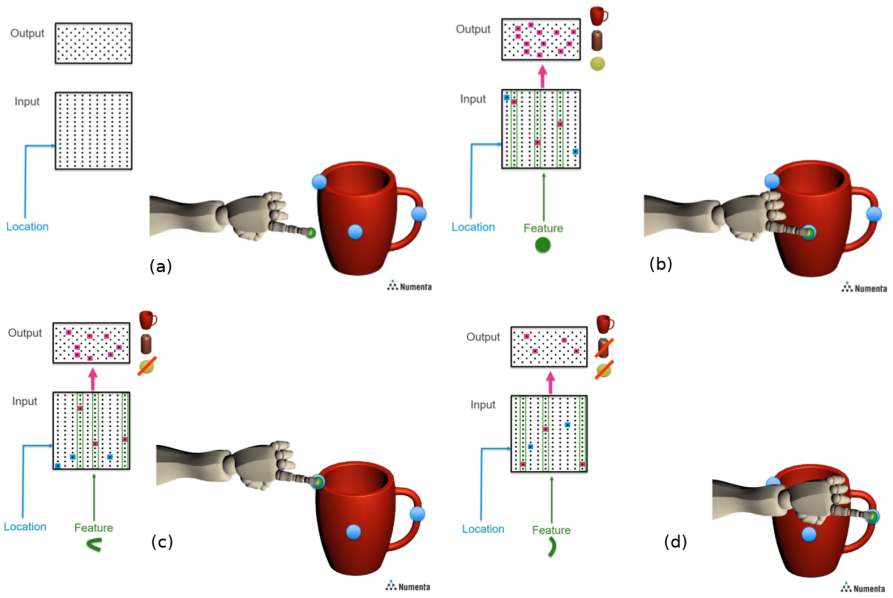

Figura 1.2: Apprendimento di un oggetto tramite un organo unisensoriale. In questo caso abbiamo un solo sensore, l'indice della mano e l'oggetto da riconoscere è una tazza. (a) Situazione di partenza. (b) Apprendimento della caratteristica dell'oggetto che ne indica una superficie curva. Tre possibili oggetti che rispettano la caratteristica selezionata vengono predetti, la tazza, la pallina e una lattina. (c) Apprendimento del bordo della tazza, viene esclusa la pallina che non ha bordi. (d) Apprendimento dell'esistenza del manico della tazza, anche la lattina viene scartata e solamente la tazza viene predetta. La figura è stata realizzata estrapolando delle immagini da [4, 21].

Una caratteristica da notare della neocorteccia è che essa non conosce cosa gli input e output rappresentano. Essa è solo capace di imparare come i due input, costituiti da uno flusso continuo di dati, sono relazionati fra loro, e come può influenzarli dando un certo output. Andando ancora più in dettaglio ogni regione della neocorteccia non sa dell'esistenza delle altre e non è neanche pre-programmata nel compiere qualcosa. Ciò basta per ottenere un'intelligenza.

Le regioni della corteccia nominate precedentemente sono composte da colonne corticali o macrocolonne. Esse a loro volta sono suddivise in diversi strati o, utilizzando il termine inglese, layer. Solitamente uno solo di questi strati è in grado di comunicare con le colonne adiacenti. Gli algoritmi spiegati in seguito possono essere collocati all'interno di un layer di queste colonne. Ciascuno strato è composto da delle strutture chiamate mini colonne. Esse non sono delineate visivamente. La loro caratteristica è che tutti i neuroni che le compongono ricevono tutte lo stesso input di tipo feed-forward cioè quell'input che proviene da altri livelli della colonna. Oltre al segnale già indicato i neuroni ricevono segnali dagli altri neuroni presenti all'interno dello strato della colonna. Il neurone piramidale o cellula piramidale è composta da un nucleo e da connessioni per comunicare con le altre cellule dello stesso tipo. Le connessioni avvengono attraverso delle sinapsi raggruppate in dentriti, questi sono di due tipi: i distali sono le connessioni che ricevono dati da altri neuroni dello stesso livello e rappresentano il contesto, i ricevono input da layer sottostanti o sovrastanti e sono quelli che ricevono l'input di tipo feed-forward. Un tipo particolare di dentrite è l'assone, esso serve per ricevere l'ingresso di feedback. Il nucleo elabora i segnali in ingresso e produce un'uscita che a sua volta verrà rilevato ad altri neuroni. Questa struttura viene ripresa molto fedelmente nelle reti HTM [5].

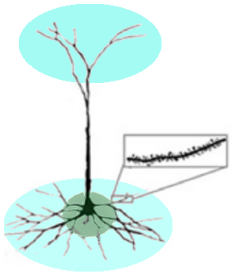

Figura 1.3: Rappresentazione di un neurone piramidale. Al centro della zona verde vi è il nucleo circondato dai dentriti prossimali. Nelle zone in azzurro ci sono i dentriti distali. L'ingrandimento su uno dei dentriti mostra le sinapsi.

1.2 Confronto con le reti neurali

L'approccio nella realizzazione del modello è la differenza principale che causa tutte le altre distinzioni tra le classiche reti neurali, cioè le artificial neural network (ANN) e le HTM. La mission di Numenta è proprio quella di studiare e conoscere a fondo il cervello e applicare le conoscenze acquisite al machine learning perciò i modelli sviluppati sono sempre collegati alla struttura del cervello [2]. Dato che le reti vengono eseguite da dei calcolatori elettronici vengono effettuati diversi accorgimenti ma sempre con l'obiettivo di emulare il più possibile i processi che si sviluppano nella neocorteccia. Le reti neurali e gli altri modelli di deep learning sono costituiti applicando algoritmi matematici e altre conoscenze dell'ambito software a strutture simili a quelle del cervello.

Per esempio le ANN implementano dei neuroni collegati fra loro da sinapsi come nella struttura del cervello ma poi l'attivazione o meno dell'unità è calcolata attraverso una sommatoria pesata e una funzione di attivazione. Quest'ultimi sono modelli matematici elaborati e molto diversi rispetto a quello che avviene nei neuroni biologici.

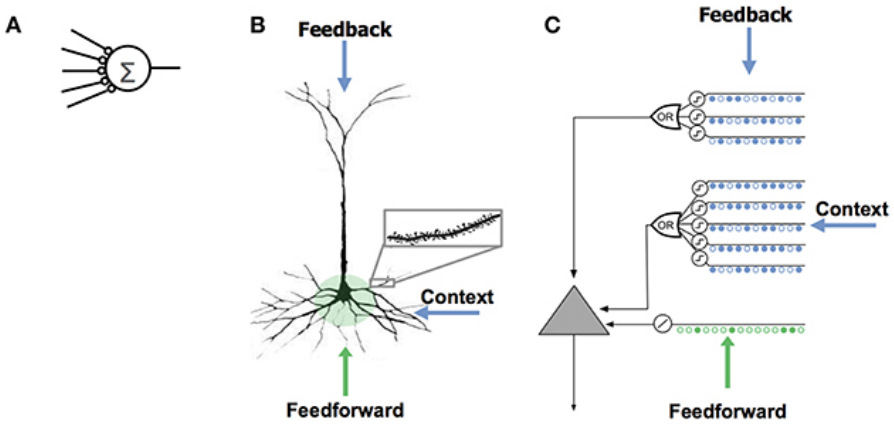

Figura 1.4: Confronto tra i modelli di neuroni. (A) Il modello di neurone utilizzato nelle reti neurali. Vi è solo una tipologia di ingresso e una somma pesata che definisce l'uscita. (B) Il modello del neurone piramidale presente nel cervello con più tipologie di ingressi, l'ingresso feed-forward ottenuto dai livelli inferiori, il contesto dai neuroni dello stesso livello e il feedback dai neuroni del livello superiore. (C) Il modello del neurone HTM simile a quello piramidale cioè con diverse tipologie di input.

Sempre nella struttura vi è una differenza nell'input ricevuto dai neuroni. Come mostrato nella figura 1.4 il neurone piramidale riceve diversi input di cui ciascuno ha un significato diverso, ciò viene riproposto nel neurone delle reti HTM ma non in quello delle reti neurali.

I due approcci differiscono anche sul target dei problemi da risolvere. Mentre le ANN sono implementate per risolvere una precisa gamma di problematiche con una certa tipologia di dato, le HTM propongono di applicare lo stesso approccio per tutto, sia che essi riguardino di elaborare dati audio, immagini o dati prodotti da altri sensori che poi è ciò che accade nella neocorteccia dove tutte le regioni sono uguali pur trattando dati di tipologia differente. Anche la modalità di apprendimento cambia, nelle ANN si ha una fase per il training cioè destinata a definire il modello in modo che dia i risultati richiesti e poi vi è la fase di utilizzo. Con questo nuovo paradigma non vi è bisogno di avere due fasi, la rete impara continuamente. Più dati riceve più i risultati vengono accurati. Inoltre ha un approccio senza sorveglianza cioè non ha bisogno di ricevere come input il risultato richiesto in modo che si adatti a dare quel risultato come avviene nella maggior parte degli algoritmi classici di deep learning.

1.3 Funzionamento

La Hierarchical Temporal Memory, tradotta in italiano in Memoria Temporale Gerarchica, contiene nel nome le tre caratteristiche principali della loro architettura. Queste reti sono delle memorie diverse da quelle a cui siamo abituati a pensare comunemente in ambiente informatico. La memorizzazione avviene con la costruzione di connessioni tra diversi neuroni. La gerarchia permette al sistema di imparare elementi sempre più complessi unendo sezioni semplici del sistema tutte uguali nel funzionamento. In questo modo le regioni nei livelli minori imparano sequenze piccole e semplici e le regioni nei livelli superiori imparano sequenze di sequenze. Così le parti superiori sono indi-

pendenti da piccole variazioni ma riescono a imparare concetti più ampi. Infine sono temporali perché si occupano di flussi di dati che cambiano nel tempo e se vi sono dei modelli che li guidano esse sono in grado di apprenderli.

Per la realizzazione di questa architettura si ha bisogno di diverse tecnologie e algoritmi. La rappresentazione usata nelle HTM è denominata sparse distributed representation (SDR). Essa permette di rappresentare qualsiasi tipo di dato applicando l'opportuna codifica. In questo modo si rispetta l'obiettivo di mantenere queste reti universali. Nelle sezioni successive verrà descritto in maniera approfondita come viene effettuato l'apprendimento all'interno della rete. Esso avviene durante la ricezione dei dati senza l'utilizzo di una fase di training obbligatoria e perciò è chiamato online. Bisogna precisare che in alcuni casi viene suddivisa la fase di allenamento con quella di utilizzo della rete ma sono situazioni particolari, la rete non è stata progettata con questa ottica. Il flusso base della rete è abbastanza semplice. Essa riceve dei dati e da questi predice ciò che si aspetterebbe di ricevere nell'istante successivo. Ciò che si verifica viene utilizzato per rinforzare la rete e migliorare così le successive predizioni.

1.3.1 SDR

La SDR è una rappresentazione distribuita e sparsa. Essa deriva dallo studio dei neuroni all'interno del cervello i quali sono attivi contemporaneamente in una piccola percentuale, solitamente minore del 2%. Una SDR è rappresentata con un vettore di migliaia di bit di cui una piccola porzione indica i neuroni attivi, cioè impostati a 1. Il resto, cioè i neuroni inattivi, sono impostati a 0. Le SDR sono usate come input, come output e nella comunicazione all'interno della rete tra diverse regioni. Ciò consente all'architettura di non avere problemi nel trattare dati di tipologia diversa, l'importante è che vengano codificati in modo opportuno in una SDR.

Usare questo metodo per la comunicazione permette di inserire la semantica all'interno dei dati e di dare alle macchine la flessibilità e la creatività caratteristiche degli esseri umani. La rappresentazione binaria utilizzata dai computer solitamente è una codifica rigida e ciò rende difficile creare algoritmi che interagiscono con il mondo non discreto dove praticamente tutto comporta eccezioni e relazioni complesse difficilmente rappresentabili con un sistema rigido. Nelle SDR ogni bit ha uno specifico significato, questo non significa che vengono etichettati ma sta alle rete imparare ciò. Se per esempio due SDR condividono dei bit attivi, significa che hanno delle caratteristiche in comune. Ogni bit attivo rappresenta una proprietà di ciò che rappresenta. Per vedere quanto due SDR sono semanticamente simili basta sovrapporli e vedere quanti bit attivi condividono.

Le SDR rispecchiano molto la memorizzazione delle informazioni nel cervello essendo estremamente resistenti all'introduzione di rumori e disturbi. Un'altra capacità interessante è quella di poter selezionare solo una sezione di una rappresentazione senza perdere informazioni come può accadere nella mente umana. Prendiamo come esempio il riconoscimento di un frutto a distanza di molto tempo dall'ultimo assaggio. Alcune sue caratteristiche potrebbero essere dimenticate ma l'uomo è capace di identificarlo anche ricordando solo alcune delle sue proprietà.

Una differenza tra la rappresentazione di informazioni nei computer tradizionali e le SDR è che solitamente si usa una rappresentazione densa, ogni combinazione di zero e uno ha un significato. Ciò non capita nelle SDR in cui la rappresentazione è di tipo sparso. Inoltre nelle rappresentazioni dense cambiare un bit a caso significa ottenere

tutt'altro risultato, nelle SDR invece potrebbe non avere influenza o averne poca perché si otterrebbe un oggetto semanticamente simile a quello di partenza. Un esempio di forte impatto può essere la rappresentazione di un carattere. Da un lato troviamo la codifica ASCII in cui ogni lettera è rappresentata con una sequenza di bit, dall'altro potremmo scegliere di utilizzare alcuni bit per indicare se la lettera è una vocale o consonante, altri per indicare se la lettera è maiuscola o minuscola, altri come suona la lettera o in che posizione è dell'alfabeto. Nel secondo caso otteniamo che ogni bit ha un significato.

Confrontando il sistema binario classico con l'SDR otteniamo che le informazioni rappresentabili per numero di bit sono molte meno con la seconda tecnica. Nel sistema binario con n bit abbiamo

$$2^n$$

possibili combinazioni mentre con le SDR, utilizzando w bit attivi alla volta abbiamo

$$\binom{n}{w} = \frac{n!}{w!(n-w)!}$$

Prendendo $n = 100$ e $w = 2$ (2% dei bit attivi) otteniamo 1.2676506×10^{30} e per le SDR 4950 combinazioni uniche. Per risolvere tale inconvenienza nelle HTM si usano SDR molto più grandi, nell'ordine del migliaio di bit. Già 256 bit con solo cinque attivi contemporaneamente dà la possibilità di ottenere oltre otto miliardi di combinazioni con un livello di sparsità minore del due per cento. Questa rappresentazione è criticabile essenzialmente per due motivi: spreco di combinazioni e di spazio di memoria dato che l'alto numero di bit occupa molto più spazio del sistema binario a parità di combinazioni. Riguardo al primo punto vedremo che le proprietà matematiche del sistema lo rendono interessante al punto da poter ignorare il problema. Invece, rispetto al secondo punto, la soluzione consiste semplicemente nel memorizzare non tutto l'SDR ma solo la posizione dei bit attivi che sono quelli interessanti perché contengono le informazioni.

$$SDR - in - binario = \{0, 0, 0, 0, 1, 0, 0, 1, 0\}$$

$$SDR - indici = \{5, 8\}$$

Riassumendo un SDR ha tre parametri:

- n: indica la dimensione, cioè il numero totale dei bit utilizzati per la rappresentazione;
- w: indica il numero di bit attivi, cioè di valore uno;
- s: indica la sparsità dell'SDR, esso è il rapporto tra i bit attivi e il numero totale di bit, $s = \frac{w}{n}$;
- r: indica la differenza minima tra due valori numerici per cui la loro codifica SDR non debba avere bit attivi in comune.

Inoltre definiamo i seguenti concetti:

- **Overlap**: è l'indicatore di similarità tra due SDR, cioè il numero di bit attivi in comune fra di essi. Si calcola eseguendo il prodotto scalare tra le due rappresentazioni;

- **Matching**: si verifica quando vi è una corrispondenza tra due SDR e perciò l'overlap supera una certa soglia prefissata. Se si usa come soglia w si può avere un match esatto che si contraddistingue da un match inesatto nel caso la soglia è minore di w. La soglia sarà indicata con $\theta, 0 \leqslant \theta \leqslant w$.

La rappresentazione delle HTM consente di ottenere una "unicità di rappresentazione": presi due SDR, con gli stessi parametri n e w, la probabilità che esse siano uguali è bassa e raggiunge livelli insignificanti al crescere di n e w. La forte tolleranza ai cambiamenti e ai rumori è dimostrata calcolando gli overlap set, cioè il numero di SDR che permettono di avere un matching con una SDR di partenza scegliendo come soglia un valore $b, b < w$. Perciò scegliendo b leggermente minore di w si ha un gran numero di SDR che hanno b bit attivi in comune. Ciò ha due utili conseguenze, ma prima introduciamo anche la probabilità di un *falso match*. Essa indica la probabilità che paragonando due SDR con b bit attivi in comune si verifichi un matching non voluto. La probabilità di falsi match è molto bassa negli SDR dove n è molto grande. Questo significa che se i dati in input presentano dei rumori o dei disturbi, cioè diversi bit sono scambiati casualmente, la probabilità che si abbiano errori nel matching rimane molto bassa. Per esempio, in un SDR con $n = 1024$, $w = 4$ e scegliendo la soglia per il matching pari a 2, perciò accettando fino a un 50% di disturbi e dati errati, la probabilità di un errore sarebbe pari a $\frac{1}{14587}$. La seconda conseguenza è la possibilità di fare subsampling, cioè la possibilità di confrontare attraverso l'overlapping due SDR selezionandone solo una sezione dei bit attivi. Questo è possibile dalle caratteristiche sopra descritte, cioè considerando solo b bit è possibile ottenere il matching tra due SDR con un bassissima probabilità di falsi positivi. Naturalmente più il grado di subsampling è alto, cioè più il numero di bit considerati è basso, più si otterranno falsi positivi. Questa proprietà può essere utilizzata per la compressione di SDR prendendo solo alcuni dei bit attivi per rappresentare un SDR. Nelle figure 1.5 e 1.6 sono riportati, titolo di esempio, dei calcoli per dimostrare quantitativamente ciò che è stato discusso [6].

Alzando il valore di soglia si ha meno probabilità di falsi matching e si ha una grande robustezza. Abbassando invece questo valore, aumenta la probabilità di errore e decrementa la sensitività, ma si incrementa la robustezza agli errori e ai disturbi. La scelta va fatta tenendo conto cosa è più importante in base alle situazioni.

Infine l'ultima caratteristica molto utile è l'operazione di unione. Essa consiste nell'effettuare l'operazione di OR tra due o più SDR ottenendone così una nuova molto particolare. Il risultato ha tutti i bit attivi nelle posizioni dei vettori usati come input. Ciò significa che applicando il matching con uno degli input si avrà risultato positivo. Questa banale proprietà consente di memorizzare una lista di SDR in un unica SDR ottenuta attraverso l'unione di tutti gli elementi della lista. Per controllare se una certa rappresentazione fa parte dell'insieme basterà eseguire l'operazione di matching fra la SDR risultante dall'unione e quella interessata. Il grande vantaggio è che con un unico vettore di dimensione n, usando la rappresentazione binaria dell'SDR, o di dimensioni al massimo $w \times M$, con M numero di elementi nel set, nel caso che si memorizzi gli SDR utilizzando gli indici dei bit attivi, si può memorizzare una grande mole di informazioni che altrimenti richiederebbe una lista di dimensioni variabili. Questa caratteristica però presenta dei limiti. Con M abbastanza grande l'unione risulterebbe in una SDR con tanti bit attivi e perciò con un basso livello di sparsità creando così tanti falsi positivi.

Le operazioni descritte sono computazionalmente efficienti anche se i vettori SDR sono di grandi dimensioni. Questo accade perché il costo computazionale è indipendente

n	w	Number of patterns	Prob. of false match
64	1	64	0.015625
64	3	41664	2.40015E-05
64	5	7624512	1.31156E-07
64	7	621216192	1.60975E-09
64	9	27540584512	3.631E-11
64	11	7.43596E+11	1.34482E-12
512	1	512	0.001953125
512	3	22238720	4.49666E-08
512	5	2.87516E+11	3.47807E-12
512	7	1.75619E+15	5.69416E-16
512	9	6.20812E+18	1.61079E-19
1024	1	1024	0.000976563
1024	3	178433024	5.60434E-09
1024	5	9.29119E+12	1.07629E-13
1024	7	2.29479E+17	4.35769E-18
1024	9	3.29326E+21	3.03651E-22

Figura 1.5: La tabella mostra la probabilità di falsi match esatti con diverse combinazioni di n e w. La probabilità di falsi match è molto bassa anche scegliendo i due parametri della SDR abbastanza piccoli.

n	w	t	Prob. of false match
64	4	4	1.57387E-06
64	4	3	0.000379303
64	4	2	0.017093815
64	4	1	0.232525308
64	8	8	2.25929E-10
64	8	7	1.01442E-07
64	8	6	9.84351E-06
64	8	5	0.000360558
64	8	4	0.006169265
64	32	32	5.45666E-19
64	32	24	6.70223E-05
64	32	16	0.59857385
1024	20	20	1.82484E-42
1024	20	17	3.50023E-31
1024	20	14	9.93621E-23
1024	20	10	9.32924E-14

Figura 1.6: La tabella mostra la probabilità di falsi match utilizzando come soglia di matching t. È possibile notare che con $n = 64$ la probabilità di errore è abbastanza alta ma si abbassa drasticamente scegliendo $n = 1024$.

dalle dimensioni dei vettori, esso è $O(w)$, solitamente piccolo rispetto alla dimensione n invece molto grande, $w \ll n$.

1.3.2 Codifica

Nella sezione precedente è stato introdotto il linguaggio utilizzato nelle reti HTM, le SDR. In questa sezione verrà spiegato come si crea questa rappresentazione partendo dai dati con cui si è soliti lavorare in ambito informatico. Per compiere questa operazione di codifica vengono usati degli encoders; questi possono essere paragonati ad organi sensoriali anche se, diversamente dalle HTM, gli encoders non sono ispirati dalla biologia perché altrimenti bisognerebbe ricreare organi molto complessi come gli occhi, il naso e tutte le altre parti del corpo da cui riceviamo informazioni. Il processo di codifica varia in base al dato utilizzato ma ci sono delle caratteristiche che tutti gli encoder devono rispettare:

1. dati con una semantica simile devono essere codificati in SDR simili, cioè con un alto valore di overlapping;
2. ad uno stesso input deve risultare lo stesso SDR in output;
3. l'output deve avere la stessa dimensione per ogni input;
4. gli output devono avere tutti all'incirca lo stesso grado di sparsità.

Rispettando gli aspetti elencati è possibile creare qualsiasi tipo di encoder per qualsiasi tipo di dato. Differenti tipi sono già stati realizzati e coprono una vasta gamma di dati. In seguito ne riporto alcuni esempi. In particolare saranno spiegati quelli utilizzati nei test effettuati.

Encoders per numeri

La codifica base per i numeri si fa assegnando a ciascun bit un intervallo di valori per cui il bit venga attivato cioè posto a 1 come mostrato in figura 1.7. L'intervallo deve essere assegnato in modo che più bit siano attivi contemporaneamente, perciò gli intervalli si devono sovrapporre. La decisione di tali intervalli viene fatta definendo alcuni parametri:

- min: valore minimo rappresentabile;
- max: valore massimo rappresentabile;
- n: dimensione della SDR che si vuole creare;
- w: numero di bit attivi nella SDR.

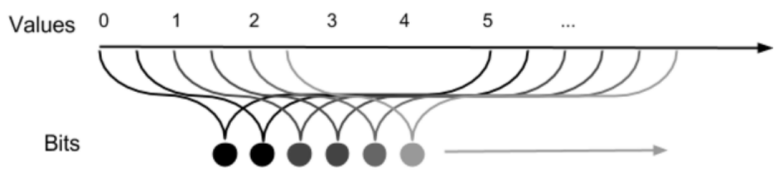

Figura 1.7: Rappresentazione di un encoder numerico. A ciascun bit è assegnato un intervallo di valori per cui è attivato.

Il risultato sarà un SDR che ha w bit consecutivi attivi, cioè tutti quei bit a cui è assegnato un intervallo contenente il valore da codificare. La codifica si compie calcolando il numero di intervalli (buckets) in cui dividere i dati

$$buckets = n - w + 1$$

poi si trova la dimensione dell'intervallo di dati rappresentabili(range)

$$range = max - min$$

e infine si ricava l'indice del primo bit attivo nell'SDR che sarà seguito da w bit attivi

$$i = floor[buckets * (v - min)/range]$$

Se i dati da codificare sono minori di *min* o maggiori di *max* vengono codificati rispettivamente *min* e *max*. Perciò tutti i valori al di sopra del massimo avranno la stessa rappresentazione, ugualmente tutti quelli al di sotto del minimo. Per ovviare a ciò è possibile usare un sistema più flessibile in cui non serve definire un intervallo. Esso comporta l'utilizzo di una funzione di hashing con cui calcolare l'indice dei bit da attivare. In questo modo, con un SDR opportunamente grande, l'encoder resta valido anche se può capitare che numeri non semanticamente simili abbiano qualche bit attivo in comune.

Per ottenere un encoder flessibile è anche possibile far in modo che il minimo e il massimo non siano fissi ma che cambino nel tempo: quando l'encoder riceve valori che escono dall'intervallo codificato modifica i due parametri che definiscono il range. La critica a questo sistema riguarda il fatto che non rispetterebbe la caratteristica numero 2 sopra definita. Infatti variando il range un valore può essere codificato in due modi diversi in base a quando questo viene dato in input all'encoder. Però si può ammettere ciò dato che solitamente l'encoder è applicato a uno flusso continuo di dati, più dati riceve e più convergerà sul corretto range. Dall'altra parte la rete HTM non avrà problemi dato che una delle sue particolarità è proprio quella di imparare durante l'elaborazione dei dati e perciò sarà in grado di adattarsi a questi cambiamenti.

Un altro variante dell'encoder consiste nel codificare il logaritmo del valore, per esempio invece di codificare x si codifica $log(x)$ ottenendo così la proprietà che per i numeri di valore basso vi è un'alta risoluzione mentre per i numeri più grandi la risoluzione diventa molto più bassa. Ad esempio 3 e 4 vengono considerati simili come 3000 viene considerato simile a 4000.

Invece se l'obiettivo è quello di catturare la modifica di un certo valore più che il valore in sé per sé, vi è il Delta Encoder: esso consiste nell'applicare un encoder numerico alla differenza fra il valore attuale e il precedente.

Encoder per le categorie

Questa tipologia di encoder serve nel caso in cui si ha bisogno di codificare dati che possono assumere valori discreti come per esempio i giorni della settimana, lettere o un valore che indica se si tratta di una vacanza o no. La codifica viene fatta affidando un certo numero di bit a ogni categoria che il dato da codificare può assumere. Si potrebbe pensare di affidare un solo bit attivo per ciascuna categoria avendo così degli SDR con $W = 1$. Esso può essere usato nelle reti HTM, non è proibito, ma è altamente sconsigliato dato che la rete difficilmente riuscirebbe a imparare perché tale bit potrebbe essere non considerato abbastanza.

Se le categorie non sono ben discrete è possibile effettuare una conversione in un valore continuo e usare un Encoder designato per i numeri. Ciò accade con i giorni della settimana. Per esempio potrebbe essere utile definire che la mattina di un giorno sia semanticamente simile alla sera del giorno prima. Rimanendo sull'esempio dei giorni della settimana un'altra caratteristica che potrebbe essere utile nell'apprendimento in alcuni casi è il fatto che la domenica e il lunedì sono due giorni consecutivi. In questo caso si può usare una rappresentazione ciclica. Per esempio se nella rappresentazione di un giorno si attivano sia i bit del giorno stesso che una parte del giorno precedente e del successivo, nel caso della domenica si attivano anche alcuni bit del lunedì. Il risultato sarebbe un SDR che ha dei bit attivi agli estremi del vettore, situazione che non capita negli altri casi.

Altri encoder

Ci sono molti altri encoder disponibili e già sviluppati. Uno abbastanza controintuitivo riguarda la codifica dei dati GPS. Una data coordinata viene codificata attivando dei bit che indicano dei punti geografici vicini alla coordinata. Ciò viene effettuato calcolando un'area intorno alla posizione da codificare, dividendola in sottoinsiemi e assegnando a ciascuno un indice numerico. Gli indici ottenuti vengono codificati attraverso una funzione di hashing e in questo modo utilizzando la stessa idea dell'encoder numerico flessibile si ottiene una SDR. Quella proposta è l'idea base ma è possibile renderla più complessa integrando la velocità.

Esistono alcune teorie sulla codifica del linguaggio naturale presentate in questo articolo[8]. Mentre si potrebbe pensare in modo semplice di usare un encoder per le categorie e applicarlo alle lettere o alle parole, nell'articolo viene presentato un modello più complesso che sfrutta a pieno le potenzialità delle SDR. Considerando ogni lettera una categoria o ogni parola una categoria senza considerarne il significato, cosa che accadrebbe con l'encoder delle categorie, si perderebbe un sacco di informazioni. Il modello più elaborato consiste nell'assegnare a ciascun bit dell'encoder una caratteristica e attivarlo o no in base alla parola. Per esempio se si vuole codificare la macchina si accenderanno i bit che significano "ha 4 route", "ha un motore" ma non i bit "ha dei pedali", "può volare". Ciò viene fatto con un complesso encoder che ricevendo una serie di testi crea queste rappresentazioni.

Degli encoder molto utili sono stati sviluppati per le date. Non sono stati presentati perché si basano su quelli numerici cioè la data viene convertita in un numero e viene data in input ad un encoder numerico.

Solitamente si ha interesse a dare in pasto diversi tipologie di dati a una rete HTM. In questi casi ogni dato è codificato con l'encoder appropriato e le SDR prodotte da ciascun encoder utilizzato vengono concatenati in un unico encoder usato come input per la rete HTM. Nel caso si concatenano più SDR bisogna porre attenzione a quanti bit attivi ci sono sulle SDR di partenza, se si vuole dare la stessa importanza a ogni tipo di dato si deve far in modo che tutti le SDR utilizzino all'incirca gli stessi parametri n e w. Se invece si ha bisogno di dare più importanza a un tipo di dato piuttosto che ad un altro si possono impostare i due parametri in modo da bilanciare i dati e sceglierli al meglio secondo le proprie esigenze. Questa tipologia di encoder saranno indicati come encoder multipli.

1.3.3 Spatial pooler

Lo spatial pooler (SP) è il primo algoritmo base delle reti HTM. La sua funzione è quella di normalizzare i dati ricevuti dagli encoder per l'utilizzo nell'algoritmo trattato nella prossima sezione. L'obiettivo è quello di mantenere un grado di sparsità fisso qualunque sia l'input, dato che l'encoder potrebbe creare delle SDR troppo dense. Infatti spesso nella definizione degli encoder non ci si preoccupa della densità del loro output. Esso inoltre mantiene le proprietà di overlap dei dati in input. Nel capitolo legato alla biologia si è fatto riferimento alle minicolonne composte da neuroni che vengono attivate tutte dallo stesso segnale di tipo feed-forward, cioè un bit attivo. Lo spatial pooling viene utilizzato per scegliere quali colonne di neuroni attivare. Sarebbe possibile utilizzare direttamente l'SDR prodotto dall'encoder per decidere le colonne da attivare, cioè ogni minicolonna corrispondente a un bit attivo va attivata, però con lo spatial pooler la rete HTM è in grado di restituire soluzioni migliori.

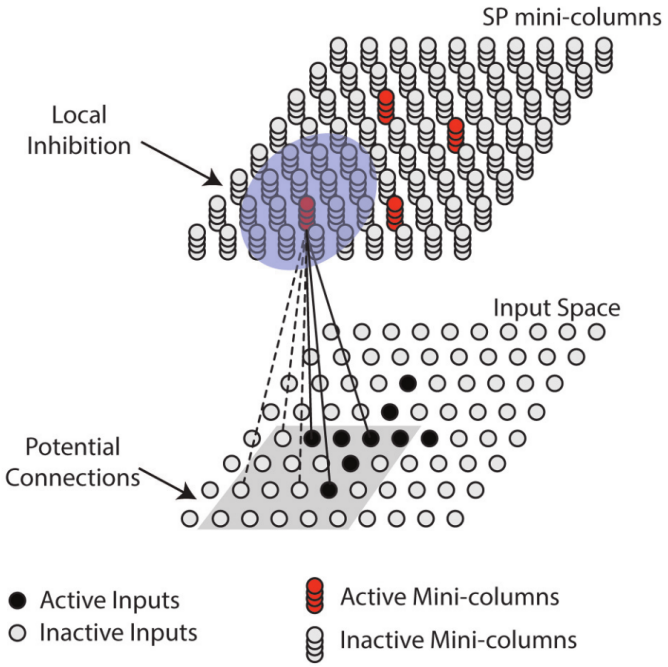

Figura 1.8: Rappresentazione di uno spatial pooler riportata in [9]. Nella rappresentazione solo le connessioni tra una minicolonna attiva e alcuni bit in input sono riportate. Alcune connessioni, quelle delineate da una linea continua sono attive dato che sono collegate a dei bit attivi mentre quelle delineate da una linea tratteggiata sono inattive. La sezione grigia contiene i bit che sono potenzialmente connessi con la minicolonna presa in considerazione. L'inibizione nel caso riportato avviene localmente invece che globalmente cioè solo i bit nel cerchio blu sfumato competono fra di loro.

L'output dell'algoritmo può essere visto come una SDR ma per comodità nella spiegazione e nella comprensione si utilizzerà il concetto di minicolonna, essa è composta da più celle o neuroni. Per ora una cella è attiva solo se tutta la minicolonna è attiva perciò si può ignorare questa suddivisione e si può far riferimento solamente al concetto di minicolonna attiva o inattiva.

Ogni minicolonna è connessa ad una serie di bit dell'input. Le connessioni possibili sono memorizzate all'interno di un potential pool creato casualmente. Una minicolonna

è connessa a una serie di bit della SDR e ciascun bit può, e solitamente è, connesso a più di una minicolonna. A ciascuna connessione è legato un parametro chiamato permanence threshold (soglia di permanenza) che serve per definire se una connessione è reale o solo potenziale. Ci si trova nel primo caso se il valore indicato è maggiore di un parametro configurabile chiamato connection threshold (soglia di connessione) e nel secondo se esso è minore. Questo valore da superare è fisso per tutte le minicolonne. Durante la creazione della rete il permanence threshold viene impostato in modo casuale rispettando una distribuzione statistica in modo che i valori cadano vicino alla soglia di connessione. Una schematizzazione di uno spatial pooler è riportato nella figura 1.8.

L'attivazione delle colonne viene svolta attuando una competizione tra di esse. Nel caso della Global inhibition, la scelta più comune, si stila una lista ordinata di colonne secondo l'overlap score, cioè il numero di connessioni attive di ciascuna minicolonna collegate a dei bit attivi. Di questo elenco, le prime w minicolonne vengono attivate. Questo passaggio è chiamato inibizione perché si vieta alle colonne con un overlap score non sufficientemente alto di apprendere dato che solo quelle attive vengono selezionate per compiere tale azione.

Il processo di apprendimento avviene incrementando o decrementando la permanence threshold delle sinapsi collegate alle minicolonne attive. Si aggiunge un valore positivo nel caso esse siano collegate a una cella attiva, un valore negativo nel caso opposto. In questo modo si creano e si eliminano continuamente le connessioni.

Boosting

Analizzando l'algoritmo descritto fino ad ora è possibile notare che nulla vieta che sempre le stesse colonne siano scelte per l'apprendimento e le altre vengano sempre ignorate. La soluzione viene prendendo spunto dal processo della biologia chiamato regolazione omeostatica dell'eccitabilità neuronale [11]. Nella teoria delle HTM è chiamato boosting. L'obiettivo è quello di permettere a tutte le colonne di esprimersi. Per rendere la competizione fra le colonne più equilibrata si moltiplica l'overlap score per un boost factor assegnato a ciascuna minicolonna in base alla sua attivazione nel tempo. Meno essa è attiva e più questo valore sarà alto.

Utilizzando questa tecnica la rete avrà un apprendimento migliore dato che più minicolonne saranno utilizzate e perciò più informazioni verranno memorizzate.

Topologia

All'interno della neocorteccia la topologia è un fattore molto importante per rilevare dei pattern spaziali. Se per esempio si analizza un'immagine è utile analizzarla per zone. Perciò ogni minicolonna può essere collegata a dei bit localmente vicini nell'SDR. In questo modo ogni minicolonna si focalizza su una zona e prende informazioni solo da essa. L'idea è simile a ciò che avviene con i filtri nelle convolutional neural network (CNN) in cui ciascuno di essi estrae informazioni localmente [10]. Oltre che nelle connessioni si può estendere alla competizione cioè non far competere tutte le colonne insieme ma applicando una local inhibition si fanno competere solo le minicolonne localmente vicine.

È un approccio sicuramente utile in alcuni campi ma poco utilizzato nelle HTM attuali perché pur aggiungendo dei vantaggi comporta un forte incremento nella computazione.

1.3.4 Temporal Memory

Il Temporal Memory (TM) è il secondo algoritmo di base delle HTM e viene utilizzato per l'apprendimento di pattern di sequenze spaziali, rappresentate da SDR, nel tempo. Più approfonditamente l'obiettivo è quello di imparare come le colonne vengono attivate dallo spatial pooler nel tempo. Ciò permette di predire basandosi sul contesto temporale di ciascun input.

Prima di procedere con l'algoritmo bisogna introdurre alcune definizioni. Con proximal input (ingressi prossimali) verranno identificati gli ingressi provenienti dallo spatial pooler cioè gli ingressi di tipo feed-forward. I distal input (ingressi distali) invece sono gli ingressi provenienti dal contesto, cioè da altri neuroni, essi rappresentano le informazioni riguardo al contesto. Queste due tipologie di input saranno chiamate segmenti. Essi richiamano il concetto di dentrite del neurone. Infatti ciascun segmento ha una serie di connessioni a delle celle come avviene con le sinapsi. Solitamente si usano solo due segmenti, uno distale e uno prossimale ma se ne possono aggiungere anche altri. Nel capitolo precedente le celle specifiche delle minicolonne erano ignorate, ora invece esse sono la struttura fondamentale del temporal memory. La controparte biologica di ciascuna cella è il neurone. Una cella può assumere tre differenti stati: attivo, inattivo o in fase di predizione. Quest'ultimo stato indica che la rete predice che tale cella sarà attiva nel successivo instante.

L'algoritmo è composto da due fasi:

1. assegnazione dello stato di attivazione;
2. assegnazione dello stato di predizione.

La prima fase consiste nell'attivare per ogni minicolonna attiva le celle in stato di predizione, se una minicolonna è attiva ma non ha alcuna cella nello stato di predizione, allora tutte le celle della minicolonna vengono attivate (bursting). Il secondo caso si verifica nel caso di un input inaspettato perciò non predetto o se è il primo input della serie, perciò la rete è vuota e qualsiasi input viene considerato non predetto.

Nella seconda fase si impostano le celle nello stato di predizione per l'istante successivo. Per ciascuna cella presente nella rete si conta per ogni segmento il numero di sinapsi connesse a una cella attiva. Se questo numero supera una certa soglia configurabile la cella acquisisce uno stato predittivo.

A questo punto è possibile comprendere perché si utilizzano delle colonne con più di una cella. Ciascuna minicolonna fa riferimento a un input o più precisamente a una parte di esso. Ciascuna cella all'interno della minicolonna fa riferimento allo stesso input ma considerando un contesto differente. Se per esempio si usasse una sola cella per minicolonna si otterrebbe una memoria di ordine singolo cioè in grado di avere un contesto di un solo elemento, cioè un solo stato temporale. Utilizzare minicolonne con una sola cella può essere utile nel caso non sia richiesto il fattore temporale ma si è interessati a catturare solo dati spaziali statici. L'apprendimento di una semplice sequenza temporale è riportato nella figura 1.9 ottenuta da [12].

L'apprendimento avviene in maniera simile all'algoritmo SP. Alle sinapsi vengono associati dei valori numerici nell'intervallo $[0, 1]$ chiamati permanence threshold. Se questo valore supera una certa soglia le sinapsi sono attive, altrimenti sono potenziali. Incrementando o decrementando questo valore le varie celle imparano dei pattern, simile a quanto accade nello SP. Viene aggiunta una distinzione tra le celle attive, usate nell'output e le celle vincenti usate per la definizione delle celle nello stato di predizione.

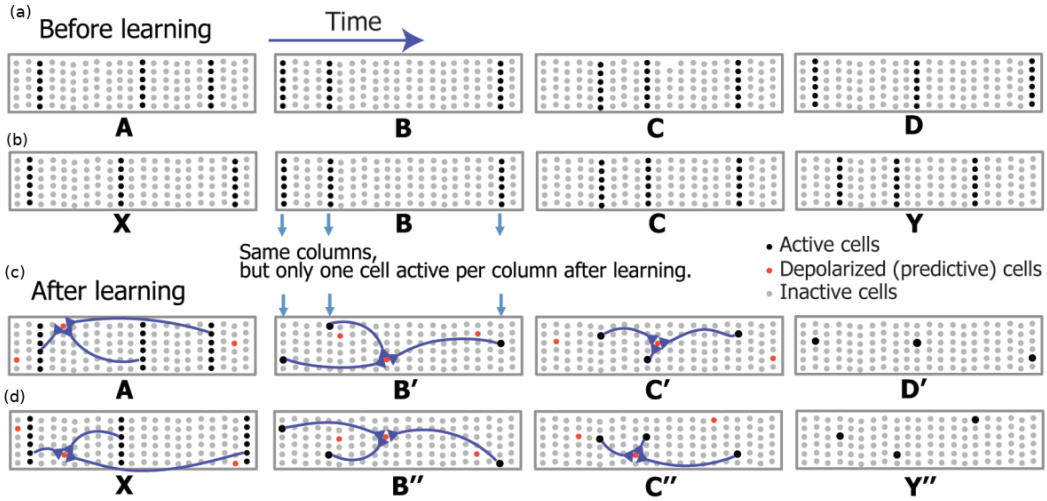

Figura 1.9: Apprendimento di una sequenza temporale. (a) Alla rete viene fornita in input la sequenza ABCD con SDR rappresentati nelle quattro figure. (b) Alla stessa rete viene fornito in input la sequenza XBCY. (c) La rete ha imparato le sequenze precedenti e alla ricezione di ciascuna lettera attiva delle celle(indicate dal colore rosso) che corrispondono alle colonne della lettera che si aspetta di ricevere successivamente. Queste celle vengono attivate perché hanno delle sinapsi connesse a celle attive. La rappresentazione di B, C e D sono indicate con B', C' e D' perché sono influenzate dagli input precedenti. (d) In questo caso B e C hanno una rappresentazione ancora diversa dato che il contesto è diverso dato che la sequenza inizia con X e non con A.

Tutte le celle vincenti sono attive ma non vale il contrario. Nelle colonne di bursting solo una delle celle attive viene contraddistinta come cella vincente.

Se una minicolonna è inattiva ma ha una cella nello stato di predizione vengono punite le sinapsi che hanno causato lo stato di predizione della cella, cioè quelle connesse a celle attive nell'istante precedente. La punizione si fa decrementando il permanence threshold.

Se una minicolonna è attiva e ha delle celle predette, vengono rinforzate le sinapsi che hanno contribuito alla predizione e punite le altre. Cambiando i permanence threshold può capitare che alcune sinapsi non siano più attive. Se ciò accade vengono create nuove sinapsi che collegano tale cella a celle vincenti.

Se una minicolonna è attiva e non ha celle predette viene scelta una cella vincente in due possibili modalità: se tutte le celle hanno delle sinapsi viene selezionata la cella che ha più sinapsi attive, altrimenti viene selezionata la cella meno attiva e vengono create delle sinapsi da tale cella. Scelta la cella vincente vengono rinforzate o punite le sinapsi collegate a tale cella come nel caso precedente.

1.3.5 Interpretazione degli output

L'output di una rete HTM è una SDR che non è direttamente utilizzabile da altre applicazioni e non è possibile estrapolarvi informazioni senza un ulteriore elaborazione. Ci sono due principali modalità per processare questi dati in base all'obiettivo che ci si pone.

Se l'obiettivo del software è effettuare predizioni, si può usare un classificatore di SDR [12] che associa all'output della rete una predizione per l'istante successivo al dato che rappresenta. Se si effettuano predizioni su una serie numerica, il classificatore prende in input l'SDR e restituisce un valore numerico, se si lavora con delle categorie discrete restituisce una categoria discreta e così via. Il suo sviluppo è stato svolto da Numenta[20] ma è al di fuori della teoria delle HTM, è solo uno strumento utile non ispirato alla neocorteccia. La classificazione perciò si svolge mediante l'utilizzo di una rete neurale artificiale che mappa ogni SDR creato dalla HTM in un'etichetta o in un valore numerico. La rete viene allenata dando in input l'output della rete HTM all'istante t e utilizzando come etichetta (label) il valore dato in input all'encoder all'istante $t+1$. Ciò avviene perché si presume che l'output della rete all'istante t sia la predizione del valore all'istante $t+1$. Se si vuole prevedere a più istanti n nel futuro si può usare lo stesso classificatore ma utilizzando come label il valore all'istante $t+n$.

L'output del classificatore ha dimensione pari agli input possibili della rete nel caso essi siano valori discreti, invece è pari al numero di bucket (intervalli usati per dividere l'input) nel caso essi sono valori scalari. La rete neurale presente nel classificatore utilizza una funzione di attivazione di tipo softmax perciò per ogni possibile uscita si avrà una probabilità che quella sia l'uscita predetta. La somma delle probabilità di tutte le uscite sarà uguale a uno. Ciò consente di avere più predizioni e di sapere per ciascuna con quanta probabilità sia quella corretta.

La seconda principale funzione per cui sono utilizzate queste reti è la rilevazione di anomalie su una sequenza di dati. Per come sono realizzati gli algoritmi non è possibile avere direttamente in output un indicatore che segnali dei valori inaspettati ma bisogna ricavarlo dallo SDR in output. L'errore di predizione viene ricavato con la seguente formula:

$$s_t = 1 - \frac{\pi(x_{t-1}) \cdot a(x_t)}{|a(x_t)|}$$

dove x_t è l'input all'istante t, $a(x_t)$ è la codifica di x_t ottenuta applicando lo spatial pooler se utilizzato o altrimenti è l'output dell'encoder, $\pi(x_{t-1})$ è la SDR che rappresenta la predizione della rete HTM fatta nell'istante $t-1$, $|a(x_t)|$ è la norma scalare del vettore $a(x_t)$, cioè il numero di bit attivi [13]. Al numeratore troviamo il prodotto scalare che serve per calcolare l'overlap tra i due SDR. s_t può così assumere valori tra 0, cioè predizione perfettamente effettuata, e 1, predizione totalmente errata. Se l'input è inaspettato e perciò è un'anomalia l'indicatore presentato avrà un valore alto. Dato che la rete si adatta al segnale anche esso si adatta, perciò un valore che in un certo momento della sequenza è un'anomalia poi potrebbe diventare la norma di conseguenza l'errore di predizione calerà.

Questo indicatore è abbastanza in alcuni scenari ma non è sufficiente nel caso di segnali molto disturbati e con un alto grado di rumore. Per questi segnali si verificano un sacco di falsi positivi. Ad esempio se un segnale è solitamente basso ma ogni tanto ha dei picchi molto alti che non possono essere predetti perché casuali ma comunque ammessi e non considerati anomalia, l'errore di predizione li segnalerebbe anche se non devono essere considerati anomalie. Per questi casi si risolve utilizzando l'anomaly likelihood, un indicatore che considera la distribuzione degli errori all'interno di una finestra W indicando la probabilità che un dato errore sia un'anomalia [13]. Il calcolo di tale indicatore è il seguente:

$$L_t = 1 - Q(\frac{\tilde{\mu}_t - \mu_t}{\sigma_t})$$

dove

$$\tilde{\mu}_t = \frac{\sum_{i=0}^{i=W'-1} s_{t-i}}{W'}$$

$$\mu_t = \frac{\sum_{i=0}^{i=W-1} s_{t-i}}{W}$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1} (s_{t-i} - \mu_t)^2}{W-1}$$

dove Q è l'approssimazione migliorata della funzione Q gaussiana [14],μ_t è la media e σ_t la varianza, entrambe aggiornate continuamente in base agli errori presenti nella finestra. W' è una piccola finestra contenente solo gli ultimi errori e ha una grandezza di molto più piccola di W. Con W' viene aggiunta la media $\tilde{\mu}_t$ per velocizzare la computazione e aggiornare meno costantemente la finestra W. Un'anomalia viene rilevata se

$$L_t \geqslant 1 - \epsilon \tag{1.1}$$

con ϵ opportunamente nell'intervallo $(0, 1)$.

1.3.6 Struttura a più livelli

Fino ad ora sono stati presentati i componenti base di un'architettura HTM e quelli per interfacciarcisi. Nella figura 1.10 ne è riportata una schematizzazione. Una rete HTM con un solo SP e un solo TM rappresenta un solo layer delle colonne che compongono la neocorteccia. Come si vedrà nel capitolo 2 ciò basterà a creare software in grado di risolvere diversi problemi. Però come nel cervello, unendo più regioni di per sé abbastanza semplici è possibile costruire algoritmi molto complessi e capace di risolvere problemi molto più elaborati rispetto al semplice riconoscimento di anomalie o predizione di una serie temporale. È possibile sviluppare architetture più complesse come mostrato nella figura 1.11 [4].

Un grado di complessità maggiore viene aggiunto nell'articolo in cui sono introdotte le grid cell nella teoria delle HTM [15]. Questo tipo di neuroni sono utilizzati solitamente per la localizzazione all'interno di ambienti ma nell'articolo viene proposto l'utilizzo di tali celle per definire oggetti, creare schemi mentali e altre complesse operazioni.

Oltre a queste strutture ispirate al cervello è possibile costruire architetture multi livello in base alla necessità. Ad esempio le HTM sono state utilizzate per reinforcement learning utilizzando l'architettura riportata in questo sito [23].

Figura 1.10: Schematizzazione di tutti gli step per utilizzare una rete HTM in un problema generico [22]. Il riquadro con il contorno verde indica la rete HTM ed è il punto in cui l'SP e il TM sono implementati.

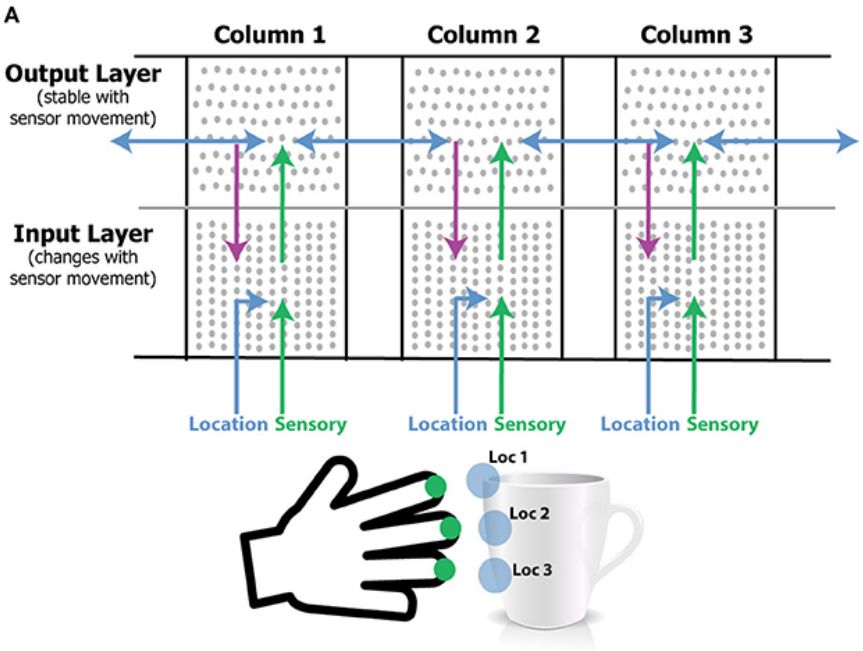

Figura 1.11: Modello di rete complesso utilizzato per il riconoscimento di oggetti attraverso tre sensori. Ciascuna colonna è composta da due strati. Il primo chiamato input layer riceve dati dai sensori, il secondo chiamato output layer riceve solo i dati dallo strato sottostante e dalle colonne adiacenti. Con questa struttura ogni colonna si specializza sui dati ricevuti da un singolo input ma dato che ciascuna colonna è connessa alle altre le informazioni vengono condivise e l'apprendimento avviene più velocemente rispetto al caso in cui venga utilizzata una sola colonna.

Capitolo 2

Applicazioni

In questo capitolo verranno presentate alcune applicazioni pratiche delle reti HTM. In tutti i casi è stata usata una rete HTM base, cioè un solo layer con uno spatial pooler e un temporal memory. Ciò corrisponde ad un solo strato delle colonne che costituiscono la neocorteccia ma comunque sufficiente a risolvere alcune tipologie di problemi.

Per tutte le applicazioni presentate è stata utilizzata la Numenta platform for Intelligent Computing (NuPIC) [24]. La piattforma è stata sviluppata da Numenta e fornisce strumenti per utilizzare HTM pronte all'uso o per crearne di nuove rivolte a casi specifici. Inoltre fornisce diverse tipologie di encoders e strumenti per elaborare gli output della rete o velocizzare le operazioni di input e output da file. È stata utilizzata la versione 1.0.3, la più recente disponibile al momento dello sviluppo. Questo framework è disponibile per diversi linguaggi di programmazione ma il più utilizzato e mantenuto è quello in python che è stato il linguaggio utilizzato nelle applicazioni presentate. La versione di Python utilizzata è la 2.7.15, l'unica supportata attualmente dal framework. In realtà lo SP e il TM vengono implementati in C++ dal framework per ottenere prestazioni migliori.

Il codice delle seguenti applicazioni è liberamente disponibile all'interno di un repository Github(*https://github.com/MesSem/HTM-for-Pattern-Recognition*). Parte del software è disponibile attraverso degli script in python, parte è organizzato all'interno di un notebook Jupyther. I grafici sono stati effettuati con Plotly.

2.1 Rilevazione di anomalie

Una delle applicazioni su cui le reti HTM base danno migliori risultati è la rilevazione di anomalie all'interno di serie temporali. Da qui in poi si indicherà con HTM base le reti composte da un solo livello contenente uno Spatial Pooler e un Temporal Memory. Per dimostrare il funzionamento di tale tecnologia sono stati utilizzati due dataset reperiti dall'UCI Machine Learning Repository [17]. Gli stessi dataset sono stati precedentemente impiegati nella valutazione del modello Markovmodulated Poisson process(MMPP), confrontato con un semplice modello base per la rilevazione di anomalie chiamato Threshold Model in [16].

Il primo dataset contiene 3 mesi di dati registrati nell'edificio Calit2 all'interno del campus Uc Irvine dell'University of California. Sono stati prodotti tramite un sensore posto nella porta principale che conta il numero di persone che entrano e escono dall'edificio. La memorizzazione dei dati avviene ogni 30 minuti. L'ingresso considerato è solo uno dei sette disponibili nel complesso, ma dato che all'incirca il 50% utilizza

questo accesso, si possono considerare i dati registrati una buona approssimazione di tutti gli ingressi e le uscite. Le anomalie da rilevare sono gli eventi che si verificano nello stabile come conferenze, riunioni o altro che causano un maggior numero di ingressi prima dell'evento e un maggior numero di uscite alla fine di esso. Gli eventi indicati nel dataset utilizzati per validare i risultati sono una sottostima del vero numero di eventi perciò nella valutazione si darà più importanza alle anomalie correttamente rilevate che ai falsi allarmi che in realtà potrebbero essere eventi non programmati.

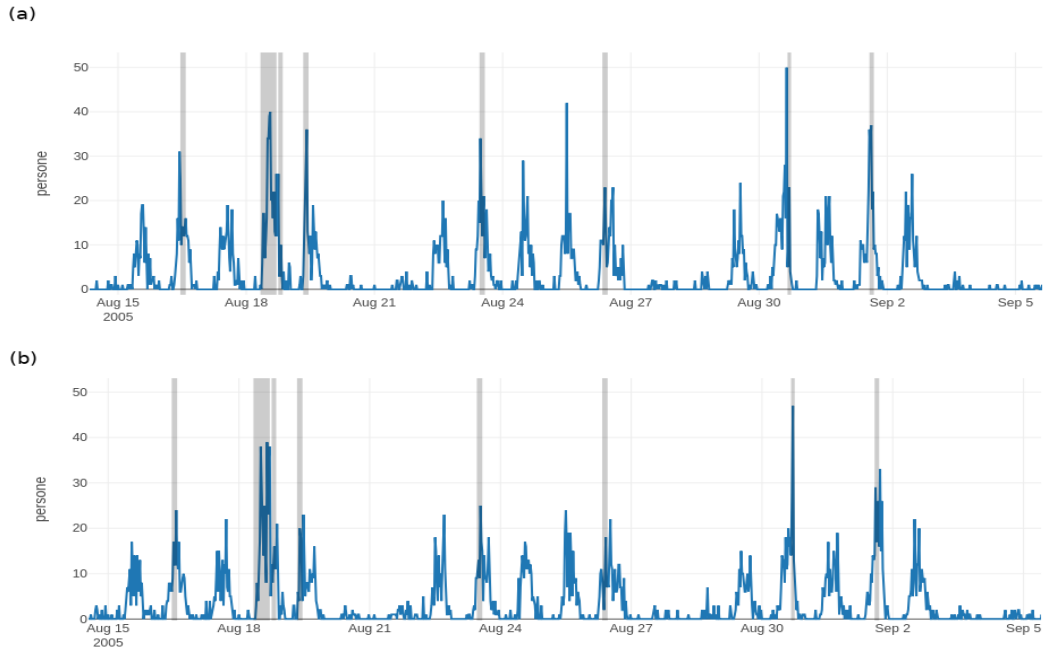

Figura 2.1: Grafici di un intervallo dei dati considerati. (a) Ingressi nell'edificio Calit2 dal 15 agosto al 5 settembre. (b) Uscite dall'edificio Calit2 dal 15 agosto al 5 settembre. In entrambi i grafici, le sezioni in grigio chiaro indicano i periodi temporali in cui si sono svolti degli eventi. I momenti di maggiore attività nello stabile corrispondono alle ore diurne seguiti dai momenti di bassa attività, cioè le ore notturne. I momenti di maggiore attività sono visivamente divisibili in gruppi da cinque elementi. Ciascuno di loro corrisponde a una giorno della settimana, lo spazio di poca attività tra ciascun gruppo è il fine settimana dove gli ingressi e le uscite sono vicini allo zero. In alcuni casi, come nel 30 agosto, è facile notare che in un certo punto della giornata, in entrambi i grafici c'è un numero non consueto di persone che entrano e escono. Infatti in quel momento vi è un evento. In altre situazioni questa anomalia è meno visibile.

Una piccola sezione del dataset è riportata in figura 2.1. Sono disponibili due flussi di dati, gli ingressi e le uscite, ma solo quello delle persone che entrano nell'edificio è stato considerato. Si è provato ad utilizzare i dati riguardo le uscite ma si sono ottenuti risultati peggiori. Si è tentato di unire i due flussi di dati in diverse configurazioni. Sono stati sommati per ottenere il numero totale di persone che attraversano la porta. Sottratti per calcolare di quanto è cambiato il numero di persone nell'edificio. Si è anche fatta la media per ogni istante delle persone entrate e uscite. In qualsiasi caso il numero di eventi rilevati è stato minore di quello utilizzando solo gli ingressi.

L'input delle rete HTM è stato realizzato utilizzando un encoder multiplo composto

da due encoder. Il primo è per il numero di persone che entrano nell'edificio, esso è di tipo scalare e ha come parametri $n = 100$ e $w = 21$. Il secondo encoder è un encoder realizzato per le date che si appoggia ad un encoder scalare. Ha come parametri $w = 21$ e $r = 4,5$. Il parametro r serve per indicare la quantità oltre il cui due dati vengono considerati totalmente differenti. Se $r = 2$, i valori che hanno una differenza di 2 vengono considerati totalmente differenti cioè con overlap pari a zero. Nell'encoder utilizzato, perciò, l'ora del giorno viene codificata in modo tale che due ore che distano tra loro più di quattro ore e mezza non hanno bit in comune. Dai grafici nelle figura 2.1 è possibile notare la periodicità dei dati. Essa si verifica su scala giornaliera dato che ogni giorno nelle ore diurne vi è un incremento di ingressi e un calo nelle ore notturne. Questa caratteristica viene presa in considerazione dall'encoder sopra specificato. La periodicità settimanale si verifica dato che nei weekend vi è un netto calo di attività rispetto ai cinque giorni precedenti. In uno dei tentativi è stato utilizzato un encoder per i giorni della settimana e uno per il fine settimana ma i risultati ottenuti sono stati peggiori di quelli senza i due encoder. È possibile che troppi dati in input creino rumore e rendano più difficile alla rete imparare tali pattern.

Per ciascun evento è indicata la data e l'ora di inizio e quella di fine. Durante l'analisi dei dati e dei primi risultati ottenuti si è visto che i picchi delle anomalie talvolta accadevano subito prima o subito dopo gli eventi. Si può presumere che ciò avvenga perché gli eventi possono durare un pò di più del programmato o che la persone si presentino in anticipo. Per non tralasciare queste anomalie si è deciso di considerare che un evento è correttamente rilevato se l'anomalia viene trovata da 35 minuti prima dell'inizio dell'evento ad un'ora dopo la fine.

L'algoritmo sviluppato considera un segnale anomalo nel caso la disuguaglianza 1.1 abbia esito positivo. Se tale disuguaglianza rimane verificata per più istanti di tempo successivi, una sola anomalia viene considerata.

L'anomaly likelihood e l'anomaly score calcolati per una sezione di dati sono visibili nella figura 2.2. L'anomaly likelihood è molto meno rumoroso nell'anomaly score che da solo restituirebbe molti falsi positivi. I risultati sono visibili in 2.3. L'accuratezza è stata calcolata in base a quanti eventi sono stati correttamente predetti rispetto al numero totale di eventi previsti. Per ottenere il numero esatto di eventi totali predetti riportato nella tabella è stato adattato il valore di soglia ϵ. La rete HTM è in linea con i risultati del modello MMPP, è migliore solo nel caso in cui il numero totale di anomalie predette siano 48.

Il calcolo dell'anomaly likelihood utilizza 388 elementi per l'apprendimento e perciò esso è utilizzabile solo dopo questi primi valori. Dato che i primi due eventi si trovano all'interno di questo intervallo, essi non possono essere predetti dalla rete HTM, perciò in un tentativo si è dato in input alla rete i primi 388 elementi per due volte. Considerando questi primi elementi l'accuratezza è comunque in linea con il modello MMPP e sempre migliore del modello Threshold. L'ultima riga riporta il numero minimo di eventi rilevati per raggiungere l'accuratezza del 100%. Questo indicatore non è riportato nell'articolo originale perciò non è possibile fare dei paragoni.

Il secondo dataset contiene dei dati riguardanti il traffico di una strada collegata all'autostrada 101-North di Los Angeles situata vicino allo stadio di baseball dei Dodger [18]. I dati vengono registrati ogni cinque minuti da un sensore del traffico per sei mesi consecutivi. Gli eventi utilizzati per la validazione sono state le partite di baseball tenutesi nello stadio nelle vicinanze. Prima di questi e sopratutto alla fine di essi è possibile notare un incremento del traffico dovuto ai tifosi che lasciano lo stadio. Anche

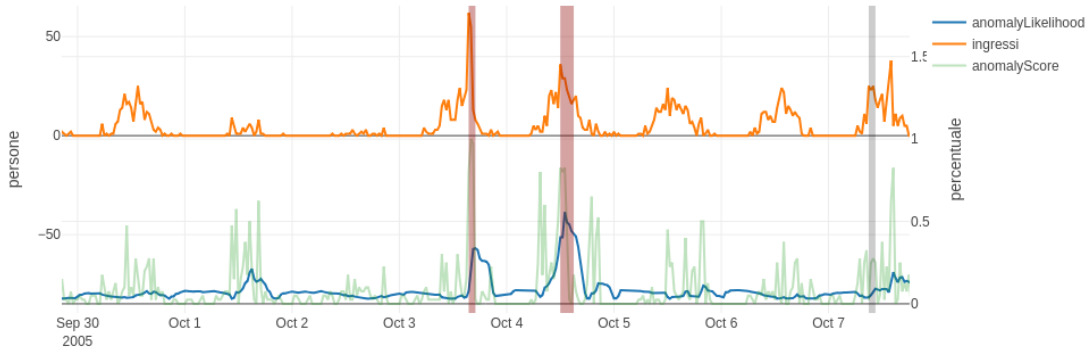

Figura 2.2: Grafico degli ingressi, dell'anomaly score e dell'anomaly likelihood. In arancione sono indicati gli ingressi nell'edificio e fanno riferimento alla scala sulla sinistra. L'anomaly score e l'anomaly likelihood, rispettivamente verde chiaro e blu, fanno riferimento alla scala sulla destra. Le sezioni in rosso chiaro evidenziano gli eventi correttamente predetti, in grigio invece quelli che non sono stati predetti. Nel caso in figura un evento viene predetto se l'anomaly likelihood è maggiore di 0.105, ciò accade tra l'uno e il due, tra il tre e il quattro e tra il cinque e il sei ottobre. Nel primo caso si avrà un falso positivo. Dalla figura è possibile comprendere come l'anomaly score sia tropo rumoroso per essere utilizzato come indicatore di anomalie senza ulteriore elaborazione.

Numero totale di eventi predetti	Modello MMPP	Modello Threshold	Rete HTM	Rete HTM+388 elementi
104	100,00%	86,20%	100,00%	100,00%
70	96,60%	75,90%	96,30%	93,10%
48	79,30%	65,60%	88,88%	86,20%
Numero totale di eventi predetti per raggiungere il 100% di accuratezza			80	85

Figura 2.3: Tabella dei risultati sul dataset del campus universitario. Per ciascun modello è riportata l'accuratezza delle predizioni, cioè la percentuale dei 29 eventi conosciuti correttamente rilevati sul numero totale di eventi predetti.

in questo caso l'elenco degli eventi non è completo dato che considera solo le partite ma ignora altre possibili manifestazioni non sportive come concerti oppure situazioni che incrementano il traffico come incidenti stradali. Una sezione del dataset è riportato in figura 2.4. Dal grafico si nota che i dati sono molto rumorosi perciò si è tentato di raggruppare i gruppi di sei instanti ottenendo così il numero di veicoli in media transitati in mezz'ora. Come si può vedere dalla figura citata il secondo grafico è più pulito.

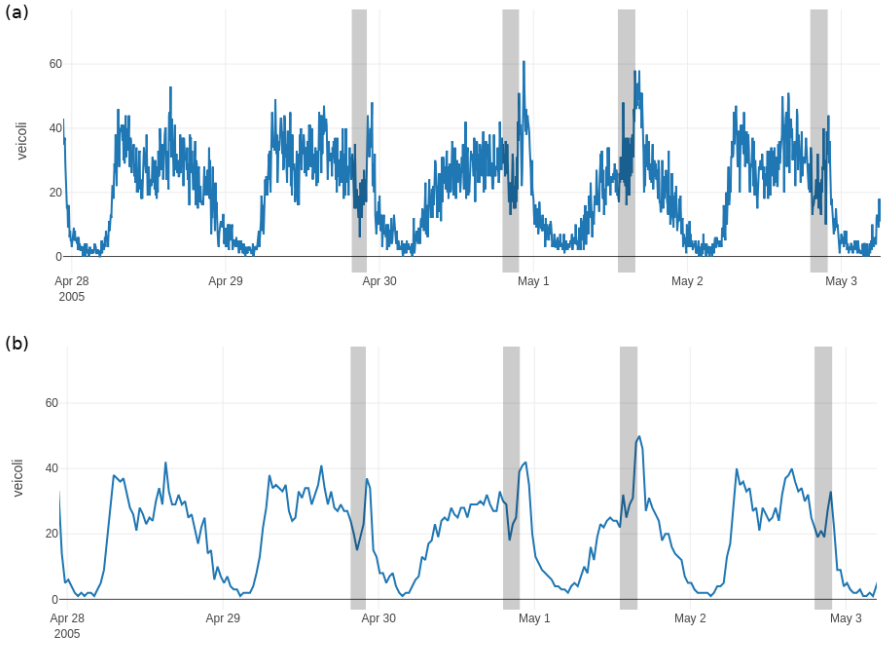

Figura 2.4: Grafici di una sezione del dataset del traffico stradale. (a) Veicoli in transito nella strada monitorata registrati con una cadenza di cinque minuti. (b) Numero di veicoli nella strada monitorata in media per ogni mezz'ora. In entrambi i grafici, le sezioni in grigio chiaro indicano i periodi temporali in cui si sono svolti degli eventi nello stadio dei Dodgers. È possibile distinguere l'attività diurna da quella notturna in cui le macchine in transito sono nettamente minori. Nei giorni del 30 aprile e del primo maggio vi è una leggera diminuzione del traffico dato che sono giorni festivi. In corrispondenza degli eventi è possibile notare un incremento di attività dovuto all'arrivo o alla partenza dei tifosi.

In questo caso l'intervallo di registrazione dei dati è molto più lungo e ricopre un grande fetta dell'anno 2005. Si è tentato di usare come informazioni in input alla rete anche il giorno del mese e il giorno della settimana che, però, non si sono dimostrati utili alla predizione. L'input alla rete è stato generato attraverso un encoder multiplo costituito da tre encoder. Per codificare il numero di veicoli è stato utilizzato uno encoder numerico con $n = 300$ e $w = 21$ con i dati campionati ogni 5 minuti e $n = 200$ e $w = 41$ con i dati campionati ogni mezz'ora. Un altro è stato impiegato per l'ora del giorno con $w = 21$ e $r = 9.5$ e infine uno è stato utilizzato per codificare il weekend, cioè se la data codificata corrisponde a una fine settimana dei bit vengono attivati, altrimenti no. Quest'ultimo ha come unico parametro $w = 21$ cioè tutti i bit in output sono attivi oppure no. Come nel dataset precedente è possibile notare nel grafico come per ogni giorno si ripeta un pattern simile del traffico e come in corrispondenza degli eventi, vi siano delle anomalie.

L'anomaly score calcolato e l'anomaly likelihood rispettivi a una sezione di dati è visibile nel grafico in figura 2.5. I dati sono molto rumorosi perciò vi sono molte più anomalie indicate dall'anomaly score rispetto al dataset precedente. È stata utilizzata una finestra di larghezza minore per calcolare l'anomaly likelihood $W = 2000$ per i dati ogni cinque minuti e $W = 1000$ per i dati ogni mezz'ora rispetto a $W = 8640$ utilizzato

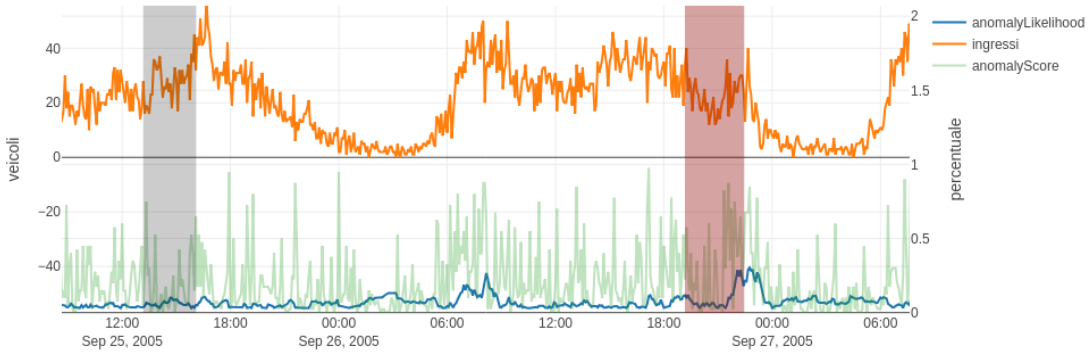

Figura 2.5: Grafico degli ingressi, dell'anomaly score e dell'anomaly likelihood. La sezione evidenziata in rosso indica che l'evento corrispondente è stato correttamente predetto. L'anomaly score è molto rumoroso perciò viene utilizzato l'anomaly likelihood per identificare le anomalie. Nel caso in figura si identificano le anomalie se il valore di anomaly likelihood supera il valore 0.16. Ciò avviene dopo le 6.00 del 26 settembre e dopo le 18.00 dello stesso giorno. Il primo caso è considerato un falso positivo. La sezione in grigio sulla sinistra del grafico non viene rilevato dall'algoritmo.

per il dataset dell'edificio universitario. È stata presa questa scelta perché una finestra troppo grande per il calcolo dell'anomaly likelihood permetteva che tutte le anomalie venissero ignorate. I risultati sono mostrati nella tabella in figura 2.6. L'ultima colonna della tabella riguarda il modello delle rete HTM applicato ai dati raccolti in istanti di 30 minuti. Raggruppando i dati si è ridotta la loro dimensione e alcuni eventi sono ricaduti nella prima parte del dataset e perciò non venivano considerati. Come nel caso precedente si è deciso di eseguire il modello su tutti i dati e poi rieseguirlo dandogli in input i primi 388 elementi. I risultati non sono allo stesso livello del modello MMPP. Però la rete HTM si comporta in linea con il modello Threshold nel caso che gli eventi totali predetti siano pari a 203, 186 e 134 ma è migliore nel caso che le anomalie totali rilevate siano 98.

Numero totale di eventi predetti	Modello MMPP	Modello Threshold	Rete HTM	Rete HTM con intervalli di 30 minuti	Rete HTM con intervalli di 30 minuti+388 elementi
203	100,00%	86,80%	81,57%	78,87%	78,94%
186	100,00%	81,29%	80,26%	78,87%	78,94%
134	100,00%	72,40%	75,00%	77,46%	75,00%
98	98,70%	60,50%	55,26%	71,83%	65,78%

Figura 2.6: Tabella dei risultati sul dataset riguardante il traffico. Per ciascun modello è riportata l'accuratezza delle predizioni, cioè la percentuale dei 76 eventi conosciuti correttamente predetti sul numero totale di anomalie rilevate.

Nel dataset dei dati provenienti dal traffico stradale è stato utilizzato il boosting. Si è notato che nei dati provenienti dall'edificio universitario questa opzione provocava una rilevazione molto alta di anomalie. Ciò avviene perché incentivando l'utilizzo di nuove minicolonne capita più spesso di compiere previsioni errate ma permette alla rete di imparare pattern più complessi nel lungo periodo. Probabilmente il primo dataset utilizzato non è abbastanza grande da ricevere guadagno da questa opzione. Invece nel secondo caso di studio il boosting è stato fondamentale. Senza di esso la rete non era in grado di imparare i complessi pattern producendo un numero molto alto di falsi positivi. Questo dimostra che il boosting è utile se si hanno input sufficientemente estesi.

2.2 Predizioni di consumi energetici

Un'altra funzione in cui le HTM possono essere utilizzate è la predizione di valori all'interno di una serie di dati. Per testare tale tipologia di applicazione è stato preso il dataset "ElectricityLoadDiagrams20112014 Data Set" dal repository [17]. Esso contiene il consumo il kw di 370 clienti dal 2011 al 2014. Per alcuni casi la data di inizio registrazione è successiva al 2011, per questi casi la prima parte dei dati, posta a zero, non viene considerata. I dati sono memorizzati con una cadenza di 15 minuti.

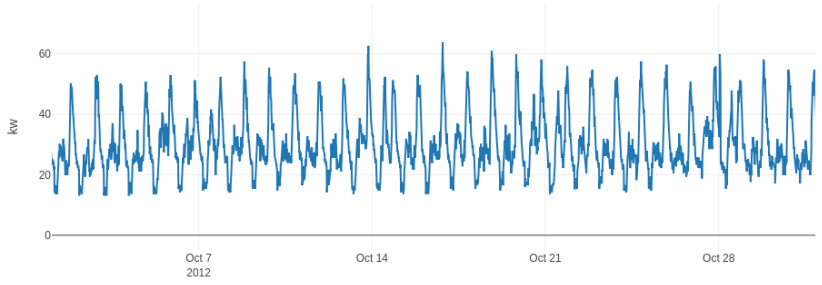

Figura 2.7: Grafico di un mese di dati del cliente numero 16

Il consumo di corrente elettrica di uno di questi clienti è riportato in figura 2.7. Il segnale segue dei pattern che si ripetono giornalmente ma è molto rumoroso. La codifica dei dati è stata effettuata sempre con un encoder multiplo per poter trarre informazioni dai diversi dati, cioè il consumo, la data e l'ora. Per il consumo è stato utilizzato un encoder numerico con $n = 100$ e $w = 21$. Differentemente dalle applicazioni precedentemente riportate non è stato impostato un massimo e un minimo fisso ma si è utilizzato l'AdaptiveScalarEncoder che adatta questi due valori automaticamente. Alla data è stato applicato solo l'encoder che estrapola l'informazione riguardo al fine settimana, cioè se la data in input corrisponde a un weekend l'ouptut saranno dei bit attivi, in questo caso 21. Infine l'ultima informazione data alla rete è l'ora del giorno codificata con un encoder apposito con parametri $w = 21$ e $r = 3$.

Il Classificatore SDR che trasforma l'output della rete HTM in un numero scalare restituisce una serie di possibili predizioni e per ciascuna indica la probabilità che sia quella corretta. Nella valutazione dell'algoritmo è stato considerato solo il valore con la probabilità più alta. In situazioni in cui più previsioni possono essere accettate si può utilizzare il negative log-likelihood per valutare la predizione come nell'articolo [12].

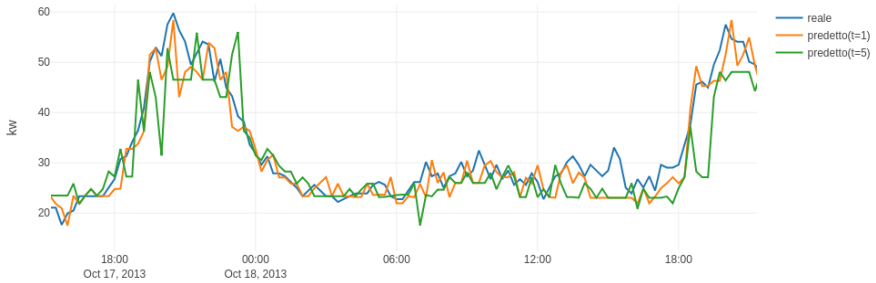

Figura 2.8: Confronto tra il valore reale e le predizioni ottenute dalla rete. La linea arancione indica le predizione realizzate ad un instante nel futuro, quella verde a cinque.

Sono state effettuate due tipo di predizioni, una ad un solo istante nel futuro, $t = 1$, l'altra a 5 istanti nel futuro, $t = 5$ come mostrato in figura 2.8. Anche se con differenti valori di accuratezza, entrambe riescono a prevedere il trend del consumo di corrente.

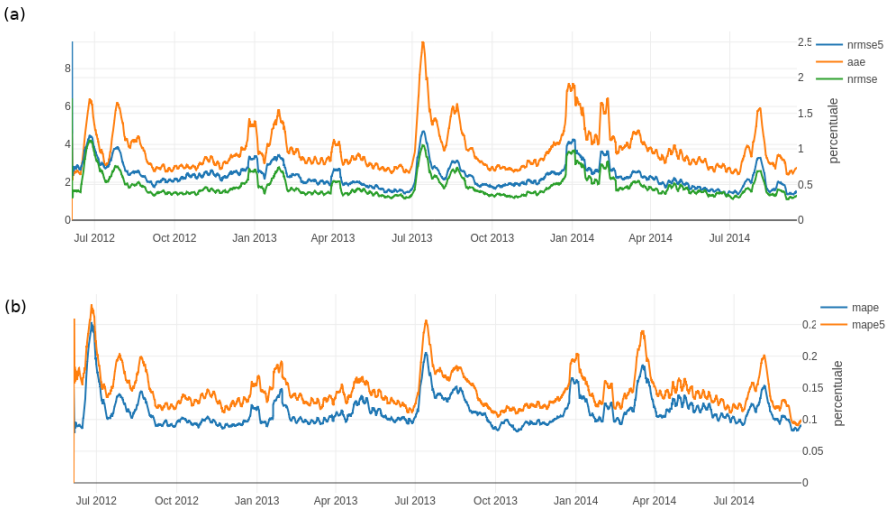

Figura 2.9: Grafici degli indicatori utilizzati per valutare l'algoritmo. (a) Confronto degli indici NRMSE, uno calcolato sulla predizione ad un istante nel futuro e uno a 5, con l'indice AAE che fa riferimento alla scala di valori sulla sinistra, gli altri due a quella sulla destra. (b) Grafico rappresentante il MAPE a 1 e 5 istanti nel futuro. Entrambi i valori sono molto piccoli ma quello calcolato sulle predizioni più a breve termine, quello blu, è drasticamente migliore dell'altro.

Una caratteristica interessante di questa rete è quella riscontrabile in figura 2.9. Nei grafici viene considerato un lungo lasso di tempo, all'incirca 80000 valori. Capita a volte che l'errore aumenti improvvisamente. Ciò è dovuto a una cambio nella tendenza dell'utilizzo della rete elettrica dovuto per esempio al cambio di stagione o ad altri fattori. L'errore aumenta sensibilmente dato che la rete non è allenata su questi nuovi valori in ingresso. Ma è interessante vedere come dopo aver ricevuto abbastanza nuovi dati in input apprende il nuovo pattern e l'errore torna a scendere. Questo è un grande

vantaggio delle HTM. In altri tipi di rete, come per esempio nelle long short-term memory (LSTM), se il pattern in input cambia la rete deve essere allenata sui nuovi dati in modo che apprenda il nuovo pattern. In questa caso invece la rete si adatta e non serve bloccarla.

cliente	5	6	7	8	9	11	13	14	16	17	21	23	28	29	33
mape	0,17	0,12	0,21	0,09	0,16	0,12	0,12	0,1	0,12	0,08	0,13	0,14	0,11	0,08	0,09
nrmse	0,59	0,5	0,14	0,51	0,42	0,47	0,32	0,36	0,44	0,42	0,62	0,56	0,47	0,3	0,39

Figura 2.10: Tabella dell'indice MAPE e RMSE calcolato su 10000 valori di alcuni clienti.

Per la valutazione della predizione sono state utilizzate diversi indicatori di errore: l'average absolute error (AAE), il normalized root-mean-square error (NRMSE) e il mean absolute percent error (MAPE). Nel grafico 2.9 sono visualizzati gli errori calcolati utilizzando una finestra di mille valori. Nella tabella in figura 2.10 invece sono mostrati il MAPE e il NRMSE calcolato usando una finestra di 10000 dati per 15 clienti. Essi non possono essere considerati come indicatori assoluti perché può essere capitato che nel momento in cui sono stati calcolati la rete si stava adattando a un nuovo pattern. Dai risultati ottenuti è possibile valutare positivamente la rete HTM dato che gli errori sono molto bassi.

2.3 Predizioni di attacchi epilettici

L'epilessia è una malattia neurologica che colpisce milioni di persone nel mondo. È caratterizzata da ricorrenti e improvvise manifestazioni con improvvisa perdita di coscienza e violenti movimenti convulsi dei muscoli, dette *crisi epilettiche*. Non esiste una cura che vada bene per tutti i tipi di epilessia, ma si può agire in diversi modi. Alcuni farmaci possono sedare le convulsioni ma non tutti i pazienti sono sensibili ai loro principi attivi; in questo caso si ricorre alla secondo opzione: l'intervento chirurgico di rimozione del fuoco epilettogeno. Anche questa possibilità, però, potrebbe non essere applicabile a causa della presenza del fuoco in una zona eloquente. Infine, in questi ultimi anni, stanno facendo la loro comparsa anche dei dispositivi impiantabili possono sopprimere la crisi, tipo dei pacemaker cerebrali. L'attività cerebrale può essere registrata con diverse tecniche ma quella più utilizzata perché meno invasiva è l'elettroencefalogramma EEG che si compie posizionando sul cuoio capelluto degli elettrodi che misurano le variazioni di tensione dovute al flusso di correnti ioniche all'interno del cervello.

La previsione di questi attacchi con sufficiente anticipo è molto utile per migliorare la vita ai pazienti che ne soffrono. Questo problema è largamente affrontato dalla comunità scientifica. Negli ultimi anni con l'avvento del deep learning e del machine learning tanti approcci sono stati proposti per la previsione delle crisi in modo automatico senza l'utilizzo di personale specializzato. Alcuni di questi sono specifici e vanno settati per ogni paziente in maniera diversa altri sono generalizzati, cioè utilizzano lo stesso algoritmo per ogni paziente. Quest'ultima soluzione è molto più vantaggiosa perché rende più fruibile e non ha bisogno di esperti del settore per impostare per ciascuna persona i giusti parametri. Dato che gli EEG possono essere considerati delle serie di dati temporali si è pensato di usare le reti HTM per questa applicazione. Fin'ora ci

sono stati vari tentativi di utilizzo delle reti HTM per risolvere questo problema ma non si sono riscontrati buoni risultati.

Un risultato molto buono presente in letteratura è quello in cui si utilizza un approccio basato sulle reti neurali di convoluzione, in inglese Convolutional Neural Network (CNN) [19]. Dati i buoni risultati si sono utilizzati gli stessi dati per poter effettuare un paragone e la stessa tecnica di elaborazione dei segnali. Dei dataset presentati nell'articolo è stato utilizzato solo quello del Boston Children's Hospital-MIT (CHB-MIT) che contiene dati EEG di 23 pazienti di cui è stato preso in considerazione solo il primo. I dati sono presentati come serie temporali. Sono registrati da 22 elettrodi con una frequenza di campionamento di 256 Hz. La parte di segnale registrato da 4 ore dopo la fine di una crisi epilettica fino a 4 ore prima di una crisi viene definito interictale. Invece si definisce pre-ictale quello ottenuto da 35 minuti prima a 5 minuti prima di una crisi.

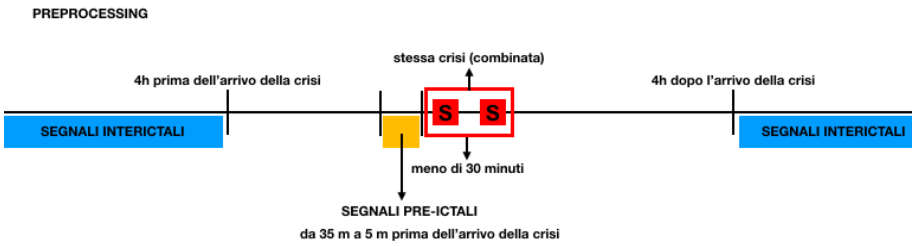

Figura 2.11: Segnali pre-ictali e interictali

L'obiettivo è di classificare la tipologia di segnale. Se viene rilevato un segnale pre-ictale il programma deve dare un allarme. È stato scelto questo intervallo per il segnale pre-ictale per dare sufficientemente tempo, almeno 5 minuti, al medico o chi di competenza di agire e per non allarmare troppo a lungo il paziente cioè al massimo 35 minuti. Se due crisi avvengono con una distanza di meno di 30 minuti vengono considerate un'unica crisi. Solo le sezioni di dati che possono essere classificate in uno dei due modi indicati sono utilizzati, il resto dei dati è ignorato.

Sul dataset è stata effettuata un'operazione di preprocessing per rimuovere alcune bande di frequenza condizionate dai disturbi. Tramite dei filtri di banda sono stati tolti gli intervalli di frequenza 57-63 Hz, 117-123 Hz e la componente a 0 Hz. Successivamente attraverso la Short Time Fourier Transform (STFT) si sono realizzati degli spettrogrammi considerando come risoluzione temporale mezzo secondo. Il risultato è che per ogni finestra di tempo si ha uno spettrogramma per ciascun canale.

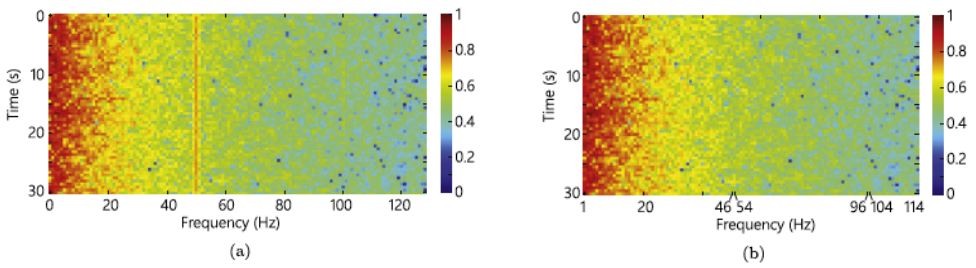

Figura 2.12: Esempi di STFT con finestra a trenta secondi prima (a) e dopo (b) la rimozione delle frequenze

In un primo tentativo sono stati utilizzati i dati grezzi come input per una rete HTM ma si è dimostrata subito una scelta non felice. La quantità dei valori in input nella soluzione considerata era molto grande dato che le registrazioni sono state effettuate per diverse ore e il campionamento è stato effettuato 256 volte al secondo. Perciò, per elaborare pochi minuti di dati, la rete impiegava diverse ore. Per questo motivo si è deciso di trasformare i dati in spettrogrammi. Per comodità di utilizzo sono stati salvati in blocchi di 30 secondi. Alla rete HTM vengono mandati tutti consecutivamente, come se venissero concatenati in un unico spettrogramma.

La rete HTM dovrebbe segnalare un basso livello di anomalie nel segnale interictale e un alto numero nel segnale pre-ictale, cioè poco prima della crisi. Dato che ciascun canale registra i dati da punti diversi del cervello, dei valori potrebbero essere considerati nella norma se registrati da un certo sensore ma potrebbero essere invece un'anomalia in un altro. Perciò per ogni canale si utilizza un'apposita HTM.

Dal preprocessing si ottengono spettrogrammi con 114 frequenze ma le onde che hanno origine nella parte più esterna del cervello e rilevabili con l'EEG raramente superano i 50 Hz perciò si è deciso di considerare solo questo intervallo di dati. La codifica dello spettrogramma viene svolta da un encoder multiplo che contiene al suo interno 50 encoder numerici, uno per ogni frequenza del segnale. Questa scelta permette di ridurre la dimensione dell'input e ottenere risultati migliori come è accaduto nelle applicazioni mostrate precedentemente in cui diminuire il numero degli ingressi risultava in un incremento dell'accuratezza. L'encoder per ciascuna frequenza ha come parametri $n = 60$ e $w = 21$.

A causa del tempo di esecuzione troppo lungo, solo 15 dei 22 canali del primo paziente sono stati selezionati nel test. Perciò i risultati ottenuti non possono essere considerati pienamente affidabili. Del paziente considerato si hanno 14,49 ore interictali e tre ore e mezzo di ore pre-ictali dovute ai 7 attacchi registrati. Due parametri vengono presi in considerazione per la valutazione delle performance della rete, la sensitività, definita come la percentuale di crisi epilettiche correttamente predette diviso il numero totale delle crisi e l'FPR, definito come il numero di falsi allarmi per ora. Con la rete HTM, utilizzando $\epsilon = 0,9$ per l'identificazione delle anomalie, vengono rilevati 7 attacchi correttamente e 55 erroneamente. Perciò la sensitività è pari al 100% mentre l'FPR è pari a 3,79. Questi valori non sono paragonabili all'articolo perché vengono utilizzati altri modelli per il loro calcolo non applicabili all'output ottenuto con le HTM. Però indicano che è possibile affrontare questo problema con le HTM. I risultati ottenuti per l'attacco numero due del paziente uno sono esposti in figura 2.13.

In una situazione reale questo approccio potrebbe avere dei problemi di esecuzione dovuti all'alto uso della memoria. Per essere utilizzato su dati in tempo reale bisognerebbe eseguire tutti i modelli in contemporanea e paragonare i risultati di ciascun modello per mandare o no un allarme. Non sono riuscito a compiere ciò in un computer con 16 Gb di RAM perché la memoria diveniva saturata alla creazione dell'undicesimo modello corrispondente all'undicesimo canale. Avendo dati storici il problema è stato risolto eseguendo un modello alla volta. La soluzione per un'esecuzione in tempo reale potrebbe essere quella di eseguire ciascun modello per un piccolo intervallo di tempo di pochi secondi. Un pò come avviene in un'architettura mono processore in cui ogni processo viene eseguito per un piccolo lasso di tempo dando l'impressione che più processi siano eseguiti in parallelo.

Un miglioramento ulteriore si potrebbe apportare utilizzando una struttura multi layer ottenendo i vantaggi riportati nel capitolo riguardo alla struttura a più livelli. In

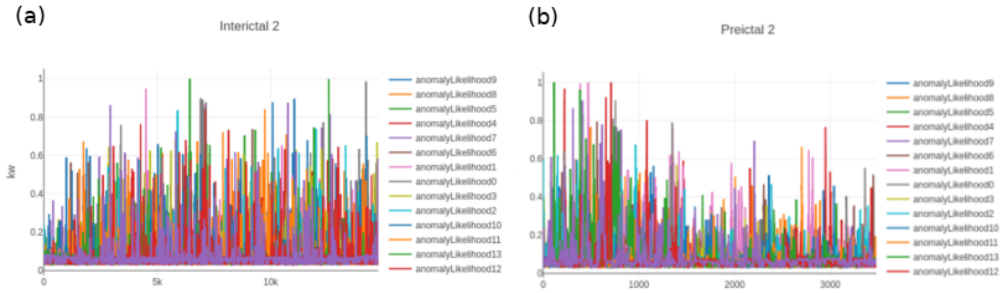

Figura 2.13: Risultati della rete HTM applicata a dei dati pre-ictali e interictali. Nei grafici vi sono l'anomaly likelihood ottenuto da diversi modelli, ciascuno applicato a un canale differente. In (a) sono utilizzati dati interictali, in (b) dati pre-ictali.

quel modo tutti i canali potrebbero essere dati in input contemporaneamente e i più livelli consentirebbero un'astrazione maggiore e la rilevazione di pattern più complessi che con un solo livello sfuggono.

Capitolo 3

Conclusioni

In questo elaborato è stata riportata un'analisi del modello HTM, le componenti basi ed alcuni cenni teorici sugli argomenti della biologia da cui il modello prende ispirazione.

Nella seconda parte sono state proposte alcune applicazioni di questa tecnologia. In tutte le applicazioni è stata utilizzata un'architettura a layer singolo cioè con un solo livello di SP e di TM. Questo approccio si è dimostrato sufficiente e ha dato ottimi risultati nella prima applicazione presentata. Risultati meno buoni si sono ottenuti con il dataset del traffico. Anche nella predizione la rete HTM si è dimostrata un buono strumento in grado di apprendere un trend e di compiere delle previsioni abbastanza accurate. Inoltre la rete HTM utilizzata sul dataset del consumo di corrente elettrica ha dimostrato una delle potenzialità di questa tecnologia cioè l'apprendimento online e la capacità di adattarsi al variare dell'input in tempo reale.

La limitazione di questa architettura si è rivelata con l'applicazione ad un problema più complesso quale la predizione di attacchi epilettici. I lati negativi riguardano sia i risultati che l'esecuzione dell'algoritmo che si è rilevato più lento del previsto. L'esito ottenuto nei tentativi svolti dimostra, però, che è possibile usare le HTM per questo problema a patto di modificare l'architettura. In futuro si potrebbero utilizzare sistemi più elaborati composti da più strati che dovrebbero dare soluzioni migliori. Questi ultimi possono risolvere problemi più complessi. Con l'evoluzione della teoria delle HTM nuovi elementi vengono aggiunti continuamente rendendo le HTM sempre più simili alla neocorteccia, sempre più capaci di emularla e affrontare situazioni complesse.

Bibliografia

[1] Hawkins, J., & Blakeslee, S. (2007). *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines.* Macmillan.

[2] Hawkins, J. et al. 2016. *Biological and Machine Intelligence.* Release 0.4. Accessed athttps://numenta.com/resources/biological-and-machine-intelligence/.

[3] Zingg, B. (2014) *Neural networks of the mouse neocortex.* Cell, 2014 Feb 27;156(5):1096-111. doi: 10.1016/j.cell.2014.02.023

[4] Hawkins J, Ahmad S and Cui Y (2017) *A Theory of How Columns in the Neocortex Enable Learning the Structure of the World.* Front. Neural Circuits 11:81. doi: 10.3389/fncir.2017.00081

[5] Hawkins, J., & Ahmad, S. (2016). *Why neurons have thousands of synapses, a theory of sequence memory in neocortex.* Frontiers in neural circuits, 10, 23.

[6] Ahmad, S. & Hawkins, J. (2015). *Properties of sparse distributed representations and their application to hierarchical temporal memory.* arXiv preprint arXiv:1503.07469.

[7] Purdy, S. (2016). *Encoding data for HTM systems.* arXiv preprint arXiv:1602.05925.

[8] Webber, F. E. D. S., & De Sousa, E. (2015). *Semantic Folding and Its Application in Semantic Fingerprinting.* Cortical. io White Paper, Version, 1.

[9] Cui Y, Ahmad S and Hawkins J (2017) *The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding.* Front. Comput. Neurosci. 11:111. doi: 10.3389/fncom.2017.00111

[10] O'Shea, K., & Nash, R. (2015). *An introduction to convolutional neural networks.* arXiv preprint arXiv:1511.08458.

[11] Davis, G. W. (2006). *Homeostatic control of neural activity: from phenomenology to molecular design.* Annu. Rev. Neurosci., 29, 307-323.

[12] Cui, Y., Ahmad, S., & Hawkins, J. (2016). *Continuous online sequence learning with an unsupervised neural network model.* Neural computation, 28(11), 2474-2504.

[13] Ahmad, S., Lavin, A., Purdy, S. & Agha, Z. (2017). *Unsupervised real-time anomaly detection for streaming data.* Neurocomputing, 262, 134-147.

[14] Karagiannidis, G. K. & Lioumpas, A. S. (2007). *An improved approximation for the Gaussian Q-function.* IEEE Communications Letters, 11(8).

[15] Hawkins, J., Lewis, M., Klukas, M., Purdy, S., & Ahmad, S. (2018). *A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex.* bioRxiv, 442418.

[16] Ihler, A., Hutchins, J. & Smyth, P. (2006, August). *Adaptive event detection with time-varying poisson processes. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 207-216). ACM.

[17] Dua, D. and Karra Taniskidou, E. (2017). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[18] *Freeway Performance Measurement System (PeMS)*, "http://pems.eecs.berkeley.edu"

[19] Truong, N. D., Nguyen, A. D., Kuhlmann, L., Bonyadi, M. R., Yang, J., Ippolito, S., & Kavehei, O. (2018). *Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram.* Neural Networks, 105, 104-111.

Siti consultati

[20] Numenta inc., http://www.numenta.com consultato il, 19.10.2018

[21] https://youtu.be/BvJJn9VS4rk consultato il, 19.10.2018

[22] https://www.slideshare.net/ibobak/hierarchical -temporal-memory-for-realtime- anomaly-detection consultato il 16.10.2018

[23] https://discourse.numenta.org/t/exploring-reinforcement -learning-in-htm/1545 consultato il, 20.10.2018

[24] http://nupic.docs.numenta.org/1.0.3/index.html consultato il, 17.10.2018