Relazione progetto Sistemi Intelligenti

Corso di Laurea Magistrale in Ingegneria Informatica
Anno Accademico 2012-2013

Stefano Cicero

Mattia Ridolfi

1 Introduzione

Il progetto prevede l'analisi di un insieme di dati allo scopo di applicare tecniche di computational intelligence per la valutazione dei consumi energetici di un edificio, adibito ad ufficio, relativamente all'impianto di condizionamento.

Lo scopo è quello di valutare il consumo energetico sostenuto per raffrescare o riscaldare un ambiente all'interno dell'edificio, conoscendo la situazione ambientale esterna all'edificio.

Con situazione ambientale esterna si intende la condizione definita da un insieme di parametri ambientali tra cui l'irraggiamento solare, la temperatura esterna, l'umidità esterna. I dati includono la data e l'orario del giorno.

Ovviamente il consumo dipende anche dall'utilizzo stesso dell'ambiente da parte degli occupanti che agiscono direttamente sull'impianto di condizionamento per ristabilire il confort termico. Ad esempio, non appena la temperatura interna all'edificio diventa troppo alta, gli occupanti accendono l'impianto di condizionamento incrementando il consumo energetico.

La presente relazione si articola principalmente in tre punti: il primo consiste in un'analisi e in seguito ottimizzazione dei dati che ci sono stati forniti, per migliorare le performance delle reti neurali che andremo a utilizzare in seguito. I punti successivi prevedono la predizione del consumo energetico dell'edificio tramite i seguenti strumenti forniti dal Matlab:

- Neural Network Time Series Tool
- Fuzzy Logic Tool
- Anfis Editor

2 Analisi dei dati

I dati ci sono stati forniti in forma tabellare all'interno di un foglio di calcolo Excel. Le righe della colonna costituiscono i singoli campioni, mentre le colonne rappresentano i vari attributi di ciascun campione.

In particolare gli attributi presenti sono:

- Timestamp: è l'istante di campionamento ed è costituito da anno, mese, giorno, ora, minuti e secondi
- Irraggiamento solare: esprime la quantità di radiazione solare per unità di superficie (espresso in Watt/m^2);
- Consumo energetico: è espresso in funzione della potenza attiva media relativa all'intervallo considerato ed è misurata in Watt;
- Temperatura esterna espressa in °C;
- Umidità esterna espressa in %;

I valori di timestamp, irraggiamento solare, temperatura esterna e umidità esterna costituiranno l'input della rete neurale, mentre i valori di consumo energetico saranno il target. I campioni fornitici coprono un intervallo di 5 mesi, che va dal 1 Giugno 2005 al 31 Ottobre 2005. La frequenza di campionamento è di un campione ogni mezzora, pari a $\frac{1}{1800}$ Hertz.

Come detto nel capitolo introduttivo, prima di dare in ingresso questi dati a una rete neurale per addestrarla, occorre eseguire delle analisi su di essi. Bisogna infatti realizzare una feature selection al fine di usare il numero minimo di variabili di ingresso che fornisca buone prestazioni in quanto il contributo di una variabile potrebbe solo aumentare la complessità della rete senza dare un significativo contributo. Le migliori variabili d'ingresso saranno quelle maggiormente correlate con l'energia consumata, che costituisce il valore che la rete deve predire.

Da un'ispezione dei dati fornitici, abbiamo constatato che questi ultimi sono incompleti: su 7344 campioni, 313 (4,26% circa) presentano valori di attributi mancanti. Per colmare la mancanza di tali valori è stata effettuata una interpolazione (tramite Microsoft Excel) facendo la media tra il valore dell'attributo precedente e quello successivo. Inoltre, osservando il grafico relativo alla temperatura esterna abbiamo riscontrato un valore anomalo in un determinato campione: in Figura 1 è riportato l'andamento della temperatura rilevato il 5 di luglio. Nell'asse x è indicata l'ora del giorno, mentre sull'asse y la temperatura registrata. Come è facile notare, alle ore 16:30 risulta una temperatura di ben 52° C che si discosta parecchio dall'andamento della temperatura nei campioni precedenti e successivi. In particolare la temperatura alle ore 16:00 è di 29,4° C mentre alle ore 17:00 è di 29,3° C. Lo sbalzo termico sarebbe di 23° C nel giro di mezzora. Ciò è molto improbabile che si verifichi nella realtà, pertanto abbiamo ipotizzato ad un errore nel campionamento e abbiamo sostituito il valore di 52° C con 30° C.

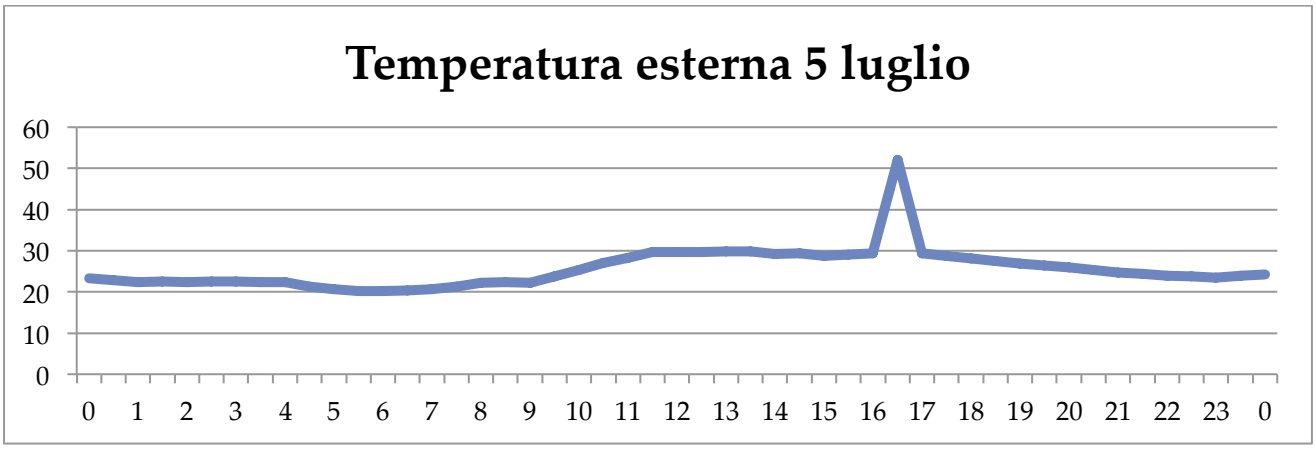
Temperatura esterna 5 luglio
60
50
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0

Figura 1

3 Svolgimento con Neural Network Time Series

3.1 Preparazione dei dati

Una volta attestata l'integrità generale dei dati possiamo passare all'analisi dei singoli attributi in modo da valutarne l'utilità ai fini della predizione del consumo energetico con la Neural Network Time Series.

Per cominciare, possiamo banalmente constatare che l'attributo "secondi" non è di nessun aiuto per l'obiettivo che dobbiamo raggiungere poiché i campioni hanno una risoluzione di 2 ogni ora pertanto i secondi possono essere tranquillamente trascurati.

Per quanto riguarda i restanti attributi facenti parte del timestamp possiamo dire che presi così come sono non danno un significativo contributo ai fini della predizione. Possiamo arguire che la data e l'ora influiscono sulla temperatura, sull'umidità e sull'irraggiamento che a loro volta influiscono sul consumo di potenza, pertanto non sarebbe necessario mantenere come ingresso anche il timestamp.

Leggendo attentamente le specifiche forniteci, è possibile osservare che i dati sono relativi ad un edificio adibito a ufficio. Un ufficio solitamente rispetta degli orari di apertura e chiusura, che determinano la presenza o meno di persone al suo interno. Quindi, se un campione è relativo a un orario di chiusura, possiamo avere la ragionevole certezza che il consumo di potenza sia pari a 0 poiché all'interno dell'edificio non sono presenti persone. Viceversa, se un campione è relativo a un orario di apertura, c'è la possibilità che il consumo medio di potenza attiva sia diverso da 0.

Pertanto partendo dal timestamp è possibile sapere se si tratta di un giorno lavorativo o feriale, o ancora meglio, se l'orario è lavorativo o feriale. Osservando in quali orari e giorni si verificano i consumi abbiamo deciso di considerare come orario lavorativo il seguente: dal lunedì al venerdì, dalle 8:00 alle 19:00, che risulta essere un orario lavorativo standard. Stabilito ciò, abbiamo creato uno script in linguaggio Python (Allegato 1) che aggiunge un attributo "orario lavorativo" che vale 1 se il campione appartiene all'orario lavorativo, 0 viceversa.

I restanti attributi ("temperatura esterna", "umidità" e "irraggiamento") si riferiscono a parametri ambientali esterni che possono ragionevolmente influenzare il consumo di potenza media nell'ufficio.

Come detto in precedenza, le migliori variabili d'ingresso sono quelle maggiormente correlate con l'energia consumata. Quindi per decidere quali attributi mantenere come input per la rete abbiamo applicato la funzione di correlazione i cui risultati sono riportati in Figura 2.

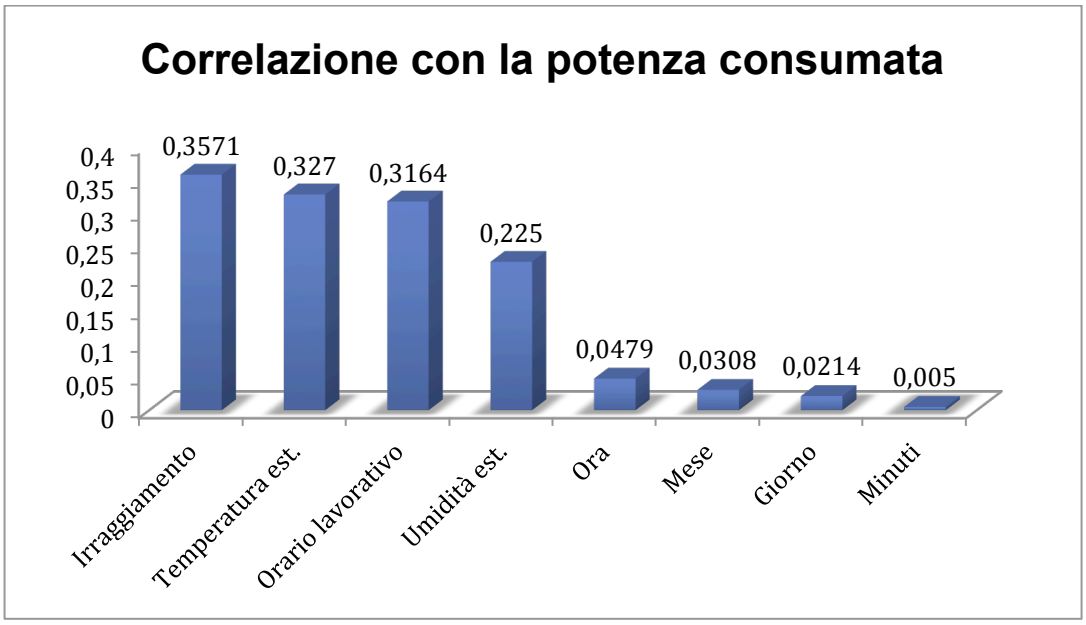

Figura 2

L'attributo "anno" non è stato riportato poiché è costante. Si nota facilmente che gli attributi maggiormente correlati con la potenza consumata sono quelli atmosferici mentre quelli relativi al timestamp lo sono molto poco. Pertanto per minimizzare il numero di attributi per campione in ingresso alla rete, e limitare quindi le dimensioni di quest'ultima, abbiamo deciso di fornire in ingresso i 3 attributi maggiormente correlati con l'attributo target, ovvero: "Irraggiamento", "Temperatura esterna" e "Orario lavorativo".

3.2 Scelta della configurazione ottimale della rete

Stabiliti quali sono gli input e il target possiamo passare alla fase di creazione e addestramento della rete neurale.

Il primo passo consiste nella scelta della rete di tipo time series. Il MATLAB mette a disposizione 3 reti. Per la risoluzione del nostro problema la scelta ricade sulla rete NARX (Nonlinear Autoregressive with External Input), il cui schema è mostrato in Figura 3.

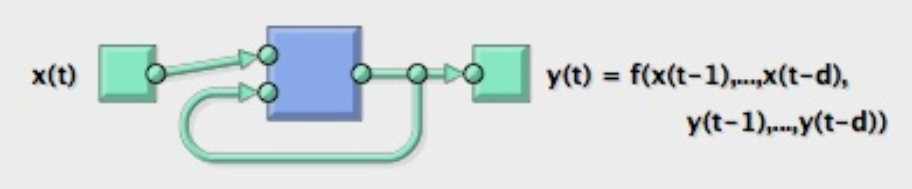

Figura 3

I parametri per tale tipo di rete sono:

- Input
- Target
- Dimensione percentuale del training set

- Dimensione percentuale del validation set
- Dimensione percentuale del testing set
- Numero di neuroni dello strato nascosto
- Numero di ritardi

Gli input e il target li abbiamo già stabiliti nella sezione precedente. Per quanto riguarda le dimensioni dei set di training, validation e test abbiamo lasciato i valori di default: 70% per il training set, 15% per il validation set e 15% per il testing set.

Per ciò che concerne il numero di neuroni dello strato nascosto, la teoria ci fornisce una linea guida per cui ad ogni peso dovrebbero corrispondere dai 5 ai 10 training patterns.

Scegliere il valore di ritardo, invece, significa scegliere quanto vogliamo che gli output passati influenzino l'output attuale. Pertanto, per effettuare una buona scelta, occorre osservare l'andamento dell'output. Ciò è stato fatto calcolando, il numero medio di occorrenze consecutive nelle quali la potenza consumata fosse diversa da zero. Il risultato di tale calcolo, ci dice che, mediamente, si verifica un consumo di potenza per circa 9 campioni consecutivi, ovvero 4 ore e mezza visto che l'intervallo di tempo tra un campione e un altro è di mezzora. Pertanto abbiamo pensato che un buon valore di ritardo fosse circa la metà di 9, cioè 5.

Per dimensionare il numero di neuroni nello strato nascosto, abbiamo deciso di testare in prima persona il comportamento della rete al variare del numero di neuroni. In prima istanza abbiamo preso come parametro di bontà della rete l'MSE (Mean Square Error). Quindi abbiamo creato uno script in Matlab (Allegato 2) che per ciascuna configurazione della rete esegue il training 10 volte visto che ogni esecuzione del training della stessa rete fornisce risultati diversi. In tutte le istanze di training, il valore di ritardo è stato costantemente pari a 5 in base a quanto detto sopra.

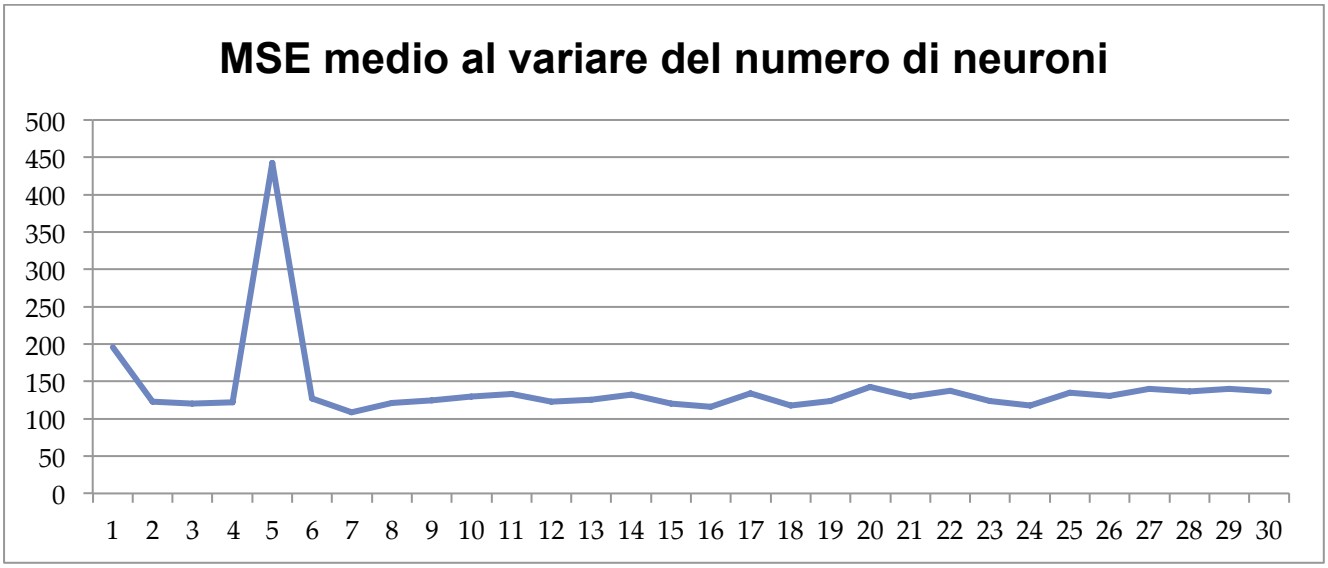

Figura 4

I risultati medi del nostro esperimento sono riportati in Figura 4. Per i risultati completi rimandiamo all'Allegato 3. Notiamo un picco del MSE medio di circa 442 causato da un'istanza

di training particolarmente "sfortunata" che ha fornito un MSE di ben 3331,6. Escludendo la configurazione con un solo neurone, è possibile osservare che al variare del numero di neuroni l'MSE medio oscilla nella fascia di valori compresa tra circa 100 e 150. Tali valori risultano realistici visto che i target che dovranno essere predetti oscillano tra 0 e 280. Pertanto possiamo dire che ci attestiamo con valori di MSE dello stesso ordine di grandezza del target.

Per aiutarci a scegliere il numero di neuroni della rete neurale che consenta di avere un MSE minimo, riportiamo più comodamente gli stessi dati nel grafico in Figura 5.

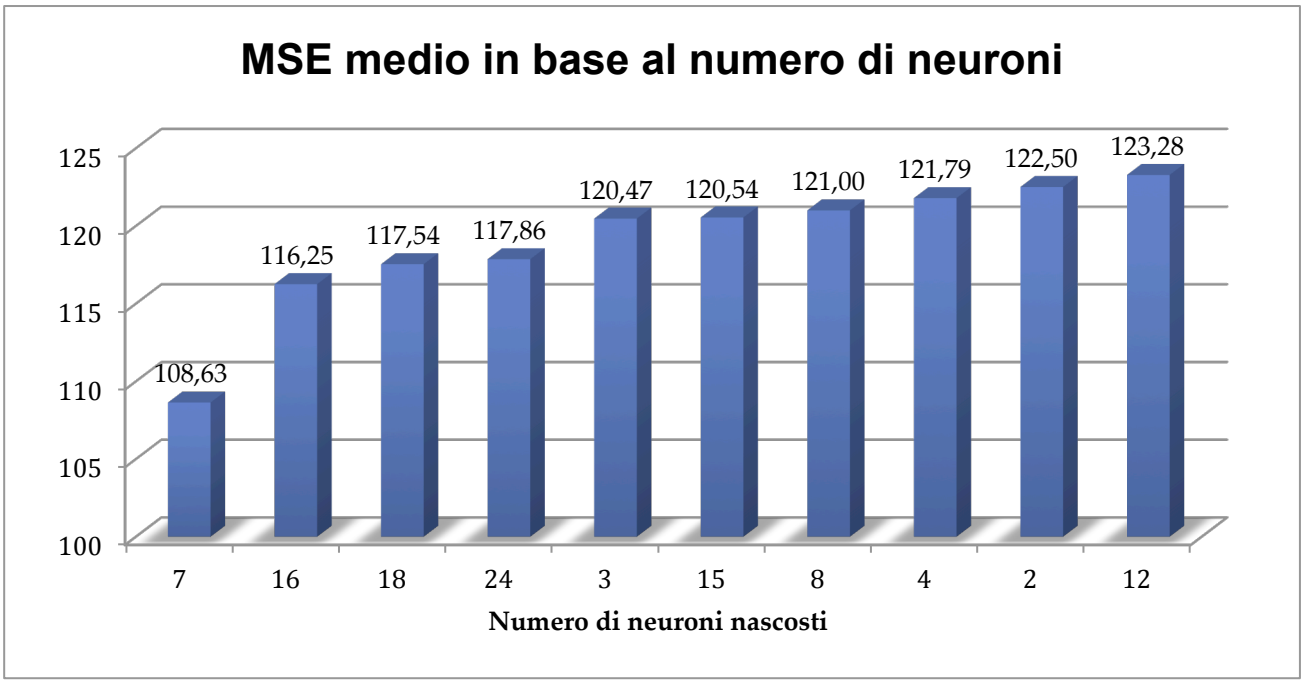

Figura 5

In quest'ultima sono riportati i valori di MSE medi delle migliori 10 configurazioni. Il test da noi condotto ci indica che la rete neurale che fornisce mediamente un MSE più basso è quella avente 7 neuroni nello strato nascosto, a seguire troviamo le configurazioni da 16 e da 18 neuroni.

Come obiettivo qualitativo ci siamo imposti di ottenere una rete che avesse un MSE minore di 100 al termine del training. Visti i risultati medi sappiamo bene che non sarà facile scendere al di sotto di tale soglia. Per scegliere la corretta configurazione allora è utile verificare anche il numero di volte che una determinata configurazione è riuscita ad avere un MSE inferiore al 100.

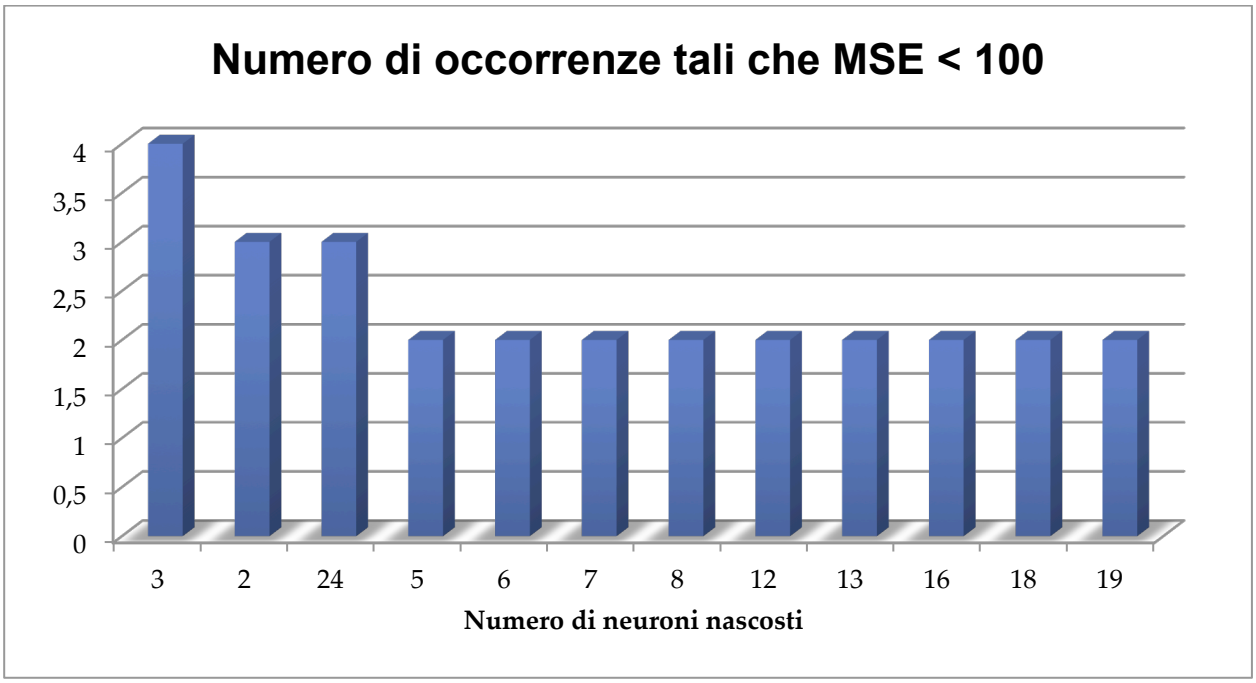

Figura 6

Nel grafico in Figura 6 è riportato il numero di volte che una determinata configurazione è riuscita a ottenere un MSE inferiore a 100 durante il test effettuato tramite lo script Matlab in Allegato 2. Per comodità abbiamo riportato solo le configurazioni che sono riuscite a soddisfare la condizione sopra citata in almeno 2 casi su 10. È evidente che dobbiamo escludere le configurazioni con 3 e 2 neuroni nascosti poiché sono valori troppo bassi per il numero di sample a disposizione. Incrociando il grafico in Figura 5 con quello di Figura 6 notiamo che delle buone configurazioni potrebbero essere quelle con rispettivamente 7, 16, 18 e 24 neuroni.

L'MSE è un parametro che ci indica la bontà di una determinata rete ma non è il solo. Vanno infatti considerati altri parametri quali ad esempio la correlazione tra input ed errore e la distribuzione dei valori di errore. A seguito di ulteriori test condotti sulle configurazioni individuate in precedenza, dove sono stati considerati tutti gli indicatori di bontà di una rete, abbiamo deciso di concentrare i nostri sforzi sulla configurazione a 16 neuroni e 5 ritardi (Figura 7).

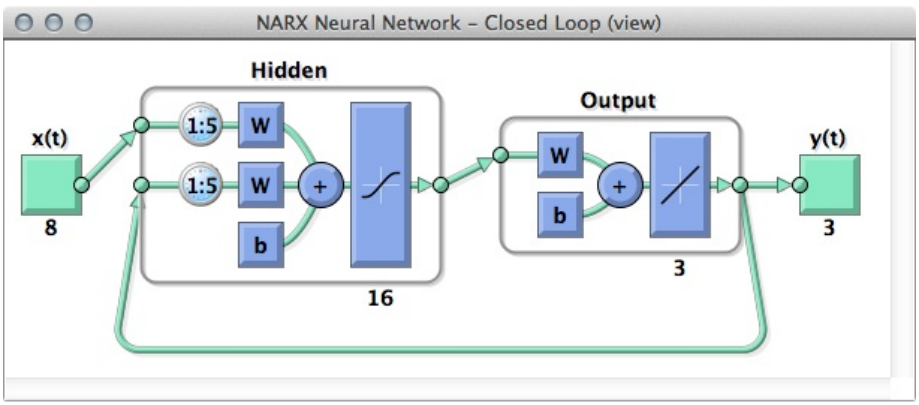

Figura 7

3.3 Risultati

Al fine di ottenere una rete che soddisfi i nostri requisiti di qualità, abbiamo creato uno script in Matlab (Allegato 4) che esegue all'infinito il training della rete neurale scelta. Nel caso in cui, in una determinata istanza di training, venisse ottenuto un MSE inferiore al 100, lo script salva tutti i grafici relativi al training e il workspace del Matlab su disco. In questo modo potremmo scegliere tra tutte le istanze di training con MSE minore di 100 quelle aventi i grafici migliori.

Di seguito riportiamo i migliori risultati ottenuti.

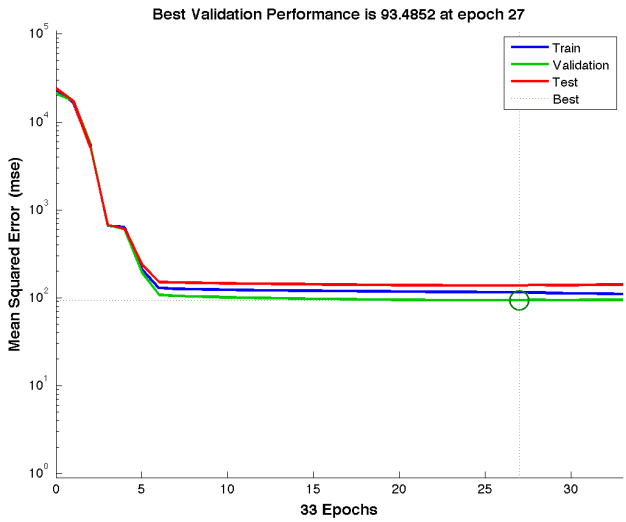

Figura 8

Iniziamo con il grafico delle performance in Figura 8, che mostra l'andamento degli errori di addestramento, validazione e testing. Come è possibile notare, siamo riusciti a ottenere un MSE pari a circa 93 all'epoca 27. L'MSE è di circa il 7% inferiore al soglia che ci siamo imposti, quindi possiamo ritenerci soddisfatti. Dall'andamento degli errori di testing e validation possiamo concludere che non sembra esserci overfitting quindi la rete è in grado di generalizzare correttamente.

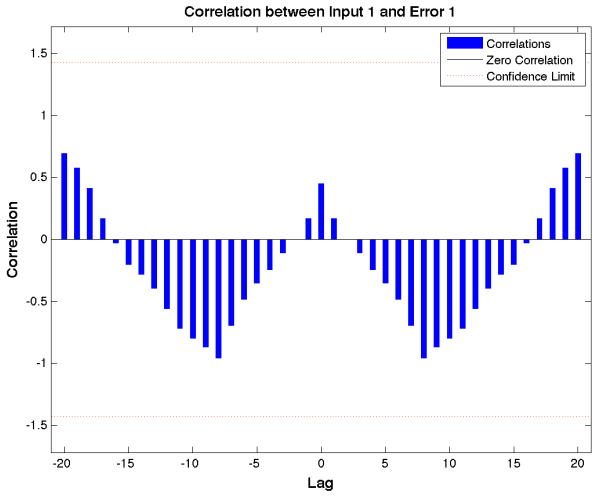

Figura 9

In Figura 9 è riportato il grafico di correlazione tra input ed errore. Esso mostra come gli errori sono correlati con la sequenza di input x(t). Per un modello di predizione perfetto, tutte le correlazioni dovrebbero essere a 0. In questo caso il modello si comporta molto bene poiché tutte le correlazioni rientrano abbondantemente nell'intervallo di confidenza.

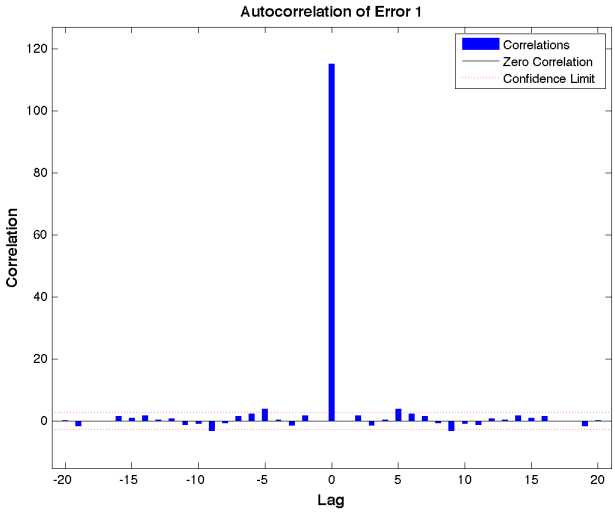

Figura 10

La Figura 10 mostra la funzione di autocorrelazione dell'errore: descrive come gli errori di predizione sono correlati nel tempo con se stessi. Per un perfetto modello di predizione ci dovrebbe essere un solo valore non nullo in corrispondenza dello 0 (questo è l'errore quadratico medio). Ciò significherebbe che gli errori di predizione sono completamente scorrelati da ciascun altro (rumore bianco). Il risultato è abbastanza buono poiché le correlazioni cadono per il 90% nell'intervallo di confidenza. Perciò possiamo dire che il modello è adeguato.

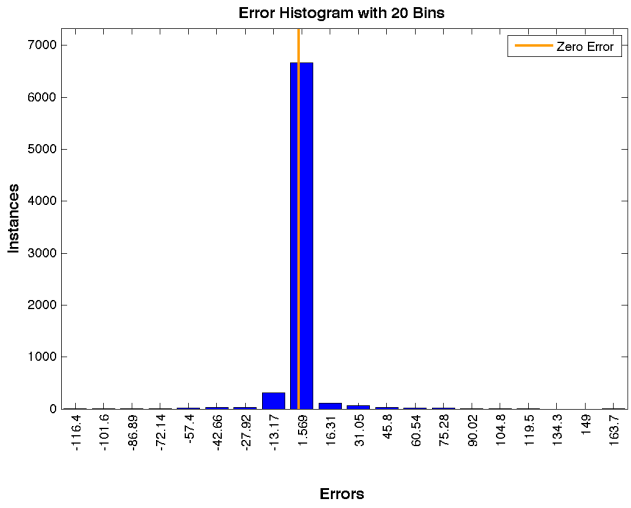

Figura 11

In Figura 11 è riportato l'istogramma dell'errore che mostra il numero di volte che un certo valore dell'errore si presenta. Purtroppo non è stato possibile mostrare la suddivisione dei vari set poiché lo script generato in maniera automatica non fornisce questa possibilità. Notiamo che il numero di occorrenze per valori di errore che si allontanano dallo zero è decisamente basso.

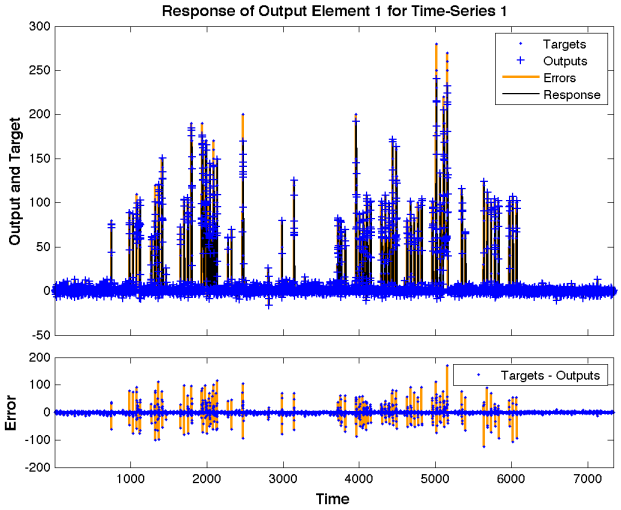

Figura 12

La Figura 12 mostra gli inputs, i target e gli errori nel tempo. Indica inoltre quali punti nel tempo sono stati selezionati per il training, il testing e la validazione.

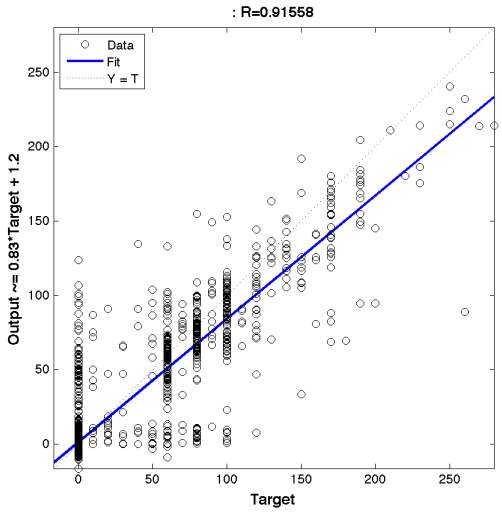

Figura 13

In Figura 13 è raffigurata la retta di regressione tra il target e l'output. Dal grafico emerge che il modello è molto buono. La retta di regressione ha un coefficiente sufficientemente vicino a 1 quindi la retta ideale si sovrappone alla retta ottenuta ad ogni prova. Tale retta minimizza la somma dei quadrati degli scarti. Non essendoci pallini estremamente lontani dalla retta vuol dire che per tutte le epoche si sono registrati errori piuttosto bassi. Anche in questo caso lo script non ci ha fornito una suddivisione per set.

Concludiamo dicendo che la nostra rete ha 353 pesi. La teoria invece afferma che tipicamente, visto il numero di campioni, il numero di pesi dovrebbe essere compreso tra 514 e 1028. Pertanto siamo riusciti ad ottenere dei buoni risultati con una rete decisamente più piccola di quella che in teoria dovremmo avere.

4 Svolgimento con Fuzzy Logic Tool

Questa parte di progetto chiedeva di realizzare un sistema Fuzzy con la metodologia Mamdani. Questo sistema offre un'elevata facilità di interpretazione del problema, basandosi sulle regole logiche di inferenza, ma si ha poca accuratezza unita ad un elevato costo computazionale.

Matlab fornisce il tool grafico Fuzzy Logic Toolbox che ci ha permesso di costruire un sistema Mamdani e testare graficamente le uscite.

Prima di passare allo studio del sistema che abbiamo sviluppato, è necessario capire la logica che abbiamo usato, ossia come i dati che abbiamo a disposizione possano essere correlati tra loro secondo alcune regole e se fosse possibile classificarli.

4.1 Analisi dei dati

Innanzi tutto abbiamo verificato quante occorrenze dei valori della potenza fossero presenti nei dati fornitici. Il risultato di tale verifica è mostrato in Figura 14.

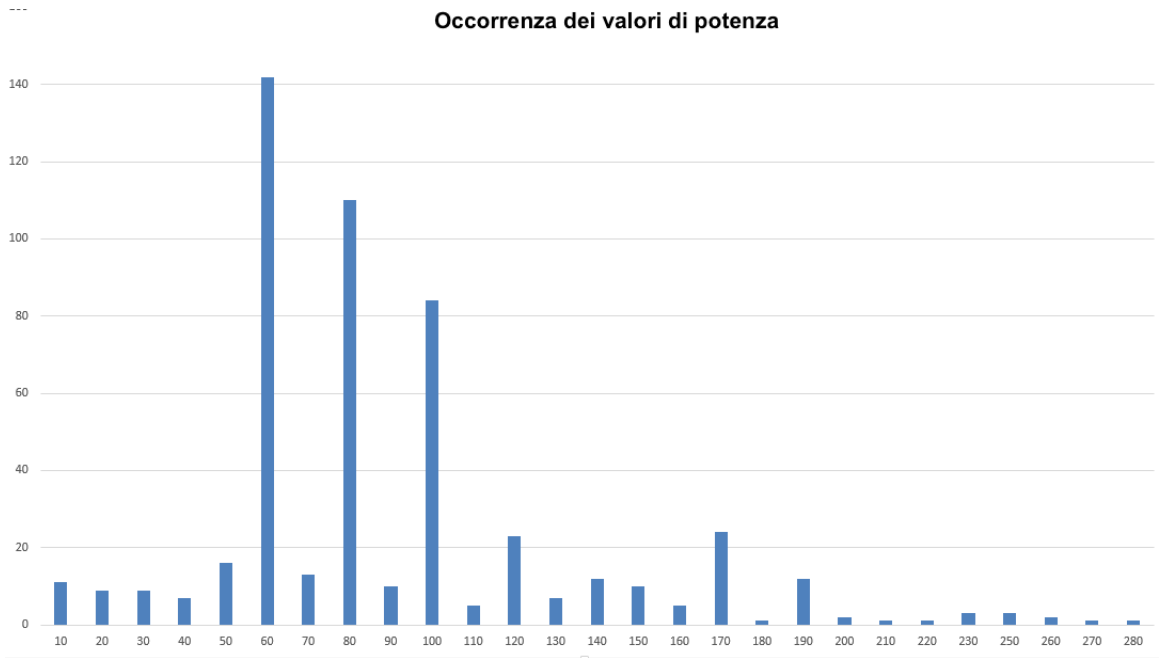

Figura 14

Come possiamo notare, occorrenze più alte si hanno per valori di consumo di 60, 80, 100, 120 e 170 Watt. Notiamo pochissime occorrenze per valori sopra i 200 Watt, che sembrerebbero più come una volontà sporadica da parte dei lavoratori dell'ufficio.

Purtroppo analizzando questi valori è stato molto difficile catalogare, o trovare qualcosa in comune, tra i vari dati perché a stessi valori di consumo corrispondono valori di temperatura, umidità e irraggiamento diversi.

Per un'analisi più approfondita abbiamo perciò visto come i tre input atmosferici (umidità, irraggiamento, temperatura) potessero essere correlati con l'uscita, cercando di capire come la variazione di uno influisse sulla variazione dell'altro. Per fare ciò abbiamo sovrapposto i valori di ciascun input con l'output su dei grafici per valutarne qualitativamente l'andamento.

Nei grafici successivi, abbiamo riportato una porzione adeguatamente rappresentativa dei training pattern. La linea arancione rappresenta il consumo, mentre la linea blu rappresenta l'input.

Temperatura

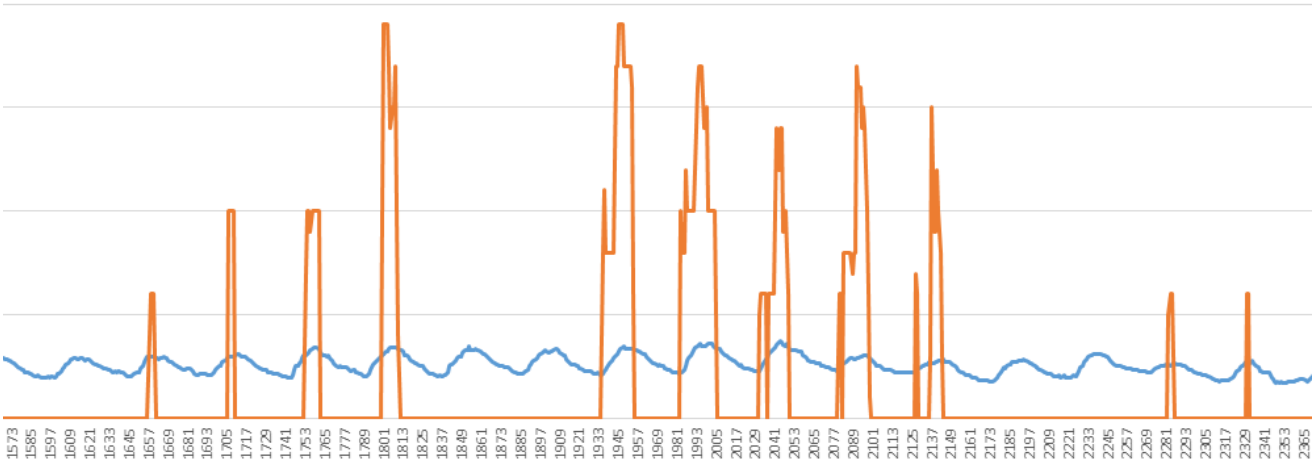

Figura 15

In Figura 15 abbiamo alcuni valori di temperatura (linea blu) e notiamo che i picchi di consumo sono in corrispondenza dei picchi più alti di temperatura. Possiamo anche ipotizzare che sotto una certa soglia di temperatura il consumo sarà sempre zero, per esempio durante la notte o nella stagione autunnale/invernale il condizionatore sarà spento.

Irraggiamento

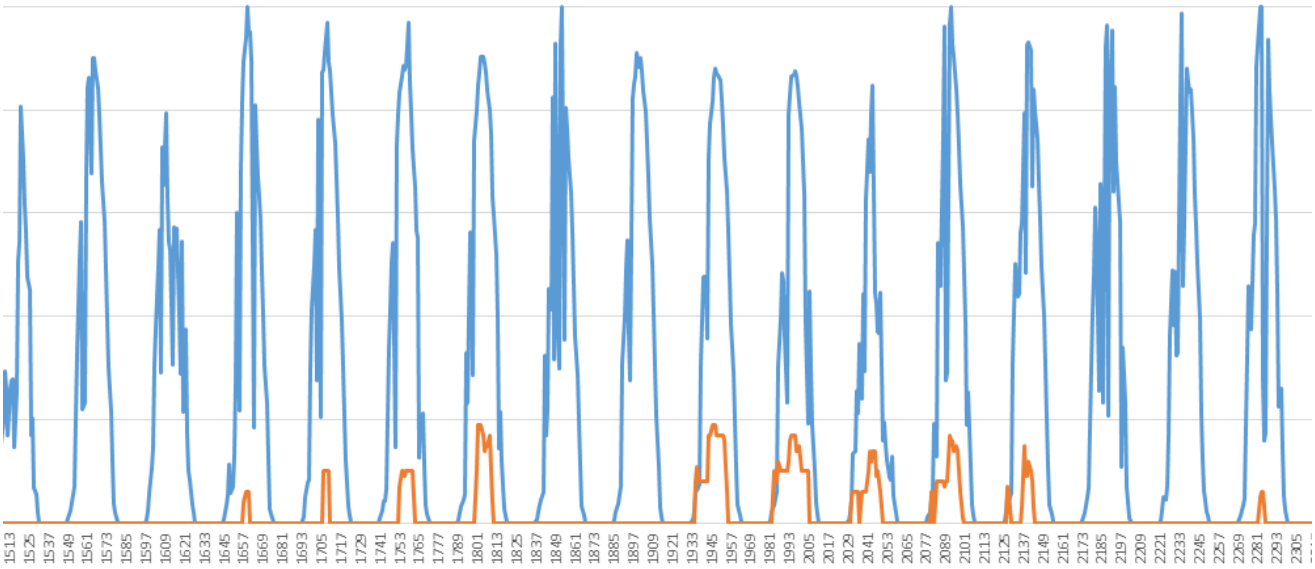

Figura 16

La Figura 16 mostra invece l'irraggiamento (linea blu). Il discorso è analogo alla temperatura: più è alta più avremo alto il consumo. Il consumo è nullo per valori di irraggiamento estremamente bassi, ossia durante la notte, alba o tardo pomeriggio.

Umidità

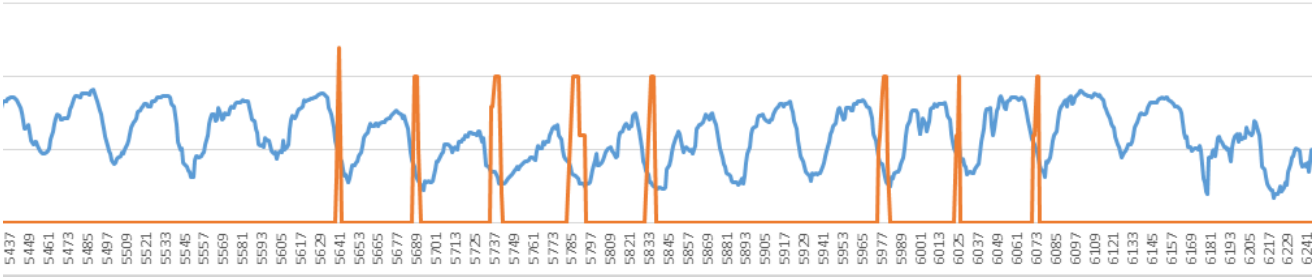

Figura 17

Umidità

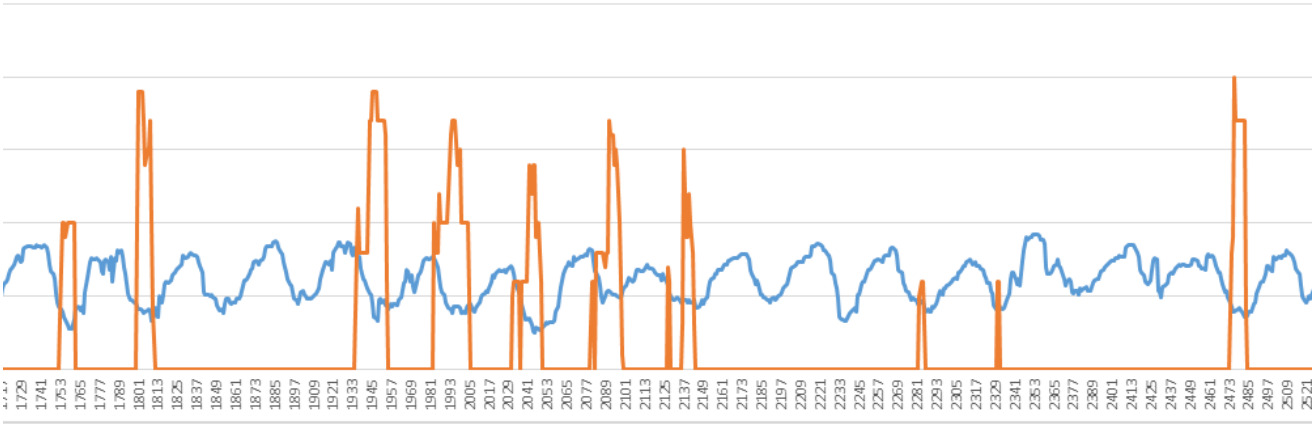

Figura 18

In Figura 17 e Figura 18 abbiamo l'andamento di alcuni valori di umidità (linea blu). Qua il profilo è esattamente l'opposto, il consumo è massimo quando l'umidità è bassa, e possiamo anche ipotizzare, come per gli altri due dati, dei consumi nulli sopra e sotto una certa soglia.

Inoltre mettendo in ingresso anche la nostra variabile Orario Lavorativo, come spiegato all'inizio, sappiamo quando il consumo è nullo in base all'orario, aumentando la precisione e la correttezza della previsione.

Unendo queste informazioni possiamo già scrivere le prime regole logiche:

- Se l'orario è festivo il consumo è zero.
- Se l'orario è lavorativo e la temperatura aumenta (diminuisce) il consumo aumenta (diminuisce).
- Se l'orario è lavorativo e l'irraggiamento aumenta (diminuisce) il consumo aumenta (diminuisce).

- Se l'orario è lavorativo e l'umidità diminuisce (aumenta) il consumo aumenta (diminuisce).

4.2 Sistema Mamdani

Passiamo adesso ad analizzare il sistema Mamdani creato.

La Figura 19 mostra la struttura del sistema con gli ingressi e uscite. Come spiegato precedentemente gli ingressi sono la temperatura, l'irraggiamento, l'umidità e l'orario lavorativo; mentre in uscita abbiamo il consumo di potenza.

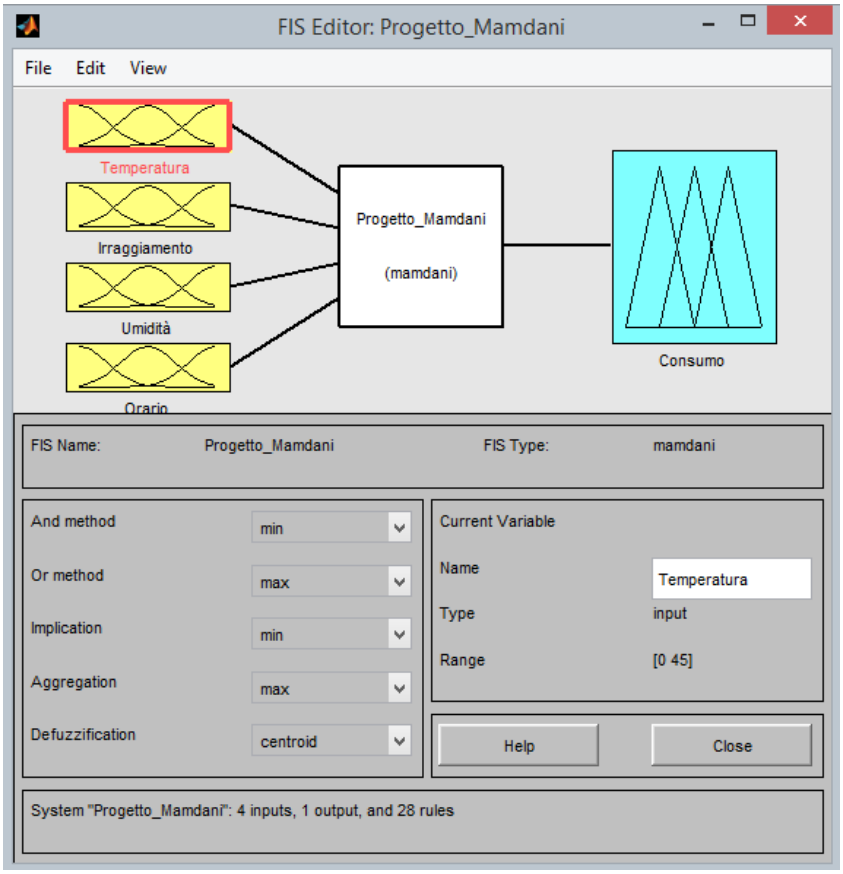

Figura 19

Qui di seguito mostriamo le MF per ciascuna delle variabili sopra citate.

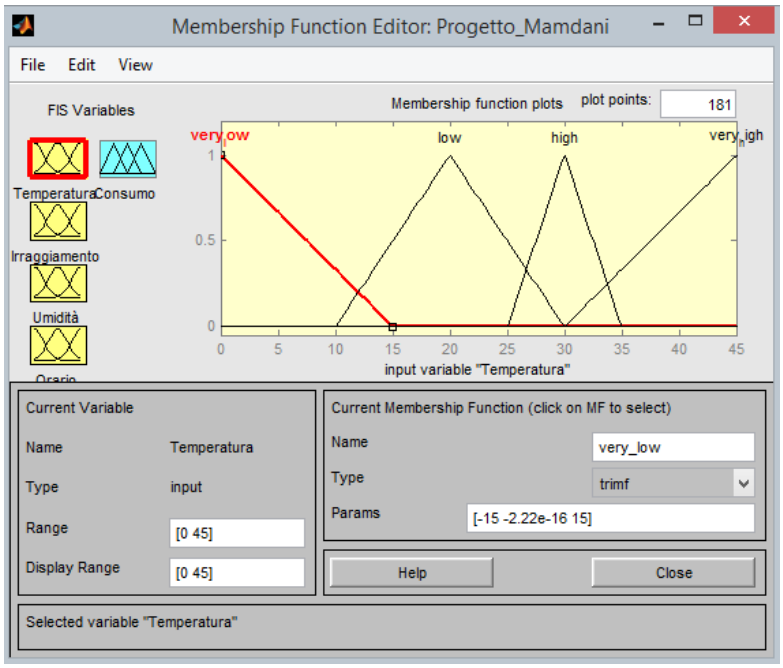

Figura 20

Per quanto riguarda la temperatura (Figura 20), abbiamo ipotizzato un range di valori che va da 0 a 45 gradi. Le classi sono molto più ampie per i valori di temperatura bassi in modo da coprire i casi in cui il consumo è nullo.

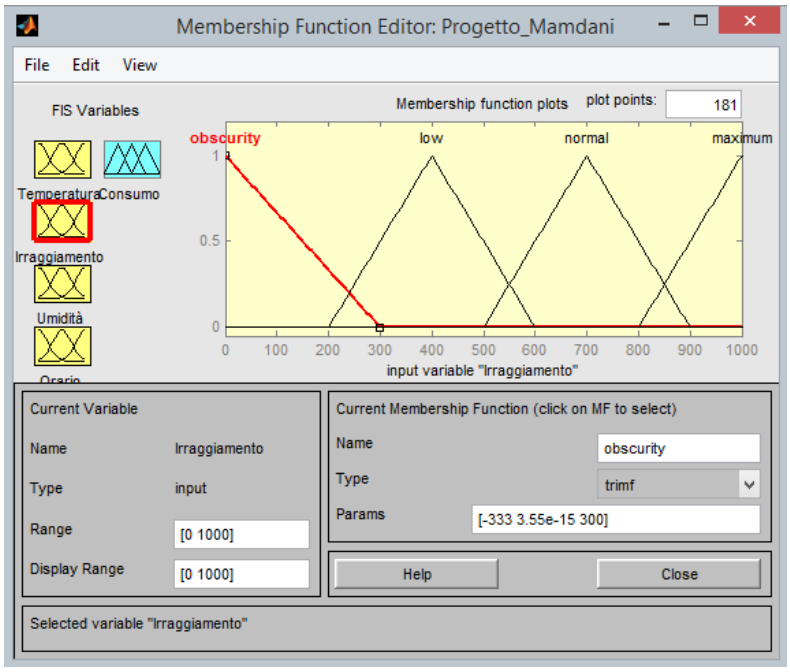

Figura 21

L'irraggiamento (Figura 21) è più bilanciato, assume valori da 0 a 1000 e sale molto velocemente durante il giorno. Tutte le MF sono quindi molto simili come dimensione.

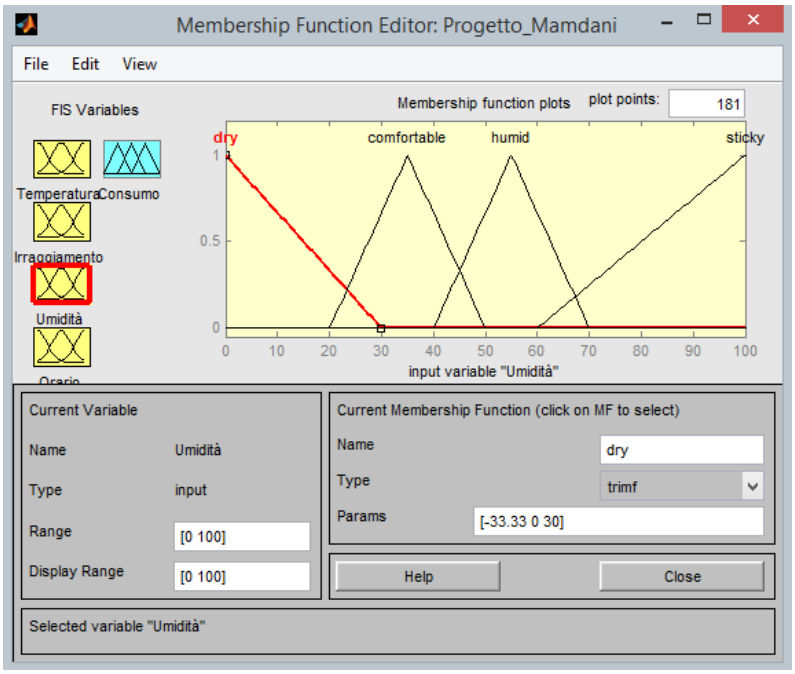

Figura 22

Per l'umidità (Figura 22) invece non abbiamo particolari casi in cui registriamo valori molto alti. In maniera analoga alla temperatura, abbiamo una MF che copre questi casi e dove il consumo sarà nullo, secondo le ipotesi che abbiamo fatto precedentemente.

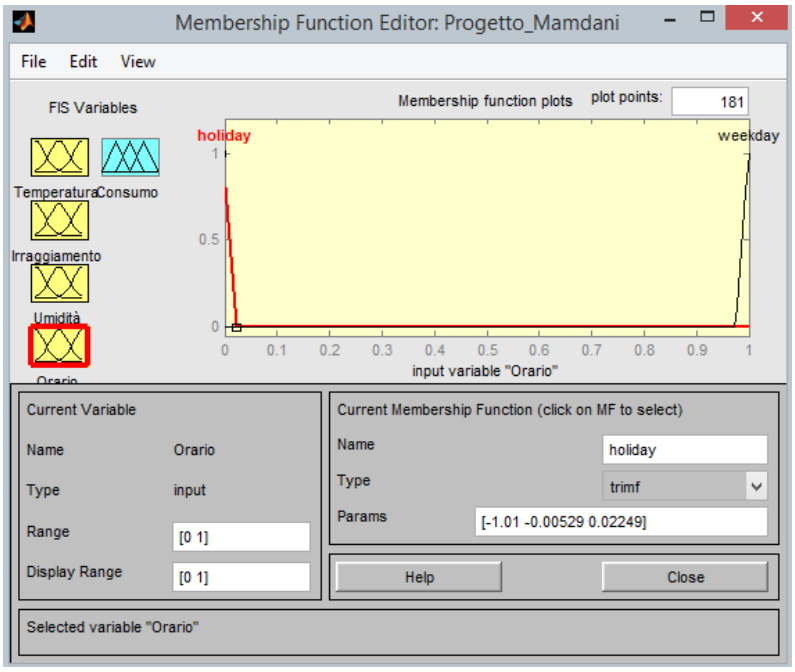

Figura 23

Per come abbiamo impostato l'orario (Figura 23) può avere solo due valori: 1 (lavorativo) e 0 (festivo).

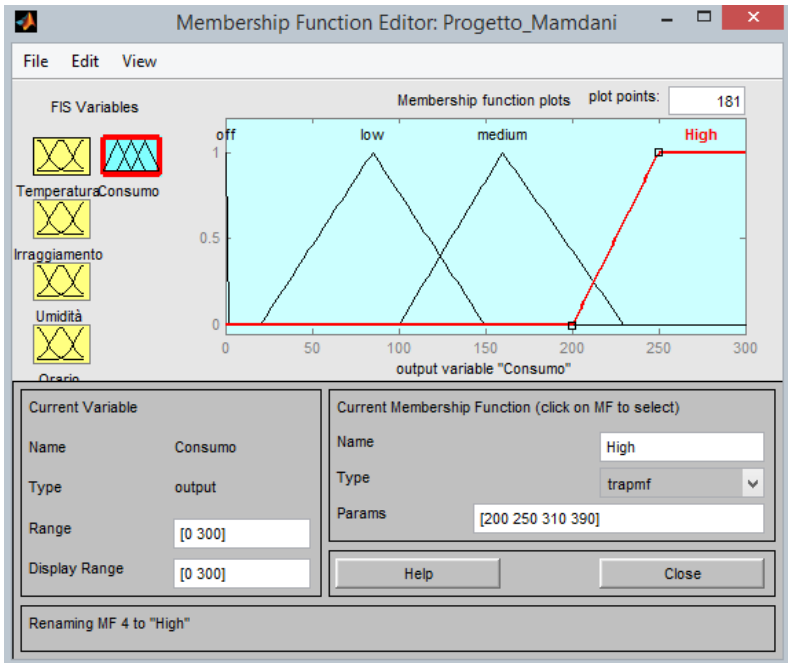

Figura 24

Per il consumo (Figura 24) abbiamo ipotizzato che abbia valori compresi tra 0 a 300 Watt. Dai dati in nostro possesso il consumo massimo ha toccato 280 Watt ma non possiamo sapere se effettivamente sia il massimo consentito dal condizionatore. Questa variabile è suddivisa da 4 MF, la prima indica se il condizionatore è spento, mettendo l'uscita a zero. Le altre 3 cercano di coprire tutte le casistiche, facendo particolare attenzione ai valori dove c'è più occorrenza, ossia dagli 80 Watt ai 170 Watt.

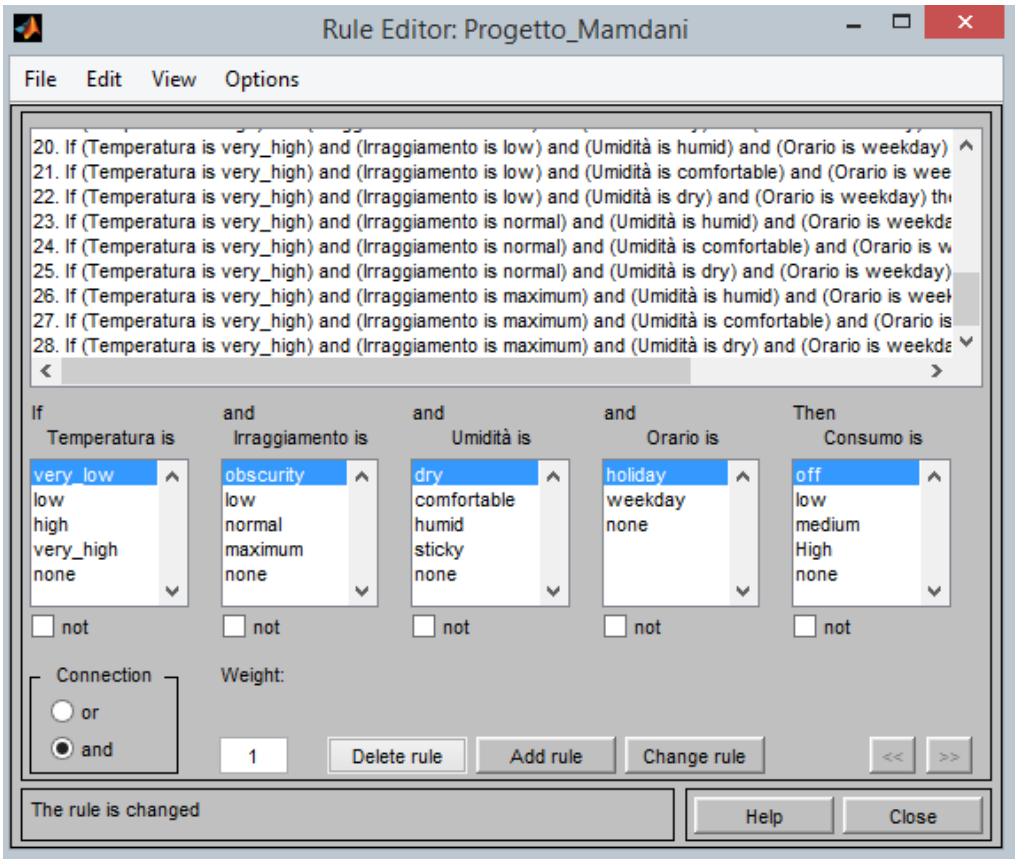

Figura 25

In Figura 25 mostriamo le 28 regole che abbiamo creato e che coprono tutte le possibili combinazioni degli ingressi. Le tre variabili atmosferiche sono divise da 4 MF ognuna. Coprendo però con una sola regola il caso in cui il condizionatore è spento, ossia quando l'orario è festivo oppure la temperatura è "very_low" oppure quando l'irraggiamento è "obscurity" oppure quando l'umidità è "sticky". Di fatto le combinazioni dei tre ingressi sono sulle 3 MF rimanenti. L'orario lavorativo non aggiunge complessità, in quanto si tratta di una sola variabile binaria.

Ecco quindi che abbiamo 27 regole generate dalla combinazione delle 3 Mf dei 3 ingressi atmosferici, ed una regola aggiuntiva per coprire il caso in cui il condizionatore è sicuramente spento.

L'elenco dettagliato di tutte le regole è riportato nell'Allegato 5.

4.3 Risultati

Il tool mostra graficamente l'uscita impostando manualmente l'ingresso. In Figura 26 è mostrato come impostando "Orario" a 0 il consumo sia nullo. I valori degli altri input vengono ignorati. Lo stesso risultato avviene anche quando la temperatura è bassa, l'irraggiamento è nullo o molto basso, e l'umidità è alta.

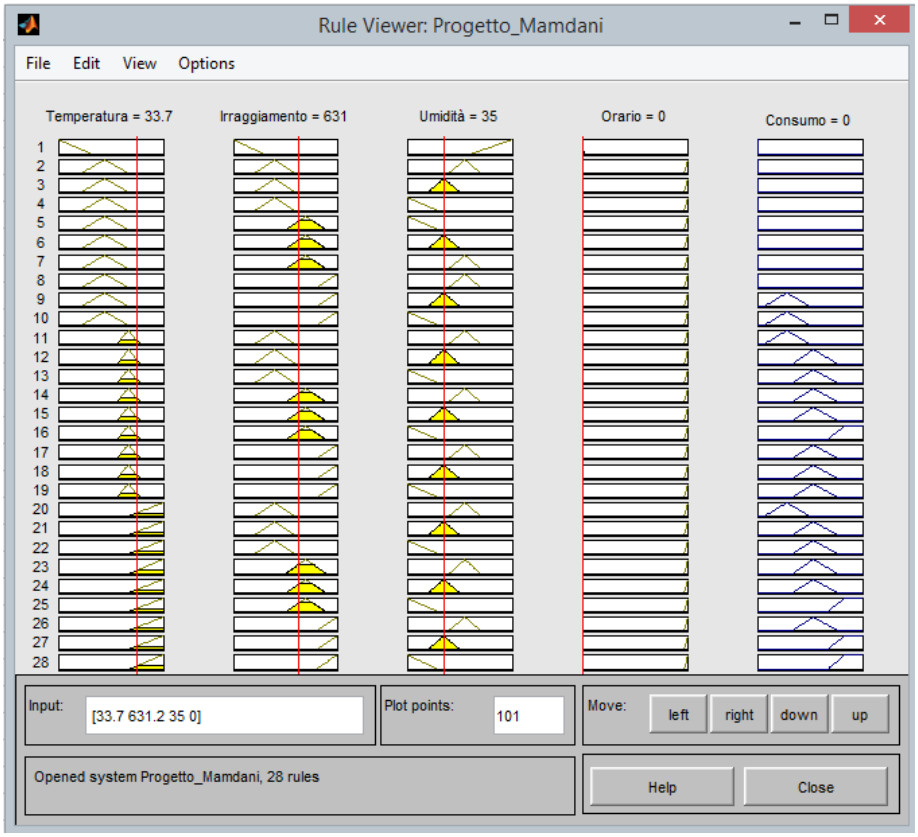

Figura 26

Abbiamo scelto casualmente qualche campione per verificare il comportamento del sistema. I risultati e i riferimenti alle relative figure sono riportati in Tabella 1.

N. Campione	Output	Target	Errore	Figura
1802	185	190	5	Figura 27
1416	84	80	-4	Figura 28
1418	162	150	-12	Figura 29

Tabella 1

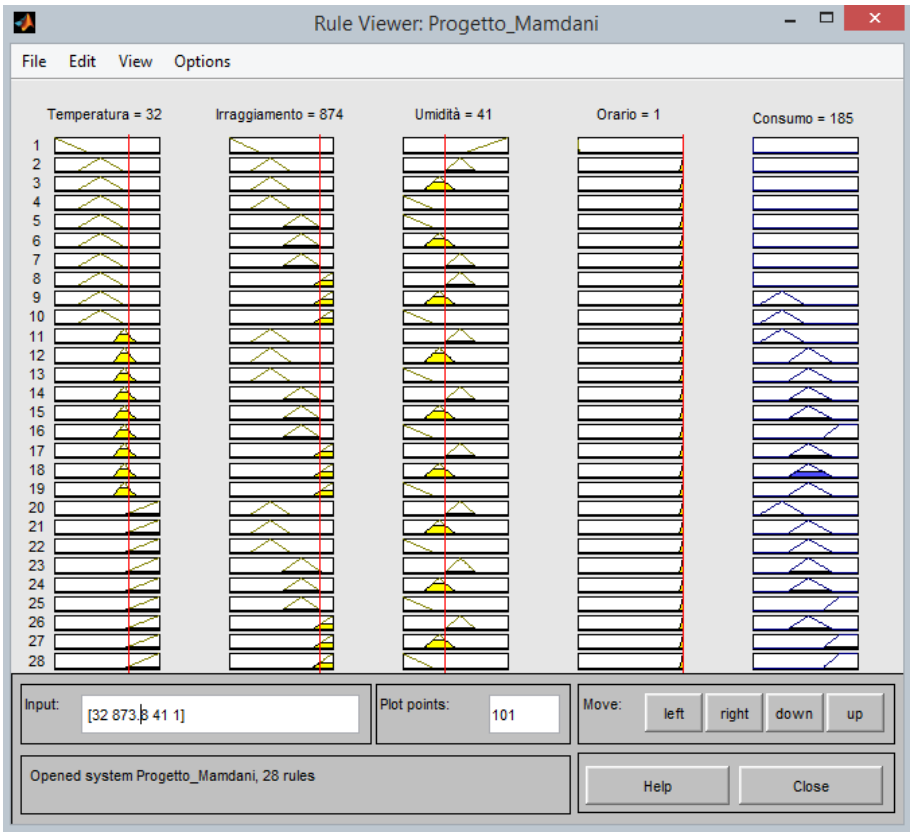

Figura 27

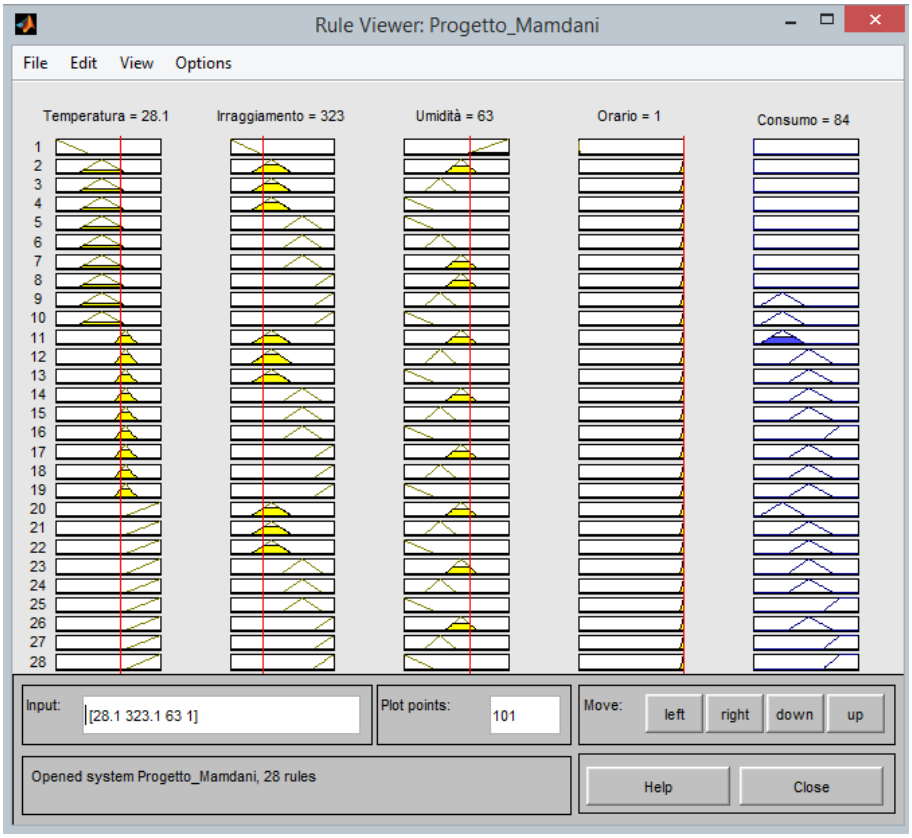

Figura 28

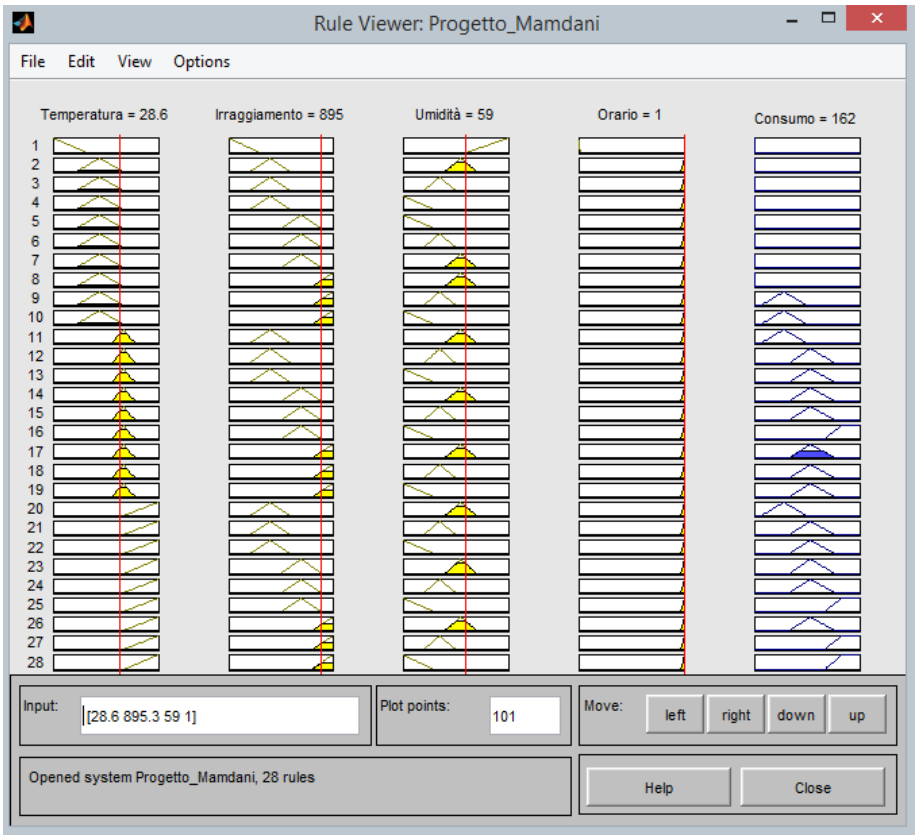

Figura 29

Secondo le nostre verifiche il nostro sistema Mamdani registra correttamente i valori di uscita di consumo, con un errore intorno al ±5 per certi casi. Non tutti i casi vengono predetti correttamente in quanto non c'è una costanza del consumo rispetto a certi valori delle variabili atmosferiche, perché dipende soprattutto dall'uso quotidiano che i dipendenti dell'ufficio fanno del climatizzatore.

5 Svolgimento con ANFIS Editor

Il tool ANFIS Editor permette di creare una rete neurale basata su un systema fuzzy del tipo Sugeno.

Il Sugeno ha come uscita MF che sono costanti o lineari.

Avendo a disposizione il nostro sistema Mamdani, è stato possibile trasformarlo in Sugeno con la funzione di Matlab *mam2sug()*.

Dato che ANFIS Editor non supporta sistemi Sugeno che hanno MF condivise con più regole, abbiamo dovuto creare 28 MF di uscita, una per ogni regola, e spesso rappresentano lo stesso valore di uscita. Abbiamo scelto tutte MF costanti il cui valore è stato determinato direttamente dalla funzione di conversione di Matlab.

Nella Figura 30 vediamo un'immagine di quanto abbiamo appena detto.

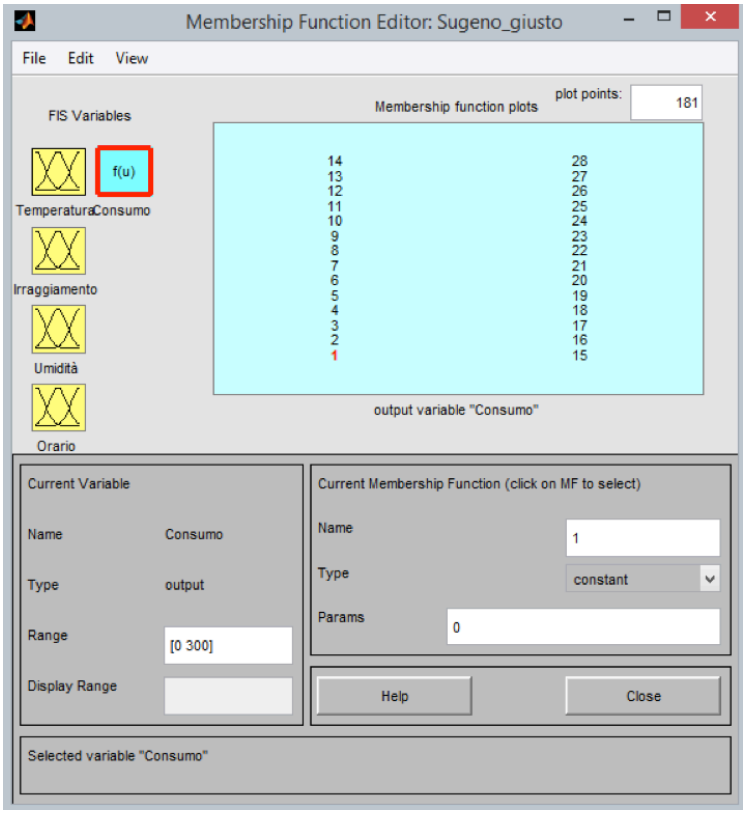

Figura 30

Per semplicità ogni MF è stata chiamata con il numero della sua regola di riferimento.

Per effettuare il training abbiamo utilizzato uno script Matlab(Allegato 6) che suddivide casualmente i dati del Fancoil in tre set: un set per il Training, uno per il test ed uno per il check; in percentuali 70%, 15%, 15% rispettivamente.

5.1 Risultati con il nostro FIS

La Figura 31 mostra il FIS Sugeno importato, come gli input si intrecciano con le regole, e ad ogni regola corrisponde un output predefinito.

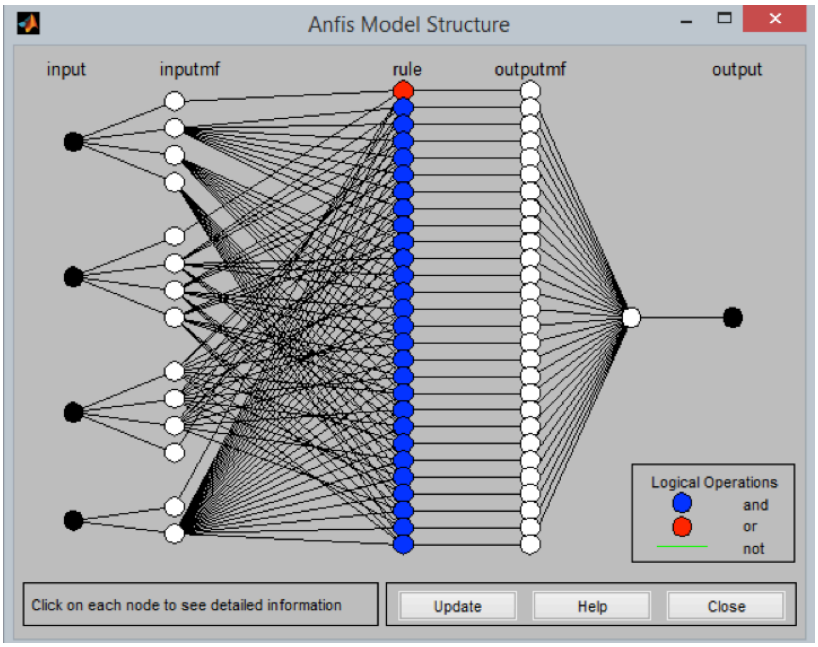

Figura 31

Iniziando a fare il training e selezionando 300 epoche abbiamo ottenuto un errore costante sia sul training set che sul check set, come mostrato nella Figura 32. Aumentando o diminuendo le epoche non c'è stato nessun cambiamento.

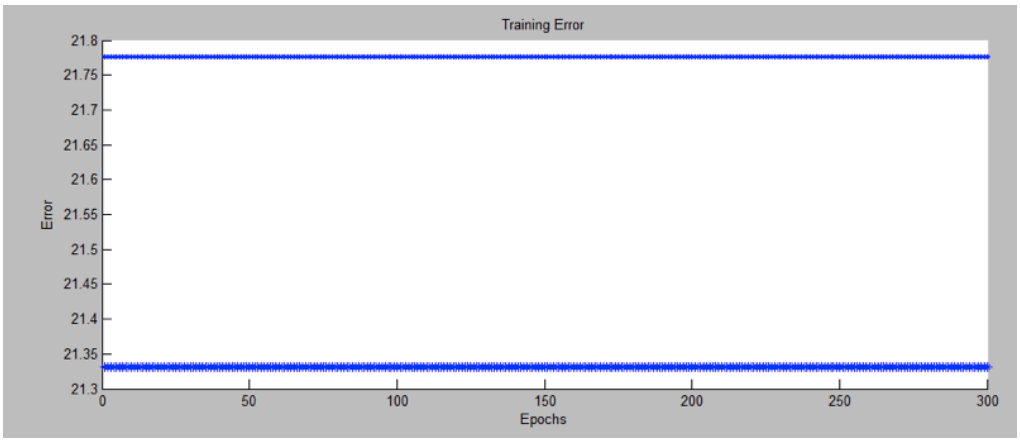

Figura 32

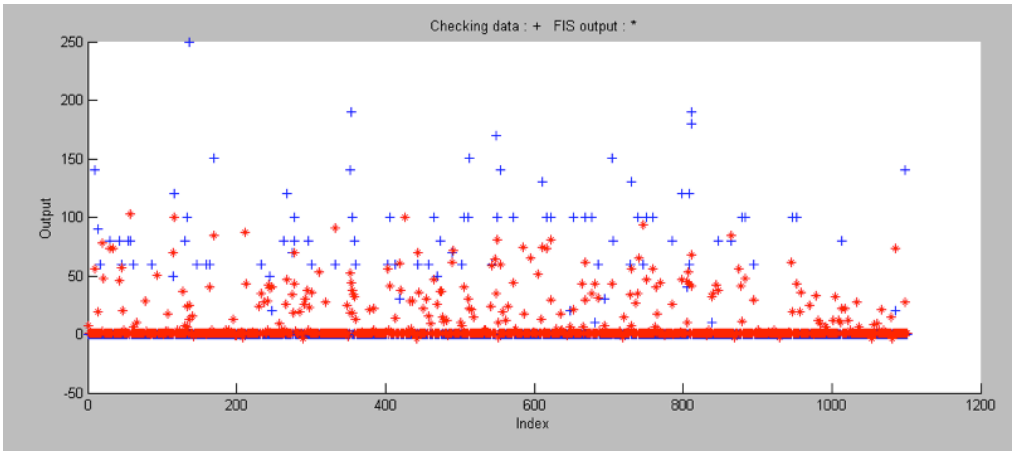

Figura 33

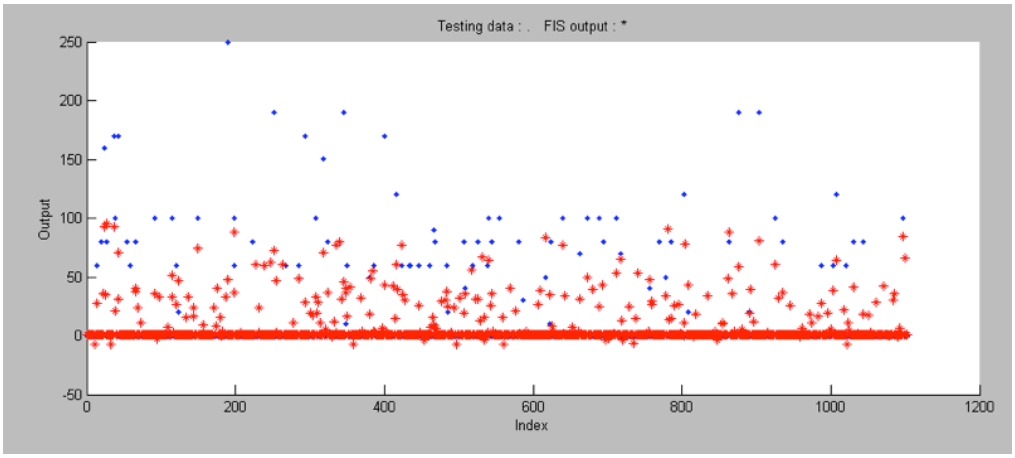

Figura 34

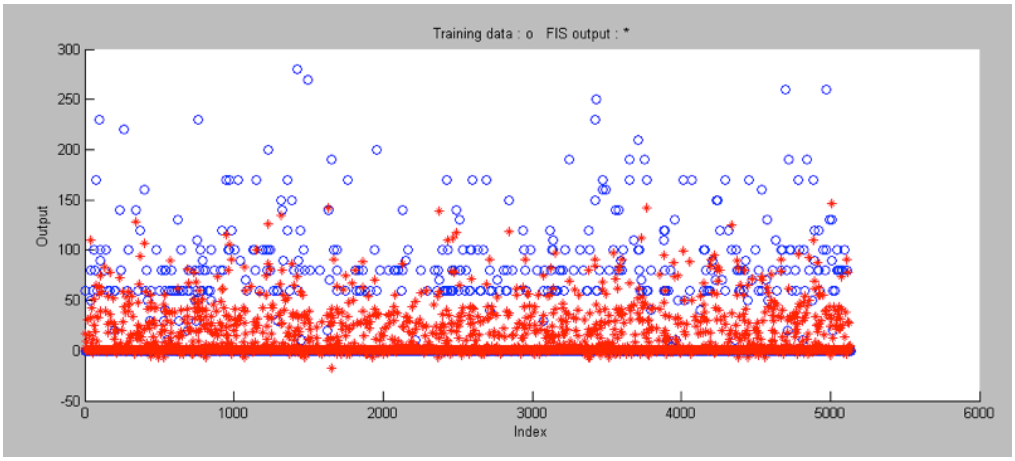

Figura 35

I tre grafici precedenti mostrano il risultato sul test del FIS usando i dati di Check (Figura 33), Test (Figura 34) e Training (Figura 35). Notiamo che sono presenti alcuni consumi che sono negativi, nella realtà è impossibile che ciò avvenga (significherebbe che l'impianto di climatizzazione produce energia invece di consumarla). La rete probabilmente non è riuscita ad imparare correttamente i casi in cui si ha consumo pari a zero. Questi casi però sono sporadici,

li notiamo soprattutto nei dati di Test, sul Check invece non esiste quasi nessun valore minore di zero e sul Training notiamo la stessa cosa. Generalmente il FIS creato riesce grosso modo a prevedere il consumo quando è nullo, invece non c'è molta precisione nel prevedere tutti gli altri casi.

Questo lo possiamo constatare guardando gli errori nei tre set, tutti molto simili:

- Training: 21.3308
- Test: 21.2024
- Check: 21.7765

5.2 Soluzione alternativa

Abbiamo provato ad usare un FIS generato automaticamente tramite Grid Partition, per verificare che quello creato da noi fosse effettivamente il migliore, mantenendo sempre 4 MF per ogni input termico e 2 per l'input dell'orario.

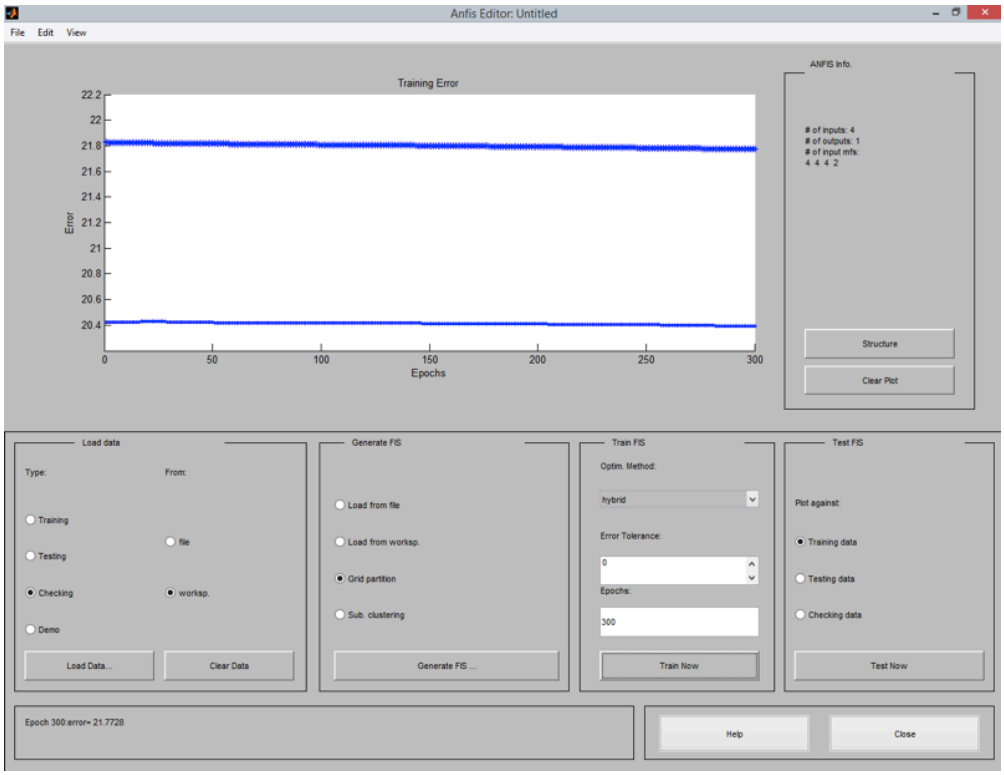

Figura 36

L'errore ottenuto è molto simile a quello generato dal nostro sistema, come in Figura 36. Abbiamo lasciato 300 epoche come compromesso tra tempo impiegato nella computazione e durata del training.

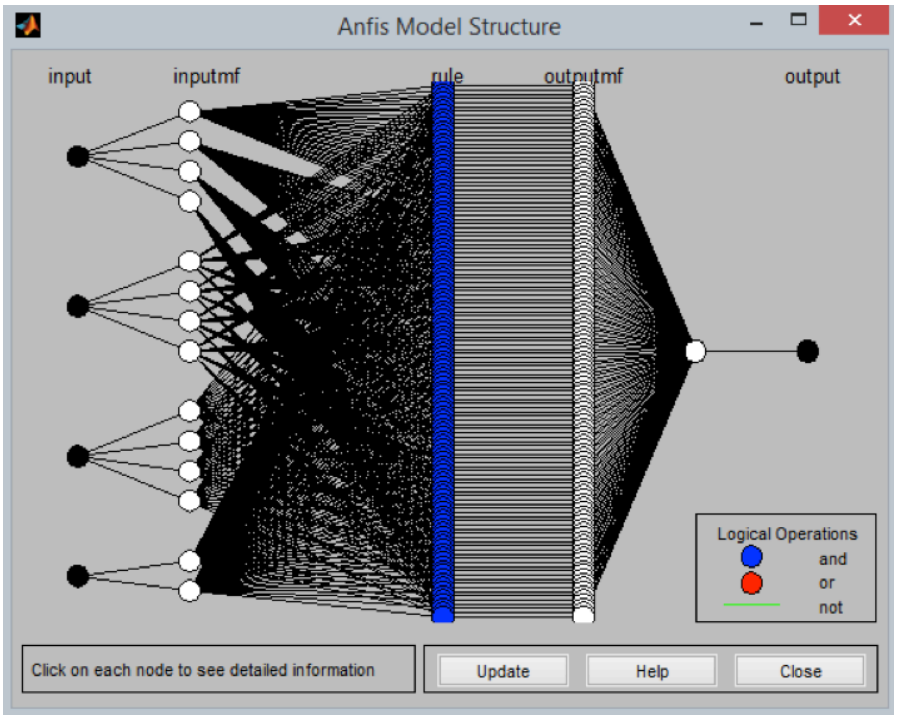

Figura 37

Nel Grid Partition tutti gli ingressi vengono combinati tra loro. Aumentando le MF e gli input, aumenta la complessità di computazione.

5.3 Risultati con il sistema alternativo

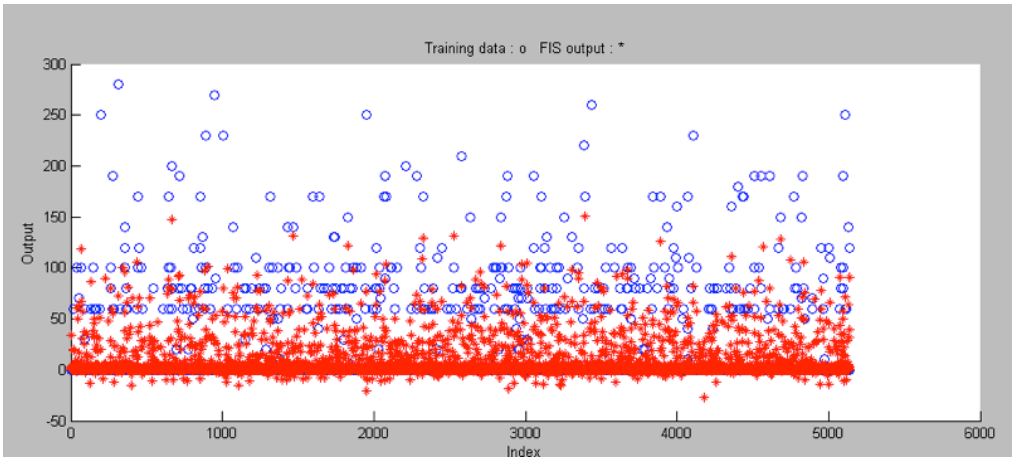

Figura 38

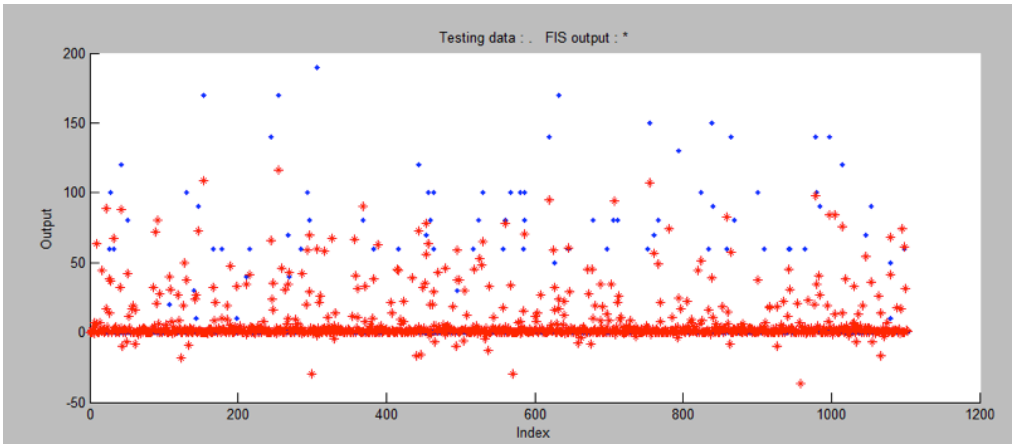

Figura 39

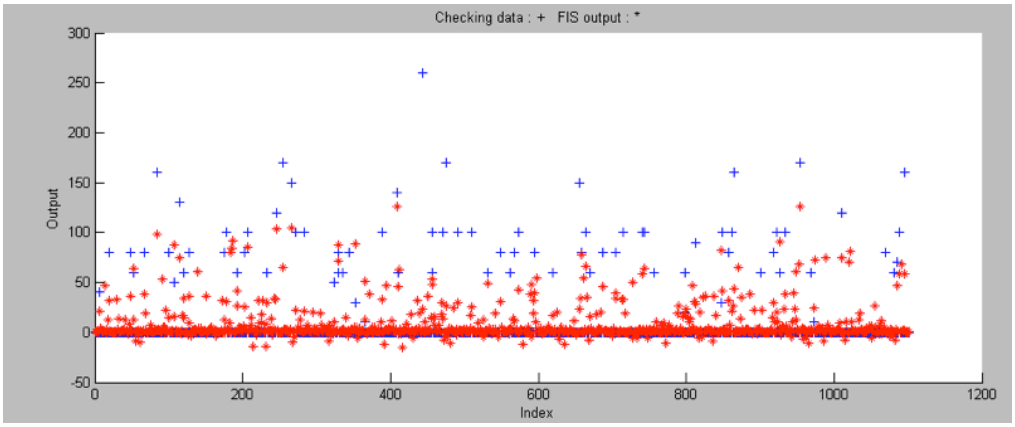

Figura 40

Gli errori per i tre set ottenuti sono:

- Training: 21.7726
- Test: 19.0119
- Check: 20.3911

Molto simili agli errori che abbiamo ottenuto con il nostro FIS.

Osservando i grafici di test notiamo però che la rete neurale non riesce ad imparare correttamente i valori intorno allo zero. Abbiamo molti valori negativi e molti valori non precisi nell'intorno di zero, soprattutto per quanto riguarda il test sul training (Figura 38).

Possiamo ipotizzare che aumentando le epoche la rete possa risultare più efficiente, ma il tempo estremamente lungo di computazione ci ha impedito di verificare questa ipotesi.

6 Conclusioni

Il progetto ha visto la predizione del consumo energetico di un edificio mediante reti neurali di tipo time series e sistemi di inferenza fuzzy (FIS) di tipo Sugeno e Mamdani.

In una prima fase abbiamo analizzato i dati fornitici per eliminare eventuali errori che potessero compromettere le performance degli strumenti che avremo utilizzato in seguito. Analisi più approfondite sono state necessarie specificamente per le reti neurali time series e per i FIS.

E' stato necessario inoltre sviluppare diversi "script di contorno" al Matlab per poter facilitare e velocizzare il nostro lavoro.

Per ciò che concerne lo strumento delle time series, dopo numerosi tentativi siamo riusciti ad ottenere una rete con delle performance migliori della media delle performance da noi rilevate.

Il nostro FIS Sugeno e Mamdani prevede correttamente i casi in cui il consumo è nullo. Per quanto riguarda invece gli altri casi c'è discrepanza tra i valori ottenuti dalla rete e quelli effettivamente registrati dal fancoil.

Come abbiamo visto dal grafico delle occorrenze all'inizio di questa relazione, è difficile prevedere una relazione logica tra i dati atmosferici ed il consumo dell'impianto. Con dati atmosferici simili abbiamo molte volte consumi diversi, inoltre con gli strumenti che abbiamo non siamo in grado di prevedere quando il condizionatore entra in modalità risparmio energetico, perché la stanza raggiunge la temperatura stabilità; e non siamo in grado di prevedere quando l'azienda chiude per ferie, dato che in quei giorni il consumo è nullo pur essendo dei giorni lavorativi.

Abbiamo inoltre riscontrato che molte volte nell'orario lavorativo, pur essendoci le condizioni climatiche tipiche per l'accensione del condizionatore, quest'ultimo non viene attivato; pertanto è difficile prevedere il comportamento umano.

Tutti questi fattori ci portano ad avere un errore lontano dall'essere nullo e dalla difficoltà di addestrare una rete neurale.

La rete neurale è adatta a lavorare con dati che hanno un'alta correlazione con l'uscita e una certa periodicità nel tempo.

Visti i risultati ottenuti con i vari strumenti possiamo concludere che il più adatto, per il tipo di problema e per i dati fornitici, è il FIS.

Questo progetto ha dimostrato come l'affinità del progettista con il problema e l'esperienza siano qualità fondamentali per ottenere buoni risultati con questi strumenti. In mancanza di una o più di queste qualità è necessario dedicare molto tempo alla sperimentazione.

7 Allegati

Allegato 1: script in Python di generazione campo orario lavorativo e feriale.

Allegato 2: script di test delle varie configurazioni di reti neurali time series.

Allegato 3: risultati script dell'Allegato 2.

Allegato 4: script di test di una sola configurazione di rete neurale time series.

Allegato 5: elenco fuzzy rules.

Allegato 6: script generatore di set.