POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di Laurea Magistrale

in Ingegneria Informatica

indirizzo Data Science

Tesi di Laurea Magistrale

Predizione di tempeste solari tramite reti neurali LSTM

Relatore

Prof. Enrico Magli

Candidato

Andrea Grippi

Dicembre 2018

SOMMARIO

1 INTRODUZIONE

Lo scopo di questa tesi è lo studio e l'analisi degli effetti del vento e delle tempeste solari sulla magnetosfera terrestre.

Per fare ciò sono stati utilizzati indici ottenuti tramite misurazioni nel corso degli anni con i quali si è cercato di addestrare una rete neurale ricorrente.

Il sole nella sua normale attività generale ed emette particelle cariche che spazzano per tutto il sistema solare.
Queste particelle portano con sé il campo magnetico del sole e interagiscono i corpi che incontrano sul loro cammino.
Questo flusso di particelle viene normalmente denominato vento solare.

Il vento solare ha effetti visibili quandunque interagisca con un campo magnetico naturale, come ad esempio nel caso di Mercurio e della Terra.

L'attività del Sole non è costante: può avere delle fasi di maggiore attività che si traducono in e una maggiore intensità delle emissioni di particelle cariche nel sistema solare.

Le tempeste solari possono arrecare danni al nostro pianeta: possono danneggiare i satelliti e interrompere le telecomunicazioni oltre ad avere effetti anche all'interno dell'atmosfera.

In particolare, andrò ad analizzare i dati rilevati nel corso degli anni riguardanti le variazioni dell'intensità del campo magnetico terrestre con l'ausilio e l'utilizzo di reti neurali.
In particolare per questo studio ho utilizzato un tipo di rete neurale ricorrente denominato LSTM.

Esporrò alcune informazioni sull'evoluzione delle reti neurali nel tempo, e partendo dai modelli più semplici arriverò a descrivere nel dettaglio la struttura di queste reti neurali piuttosto recenti, includendo la documentazione relativa alla rete specifica che ho implementato e utilizzato per il mio scopo.

Negli ultimi anni infatti le reti ricorrenti LSTM sono risultate molto utili applicate allo studio di serie temporali e sono ad oggi molto utilizzate nel campo del riconoscimento del parlato e nella comprensione dei testi.

I dati sono stati raccolti nel corso degli anni da stazioni a terra e derivati secondo uno standard definito nel corso di uno studio durato più di 40 anni.
L'indice utilizzato è denominato SYM, successore e dell'indice DST definito verso la fine degli anni Ottanta dal professor Sugiura.

È stato necessario un lavoro iniziale di studio sul dataset, in particolar modo è stato difficile capire come rielaborare il dataset per poterlo rendere utilizzabile dalla rete neurale.
Parte centrale dello studio è stato la creazione di un modello che riuscisse a prevedere l'andamento dell'indice svariate ore nel futuro.

Sono riportati i risultati di tre diversi approcci al problema, accompagnati dalle relative statistiche e grafici esplicativi.
Sono esposte inoltre informazioni utili a comprendere i risultati e le metriche usate per valutare e validare lo studio.

Confronterò il risultato ottenuti grazie alla rete neurale con quelli ottenuti in studi simili con algoritmi di classificazione tradizionali.

In conclusione, andrò a esporre i possibili miglioramenti che ritengo siano implementabili prendendo come punto di partenza il mio lavoro, sia per quanto riguarda lo sviluppo della rete neurale sia per quanto riguarda lo studio relativo al fenomeno fisico.

Sarebbe infatti interessante raccogliere e unificare dati provenienti da fonti differenti, utilizzando il potenziale delle reti neurali per cercare di ottenere una migliore comprensione del dell'attività solare.

2 LE TEMPESTE SOLARI

2.1.1 Il fenomeno

Il fenomeno fisico del vento solare ha le sue origini nell'attività della corona solare, la parte più esterna del sole e della sua atmosfera.
Questa è composta da materiale che a causa delle altissime temperature si trova sotto forma plasmatica, in particolare gas e vapori fuoriusciti dalle parti sottostanti dell'atmosfera, come la cromosfera e la zona di transizione.

Il fenomeno fisico che causa l'elevatissima temperatura è ancora in fase di dibattito e non del tutto compreso.
La corona solare è infatti lo strato con la densità minore del sole (10^{-12} kg/m³) e la sua parte più calda con una media di un milione di gradi Kelvin.
Questa temperatura è impressionante se confrontata con la temperatura della fotosfera che è di 5700K.

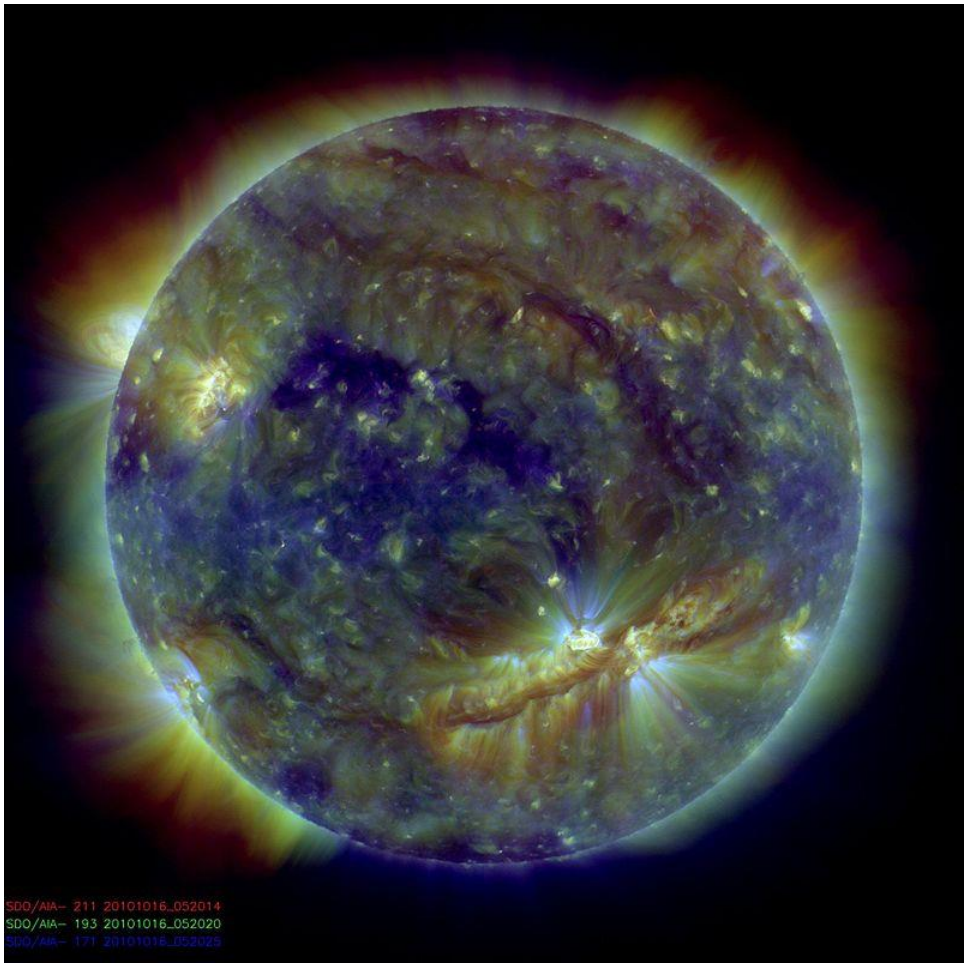


Figura 2-1 - La variabilità della corona solare
(Di NASA)

Un evento molto significativo della sua attività è quello dell'espulsione di massa dall'area della corona solare (CME, Coronal Mass Ejection), l'area più esterna dell'atmosfera del Sole che raggiunge il milione di gradi Celsius.
Il vento solare sprigionato dall'espulsione si spande per tutto il sistema solare ed è composto principalmente da elettroni e protoni con energie comprese tra 1,5 e 10 keV. Sulla Terra arriva a velocità che possono raggiungere i 900km/h.

Il plasma del vento solare porta con sé il campo magnetico del sole, che si espande in modo radiale e le cui linee di forza rimangono collegate alla loro origine nella fotosfera e interagisce con la magnetosfera terrestre, perturbandola in modo anche in modo problematico.

Un certo livello di vento solare è sempre presente all'interno del sistema solare, durante la normale attività del Sole.
Lo studio si è concentrato sulla possibilità di prevedere le tempeste solari, eventi "eccezionali" derivanti da attività atipiche della nostra stella.

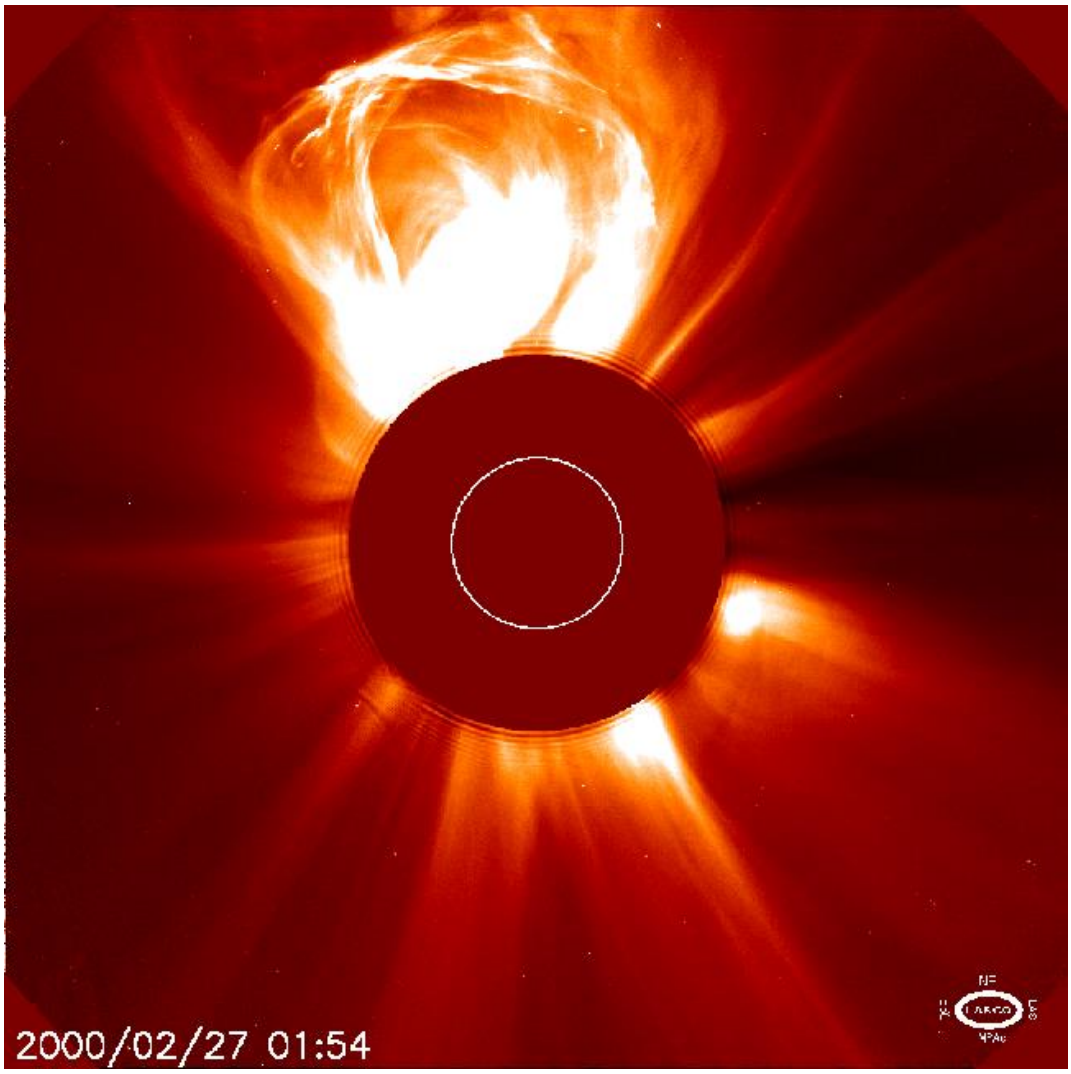


Figura 2 – CMS
(NASA)

2.1.2 Effetti

Tutti i pianeti e le lune del sistema solare sono interessati dagli effetti del vento solare, con cui interagiscono in modo diverso a seconda della composizione del proprio campo magnetico

La Luna non possiede un suo campo magnetico, non possiede alcun metodo di schermatura contro il vento solare che dunque arriva sulla sua superficie direttamente.
Le varie missioni lunari hanno riscontrato la superficie e l'esosfera della Luna siano particolarmente cariche per effetto del vento solare.
Pianeti come Mercurio e la Terra hanno invece una propria magnetosfera che interagisce con il campo magnetico del vento solare in modo diverso a seconda dell'intensità con cui questo arriva nelle vicinanze del pianeta.

Mercurio ad esempio dista circa 58 milioni di chilometri dal Sole (0,387 UA), contro i circa 150 della Terra (1 UA): il vento solare ha l'effetto di "schiacciare" il campo magnetico del pianeta, ed è stato osservato come nei momenti di massima intensità, durante tempeste solari ad esempio, riesca a farlo retrocedere a sufficienza per arrivare a interagire direttamente col suolo mercuriano.

Il vento solare non riesce ad arrivare direttamente sulla Terra, ma viene deviato grazie all'interazione con la magnetosfera terrestre, di cui contribuisce a plasmare la forma per poi scorrere attorno al pianeta.
La magnetosfera non scherma completamente il vento solare, ma si comporta più come un filtro, attenuandone gli effetti.

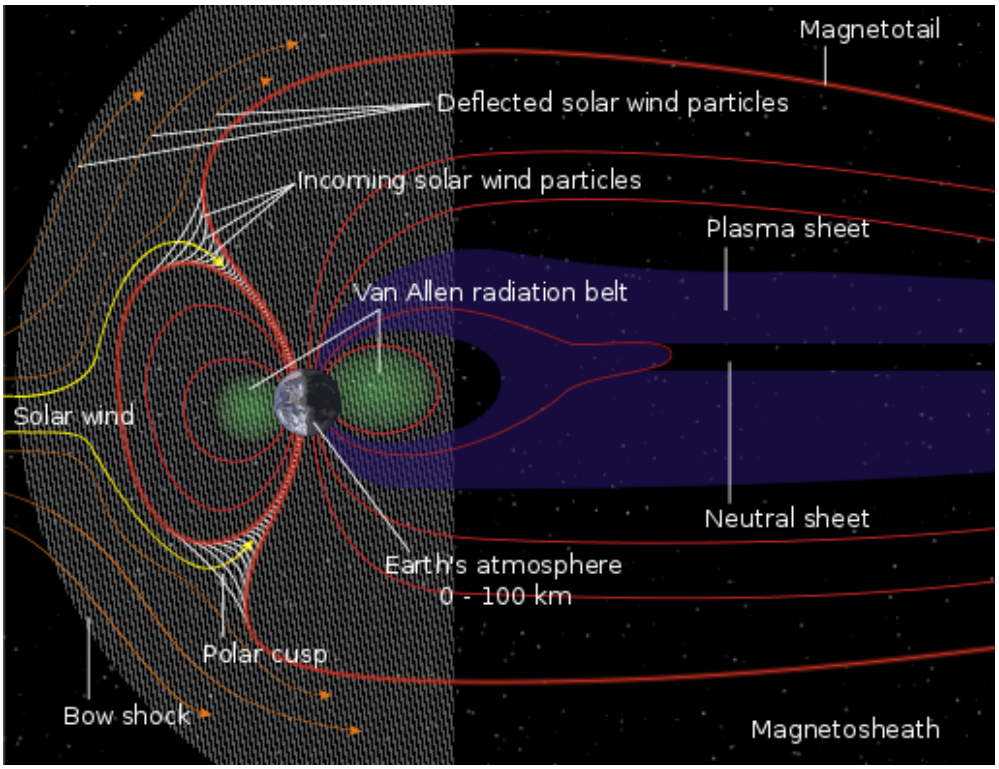


Figura-2-2 - Magnetosfera
(NASA, Aaron Kaase,William Crochot)

L’evento più innocuo e conosciuto è sicuramente l’aurora boreale/australe, causata dall’interazione delle cariche presenti nel vento solare con la ionosfera, i cui atomi vengono eccitati e in seguito rilasciano la luce che caratterizza il fenomeno.

2.1.2.1 Evento di Carrington

Tra il 28 agosto e il 2 settembre 1859 si abbatté sulla Terra la tempesta solare più intensa mai registrata, che prende il nome dall’astronomo inglese Richard Carrington.

L’aurora boreale prodotta fu visibile persino alla latitudine di Roma e di Cuba, ma soprattutto vennero seriamente danneggiate le linee telegrafiche in Europa e in Nord America.

Le stime relative all’intensità dell’evento vanno dai -800nT ai -1750nT, ed è stato stimato da uno studio dell’American Geophisical Union (AGU) che un evento del genere oggi potrebbe causare danni per svariati miliardi di dollari, con tempi di ricostruzione delle infrastrutture anche molto lunghi.

3 LE RETI NEURALI

3.1 PERCEPTRON

La storia delle reti neurali inizia nel 1958 quando Frank Rosenblatt presenta il modello del Percettrone, un classificatore binario progenitore delle moderne reti neurali, in grado di riconoscere e imparare semplici pattern basato sul modello del neurone biologico.

È composto da un unico nodo, che riceve degli input, ne calcola la somma sottraendone una soglia e restituisce il risultato ad una funzione gradino.

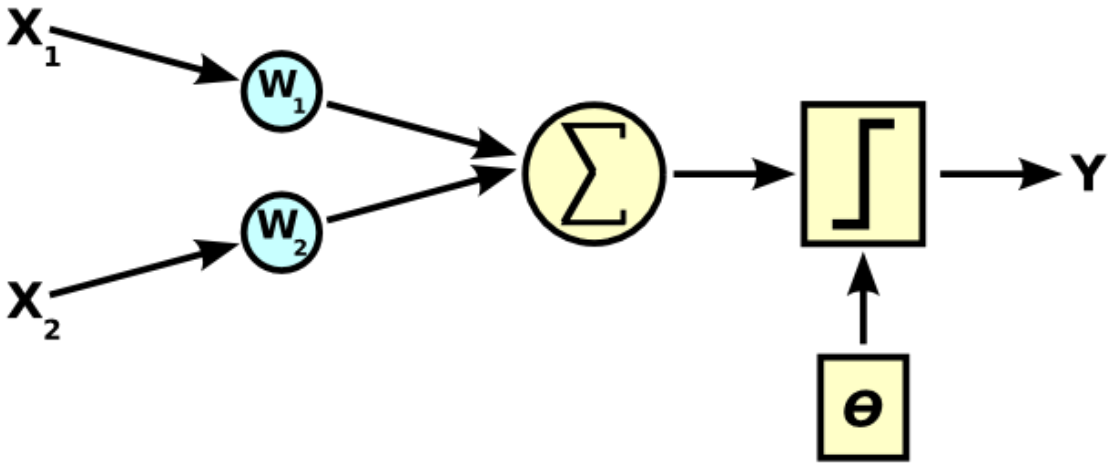


Figura 4 – Rappresentazione del Perceptron
(Wikipedia - CC BY-SA 3.0)

Nel 1969 Minsky e Papert scrivono la loro opera "*An introduction to computational geometry*" in cui dimostrano come alcune premesse del lavoro di Rosenblatt non fossero verificate, fermando di fatto lo studio sull'intelligenza artificiale per quasi vent'anni.
La limitazione maggiore del perceptron era l'essere un modello troppo semplice.

Si arrivò presto alla conclusione che un modello a più livello sarebbe riuscito a risolvere problemi più complessi in modo efficace, ma all'epoca non era disponibile alcun algoritmo che permettesse l'addestramento di una rete a più livelli.
Bisognerà aspettare il 1986 perché Rumelhart presenti il suo algoritmo di retropropagazione (BP o Back propagation) rendendo possibile la creazione di reti neurali multistrato.

3.2 RETI MULTI-STRATO (MLP)

L'evoluzione naturale del semplice percettrone sono le reti dette MLP (Multi-layer perceptron), basate proprio sull'algoritmo di apprendimento creato da Rumelhart, il back propagation.

Definizione: È un algoritmo che minimizza l'errore quadratico medio fra l'output corrente e quello desiderato di un MLP. Procede modificando i pesi in modo da muoversi in direzione opposta al gradiente della funzione di errore (calcolato rispetto ai pesi).

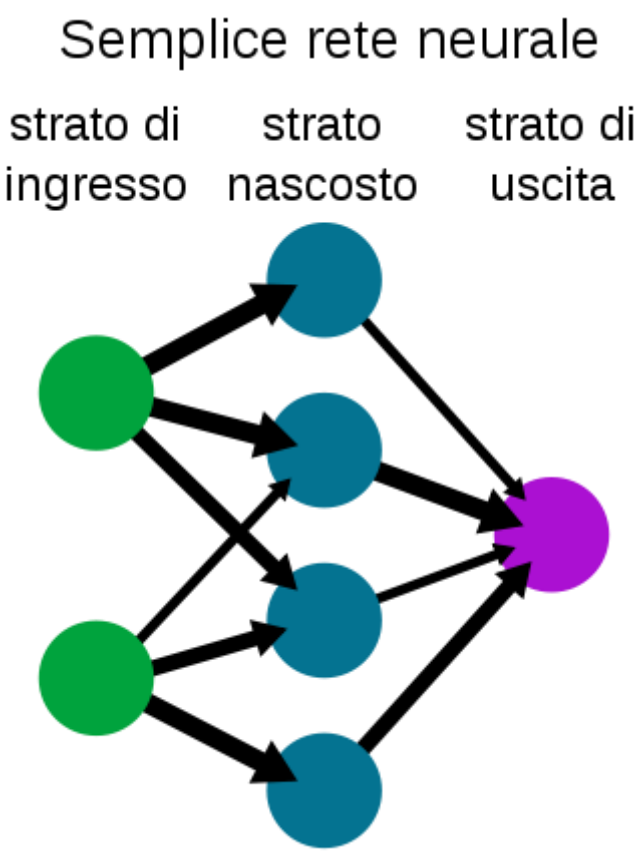


Figura 5 – Rete MLP

Questo tipo di reti che non formano cicli interni ma propagano gli input solo in "avanti" vengono definite reti **feedforward**.

Ogni neurone ha un peso, e questi vengono inizializzati generalmente in maniera causale: la moltiplicazione del peso con l'input determina l'attivazione o meno del singolo neurone, cosa che indica la rilevanza del singolo dato in ingresso.

Esistono vari metodi e tecniche di apprendimento che possono essere usati per l'apprendimento delle reti feedforward, ma il più popolare è sicuramente quello di retro-propagazione dell'errore (o back-propagation) in cui l'output viene continuamente confrontato con l'output desiderato.

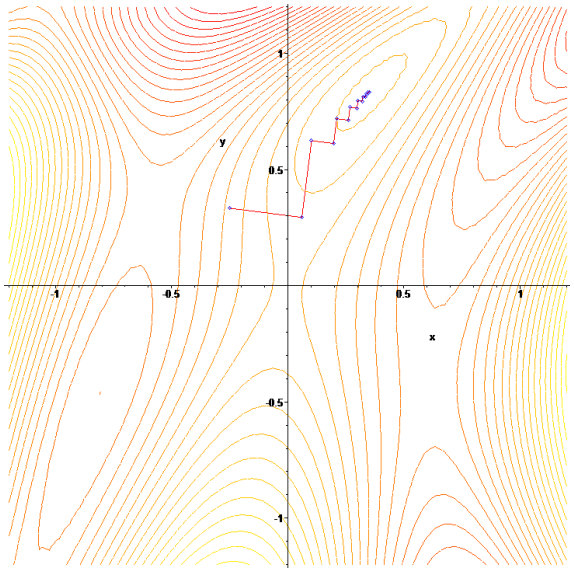
L’errore viene dunque dato in pasto alla rete assieme all’input, modificando i pesi e permettendo alla rete di imparare.

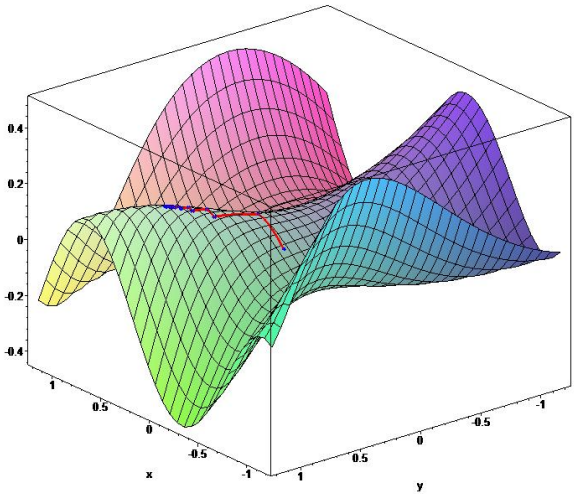
$$E(w) = \frac{1}{2} \sum_h \sum_k (out_k^h - y_k^h)^2$$

L’ Algoritmo di retropropagazione dell'errore

Per ottimizzare la modifica dei pesi si applica un’operazione chiamata “discesa del gradiente”, che consiste nella derivazione della funzione d’errore per trovare il minimo locale.

Nelle immagini seguenti sono riportate due visualizzazioni grafiche del processo di discesa del gradiente.





3.3 RETI RICORRENTI (RNN)

Le reti neurali ricorrenti sono delle reti le cui interconnessioni interne formano dei grafi orientati.

La differenza con le reti feedforward sta inoltre nella possibilità delle RNN di immagazzinare il loro stato interno ed usarlo per processare sequenze di dati in ingresso.

Grazie alla loro natura le reti ricorrenti si sono dimostrate molto efficaci su dataset con dati mancanti, come ad esempio la scrittura manuale e il riconoscimento del parlato.
Sono state anche utilizzate con successo nel campo della compressione dei dati.

Il metodo di ottimizzazione utilizzato è la discesa del gradiente, che però nelle reti ricorrenti tende a "svanire" nel caso di dataset in cui intercorrano grandi lassi di tempo tra gli eventi significativi.

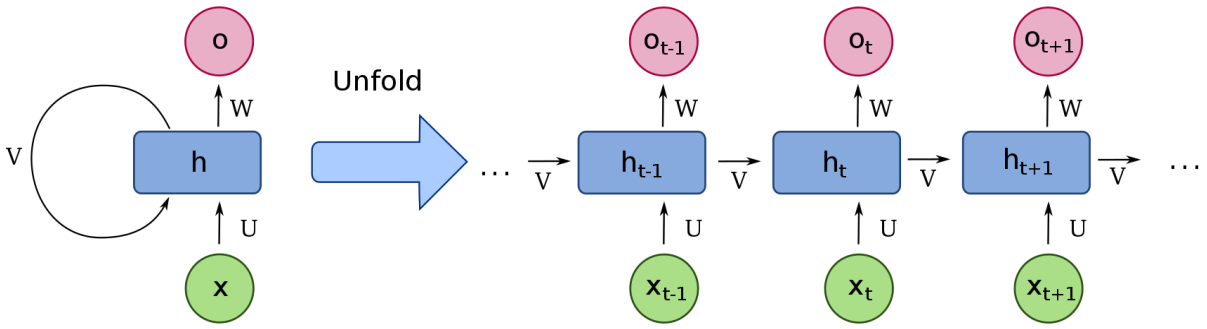


Figura 6 – Schema di una RNN
(By François Deloche - Own work, CC BY-SA 4.0)

Sono state trovate alcune soluzioni al problema della scomparsa del gradiente, con diversi gradi di successo:

- Aumento potenza di calcolo
    - L'utilizzo delle GPU e l'aumento della potenza di calcolo degli elaboratori rispetto al 1991 consente di tamponare il problema, ma non lo risolve
- Introduzione delle GRU (Gated recurrent units)
    - Meccaniscmo di gating alternativo nelle RNN, ottengono risultati paragonabili o migliori delle LSTM su dataset però più ridotti
- Reti LSTM

3.4 LSTM

Le reti neurali LSTM (sigla dall'inglese Long short-term memory, memoria a lungo-breve tempo) sono state proposte nel 1997 da Sepp Hochreiter e Jurgen Schmidhuber e sono una variante delle reti ricorrenti classiche.
Il loro nome deriva dalla particolare natura delle celle di tipo LSTM, che come vedremo a breve, permette a questo tipo di reti di essere molto efficace nello studio di serie temporali.

Dal 2000 in avanti sono state perfezionate e utilizzate in misura sempre maggiore anche da grandi aziende informatiche come Google, Apple e Microsoft.
Si sono rivelate molto efficaci applicate alla compressione di testi in linguaggio naturale, nel riconoscimento del parlato e raggiunto record in alcuni test standard.

In particolare, le reti LSTM risolvono il problema più serio delle RNN, cioè la scomparsa del gradiente nelle particolari condizioni descritte sopra.

3.4.1 Unità LSTM

Le celle LSTM sono in grado di processare i dati in maniera sequenziale e mantenere il loro stato nascosto al trascorrere del tempo.

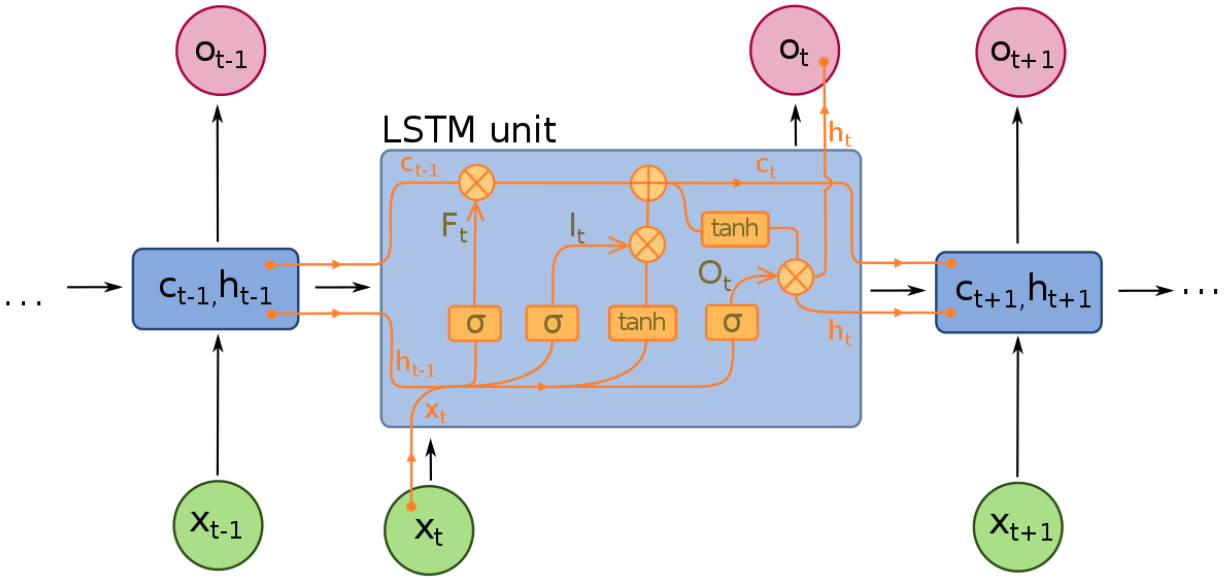


Figura 7 – Le unità che compongono le LSTM
(By François Deloche - Own work, CC BY-SA 4.0)

Dove:

- X_t: Vettore d'ingresso
- h_{t-1}: Output del blocco precedente
- c_{t-1}: Memoria del blocco precedente
- σ: Funzione Sigmoidea
- x: Moltiplicazione
- +: Somma
- *tanh*: Tangente iperbolica

Ogni cella LSTM dunque genera un nuovo output dopo aver valutato l'input corrente assieme all'output e alla memoria precedente.
Ad ogni output generato, la cella ha la capacità di modificare il vettore di stato *C*, aggiungendo o rimuovendo informazioni.

Le porte, o *gates*, sono costituite da un livello σ da un'operazione di moltiplicazione e controllano la quantità d'informazione che deve essere lasciata passare.

Il primo strato sigmoide da sinistra è chiamato *forget gate*, regola quanta informazione ricevuta in forma vettoriale dalla cella precedente debba essere dimenticata, portando in uscita un array composto di valori 0 o 1.

Il second gate è chiamato "input gate", e decide quali valori di *C* andranno modificati, e unitamente al livello *tanh* che genera dei nuovi valori possibili, va ad aggiornare lo stato della cella.

L'output della cella viene infine generato andando a modificare lo stato della cella, filtrandolo attraverso uno strato *tanh* e generando un array con valori in un range tra -1 e 1 che viene moltiplicato per il risultato dello strato sigmoideo, in modo da selezionare correttamente quali elementi modificare e portare in uscita l'output h_t.

4 I DATI

4.1 L'INDICE DI DISTURBO MAGNETICO

4.1.1 DST – Disturbance Storm-Time

Lo studio per la definizione di un indice che potesse misurare le correnti e le variazioni del campo magnetico attorno alla Terra risalgono alla fine degli anni Quaranta dall'Associazione internazionale per il Geomagnetismo e Aeronomia (International Association of Geomagnetism and Aeronomy).

Alcuni tentativi vennero portati avanti durante gli anni Cinquanta e l'inizio dei Sessanta, finchè non venne proposto uno schema di derivazione per l'indice denominato DST nel 1967 da M. Sugiura e S. Hendricks.

Era infatti noto già da tempo, fin dalla metà del diciannovesimo secolo (Broun, 1861) che la componente orizzontale del campo magnetico venisse compressa durante particolari fasi di disturbo magnetico.
I vari studi hanno evidenziato che all'Equatore e a medie latitudini la varizione della componente orizzontale (H) del campo magnetico fosse approssimabile ad un campo magnetico parallelo all'asse del dipolo geomagnetico.

Tempeste solari causano dunque una variazione misurabile della componente H della magnetosfera, formalizzata con l'indice DST (Disturbance Storm-Time, Disturbo causa tempesta su base temporale), che fornisce un indice dell'intensità della tempesta stessa.

Di seguito riporto la distribuzione originale delle stazioni di rilevamento dell'indice DST (1981).

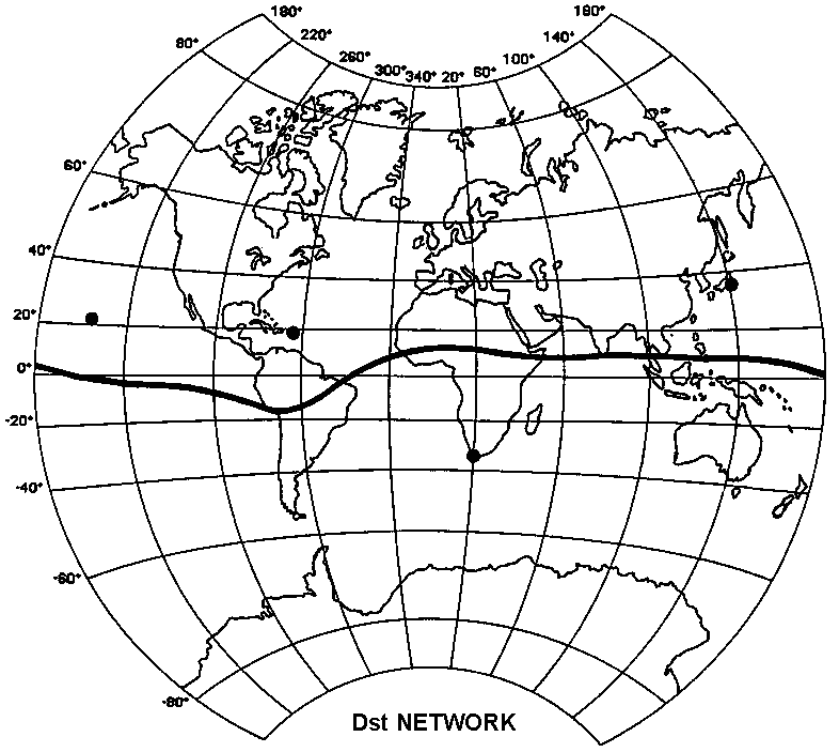


Figura 8 - Distribuzione delle stazioni di rilevamento DST

4.1.2 SYM - **Symmetric disturbance**

L'indice SYM è quello su cui effettivamente si è basato lo studio.

Stando alla definizione ufficiale, l'indice SYM è stato creato per poter descrivere le variazioni nella magnetosfera in termini di disturbi longitudinalmente simmetrici nelle componenti H e D (parallela e perpendicolare) rispetto all'asse del dipolo terrestre.

È dimostrato che SYM-H e DST siano equivalenti in termini di valori assunti, mentre differiscono in termini di risoluzione temporale. SYM ha infatti una risoluzione di un minuto, mentre DST di un'ora.

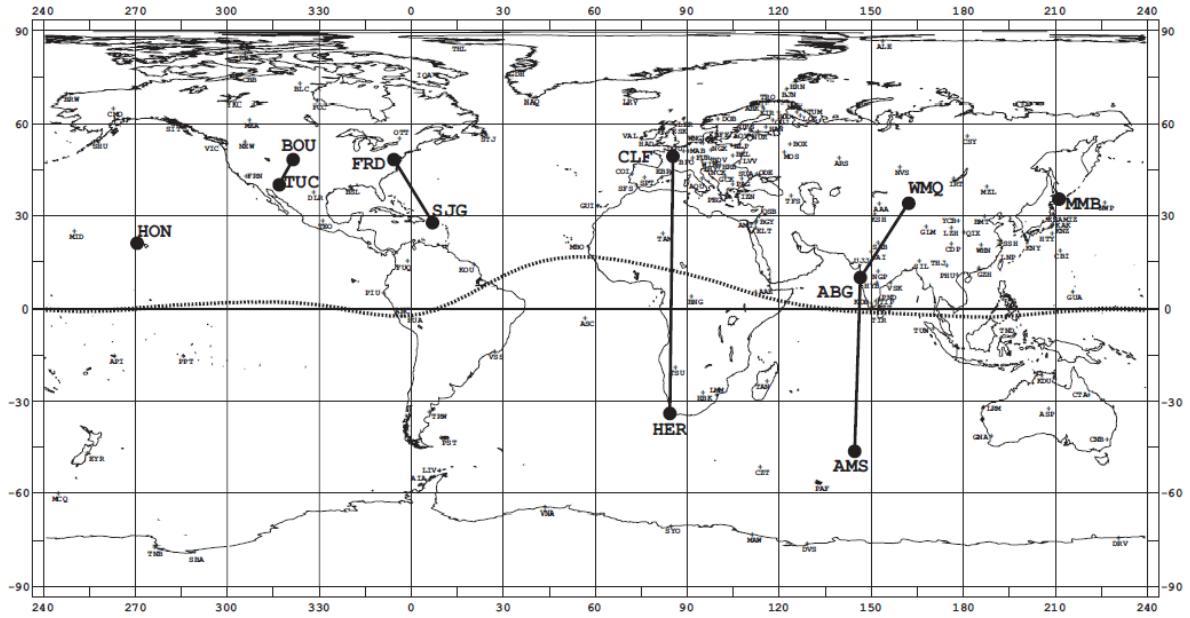


Figura 9 - Posizione delle stazioni di misurazione indice SYM

Derivazione

L'indice SYM viene ricavato in tre passi:

1. Differenza tra il campo geomagnetico e la media degli ultimi 5 giorni di "quiete"
2. Le coordinate vengono trasformate sul sistema di coordinate del dipolo
3. Calcolo del valore SYM come media dei rilevamenti di 6 stazioni

4.2 IL DATASET

Il dataset su cui è stato eseguito lo studio è un file testuale (.txt) costituito da 6.311.520 voci, strutturate come segue:

EPOCH_TIME		SYM/D_INDEX	SYM/H_INDEX
dd-mm-yyyy	*hh:mm:ss.ms*	*nT*	*nT*
01-01-2005	00:00:00.000	-2	-2

Dove:

- Epoch Time indica il giorno e il momento di acquisizione del valore
- SYM/D – valore indice nella componente ortogonale
- SYM/H – valore indice nella componente orizzontale

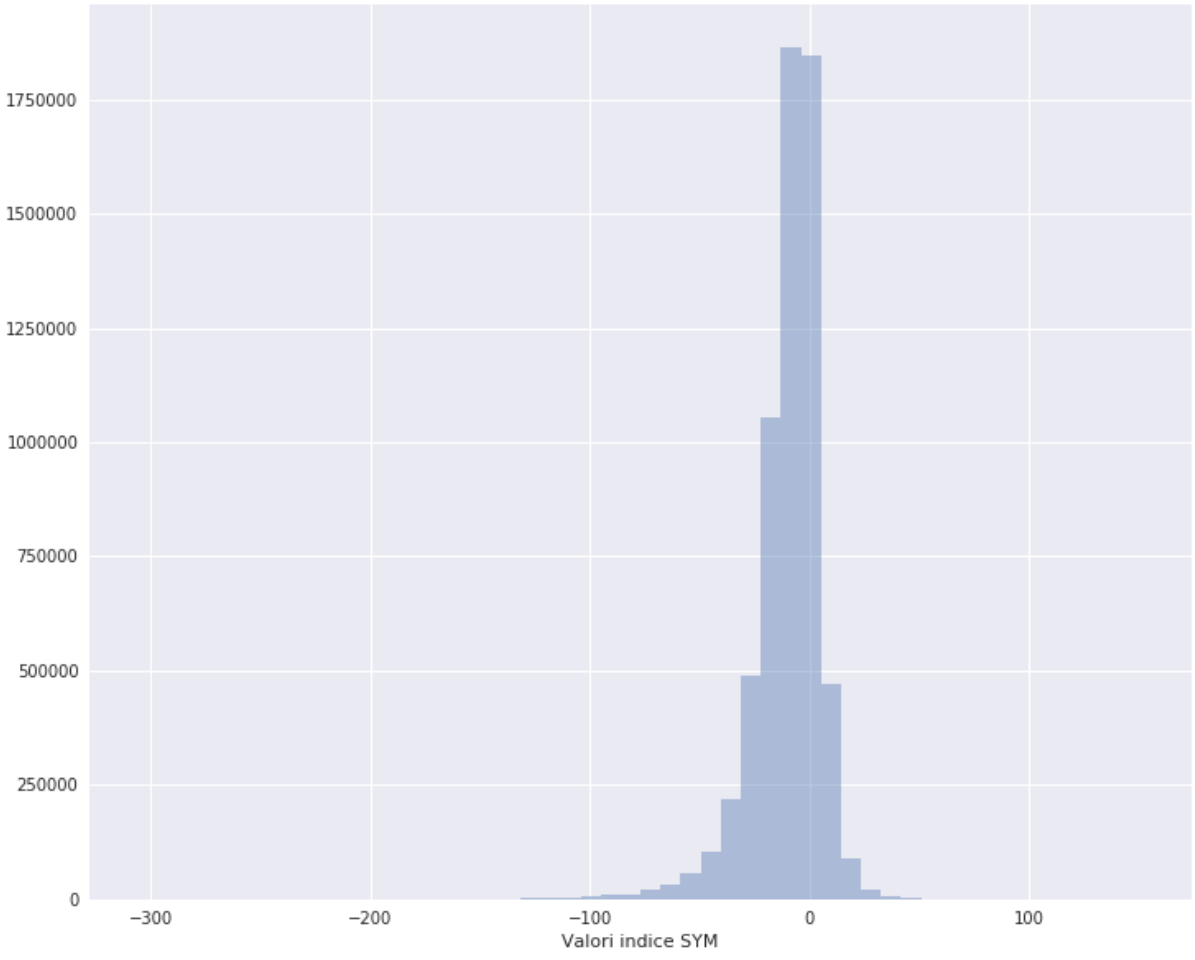
I dati sono stati raccolti dal 01/01/1981 al 31/12/2016 e forniti dal WDC Kyoto (World Data Center for Geomagnetism – Il centro dati mondiale per il geomagnetismo).

4.3 LA DISTRIBUZIONE DEI DATI

Una delle maggiori complicazioni nello studio deriva dalla distribuzione dei dati.

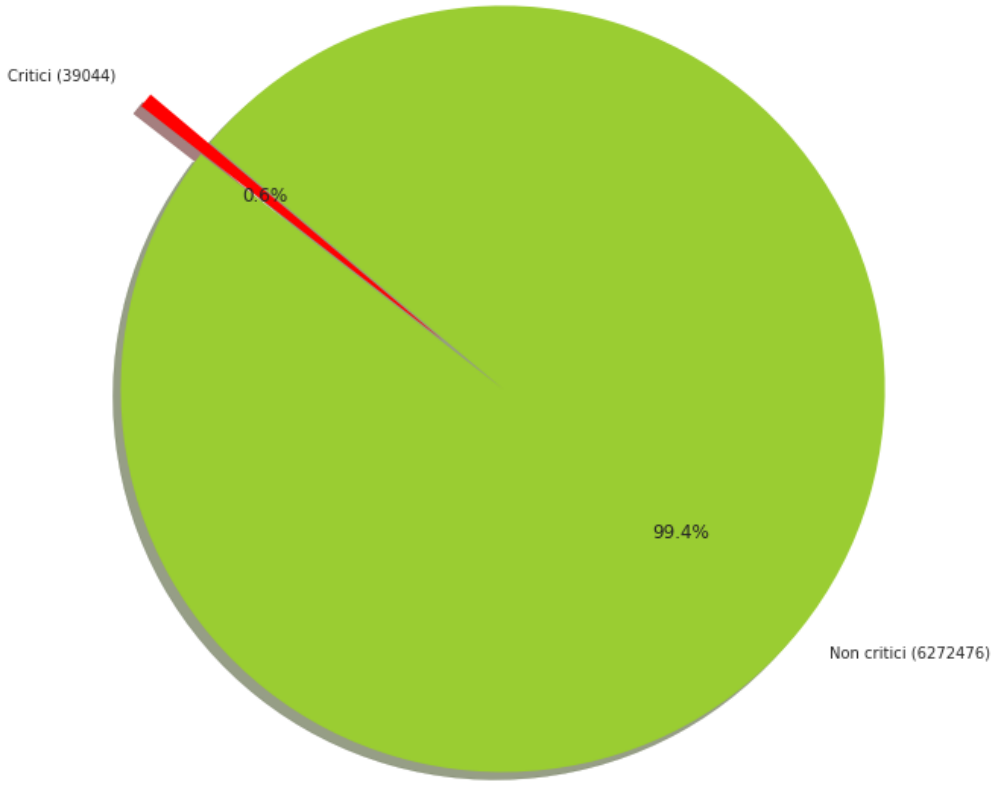
L'indice SYM infatti oscilla quasi costantemente tra gli 0 e i -20nT, e solo in pochi casi scende oltre i -50nT.

Nell'immagine seguente è possibile visualizzare la distribuzione dei dati: la coda a -300nT della distribuzione rappresenta un evento praticamente unico, ma la situazione non è molto diversa andando a considerare la soglia dei -75nT.



La soglia significativa è quella dei -75nT, oltre la quale viene identificato un evento significativo di variazione nella geosfera e una possibile manifestazione di una tempesta solare.

Il grafico successivo suddivide il dataset tra valori critici (<= -75nT) e non critici (>-75nT). Com'è possibile osservare, solo lo 0,6% del dataset ricade nella parte critica, mentre il restante 99,4% è costituito da valori non critici, rappresentanti attività solare considerata "normale".



Gestione del problema

Il problema è stato affrontato durante lo studio dei dati e l'implementazione delle reti.

Per correggere l'effetto dello sbilanciamento delle classi ho scelto come tecnica di pre-processamento il sotto-campionamento del dataset.

I parametri di sotto-campionamento sono riportati nella parte di esposizione dell'implementazione delle reti neurali.

5 LA RETE

5.1 STRUTTURA

Come specificato in precedenza il software è stato sviluppato in Python 2.7.

Per creare la rete neurale e il suo modello ho utilizzato Keras, una libreria di alto livello che sfrutta il back-end di Tensorflow.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.metrics import mean_squared_error

model = Sequential()
model.add(LSTM(64, input_shape=(1, look_back))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse', 'mae'])
model.summary()

history = model.fit(trainX, trainY, epochs=200, batch_size=100, verbose=2)
```

Figura 10 - La rete

Keras offre un'astrazione di alto livello per la creazione di reti LSTM, in cui è possibile definire svariati parametri.

In particolare, ho impostato a 64 la dimensionalità dello spazio di output e definito la forma dell'input come (1, look_back), cioè un array alla volta di dimensione look_back.

Lo screenshot in questione proviene dallo script preposto ad effettuare una previsione utilizzando il metodo della regressione.

Parametri importanti per la definizione del modello sono ancora:

- **Funzione di loss**:
    - **Regressione**: MSE (Errore quadratico medio)
    - **Classificazione**: Binary crossentropy, Categorical crossentropy
- **Ottimizzatore**: Adam

Adam è un algoritmo di ottimizzazione alternativo al classico SGD (Stochastic gradient descent) basato sul calcolo di media esponenziale mobile del gradiente.

In particolare è stato scelto per le varie prove perché è computazionalmente leggero, performante in termini di requisiti di memoria oltre al fatto di essere spesso utilizzato in presenza di grandi quantità di dati e di parametri.

I parametri di Adam possono essere modificati e manipolati a piacimento, ma per il mio scopo mi sono limitato ad utilizzare quelli standard di Keras, basati sul paper originale, che riporto di seguito:

```
lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False
```

Di seguito inoltre, una rappresentazione grafica della struttura della rete neurale utilizzata.

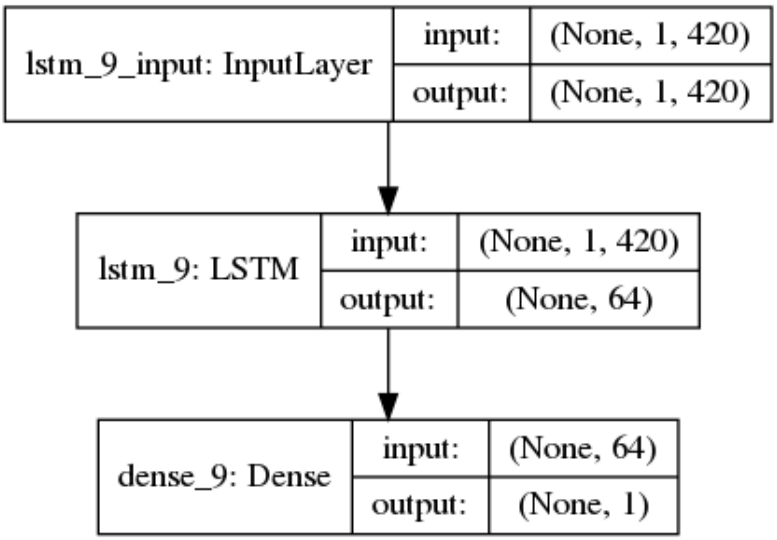


Figura 5-11 - Rappresentazione della struttura della rete

5.2 PREPARAZIONE DEI DATI

Una parte significativa del lavoro è consistita nel capire come poter organizzare i dati e darli in input alla rete LSTM.

I dati, letti da file, vengono inizialmente organizzati in un dataframe utilizzando la libreria Pandas, vengono normalizzati al range [0,1], suddivisi in train set e test set e infine raggruppati in due strutture dati denominate *dataX* e *dataY* nell'esempio seguente.

La prima consiste nella matrice degli array di input, la seconda è invece la lista delle rispettive etichette di riferimento.

```
# trasformo i dati da array a matrice
def create_dataset(dataset, look_back, steps_in_future):
    dataX, dataY = [], []
    i = 0

    while i < len(dataset)-look_back-steps_in_future:
        dataX.append(dataset[(i):(i+look_back), 0])
        dataY.append(dataset[(i+steps_in_future+look_back), 0])
        i+=1

    return numpy.array(dataX), numpy.array(dataY)


dataframe  = pd.DataFrame( data=h_index)
dataset = dataframe.values
dataset = dataset.astype('float64')

# normalizzo il dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)

# Divido in train e test
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
# reshape into X=t and Y=t+1

steps_in_future = 30 # passo di campionamento
look_back = 420 # lunghezza dell'array

trainX, trainY = create_dataset(train, look_back, steps_in_future)
testX, testY = create_dataset(test, look_back, steps_in_future)
```

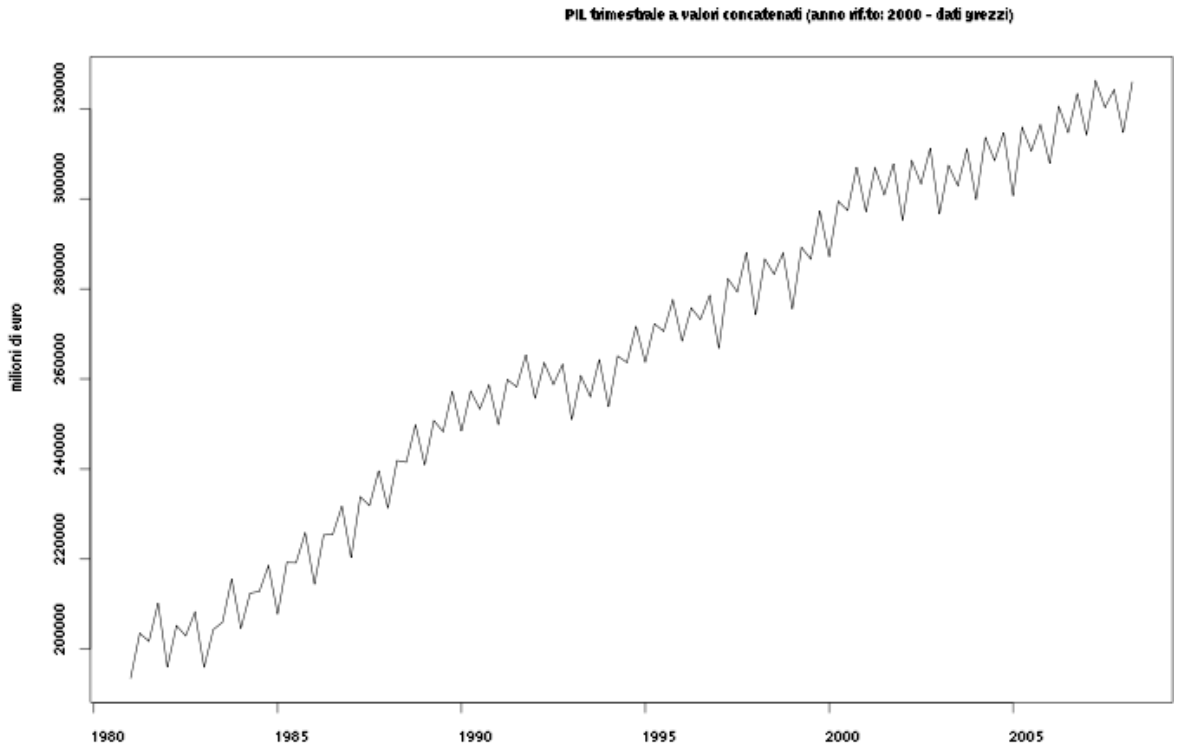
Figura 12 - Trasformazione dei dati

6 IMPLEMENTAZIONE

6.1 IL PROBLEMA DA AFFRONTARE

Per comprendere le motivazioni che giustificano lo l'applicazione e lo studio di reti di tipo LSTM bisogna sicuramente comprendere il tipo di dati su cui si sta andando a lavorare.

Il dataset contenente i dati relativi al SYM index si configura come esempio classico di serie temporale, cioè dati raccolti ad intervalli regolari e su cui è spesso possibile applicare algoritmi di previsione.



Sulle serie temporali esiste una quantità considerevole di esempi in letteratura relativi agli algoritmi classici e alle pratiche da adottare nei diversi casi.

L'uso di reti LSTM è piuttosto recente, ma si è rivelato piuttosto efficace grazie alla sua capacità di "ricordare" e riconoscere dipendenze tra dati anche piuttosto distanti.
In particolare, le reti LSTM sono particolarmente solide nel caso di valori mancanti o di intervalli non regolari.

Analisi di questo tipo sono particolarmente usate ad esempio in ambito commerciale, finanziario o nel campo della meteorologia, e di solito ci si occupa di mettere insieme tre fattori costitutivi: l'andamento, la stagionalità e il rumore.

L'INDICE SYM

Come si evince dal grafico seguente, a cui ho aggiunto l'indicazione per le due soglie significative a -25nT e -75nT, l'indice SYM ha un andamento che non sembra mostrare particolari periodicità. Il fatto che gli eventi significativi siano presenti in misura decisamente inferiori all'andamento standard giustifica l'utilizzo di una rete LSTM per l'analisi rispetto a metodi di analisi classici.

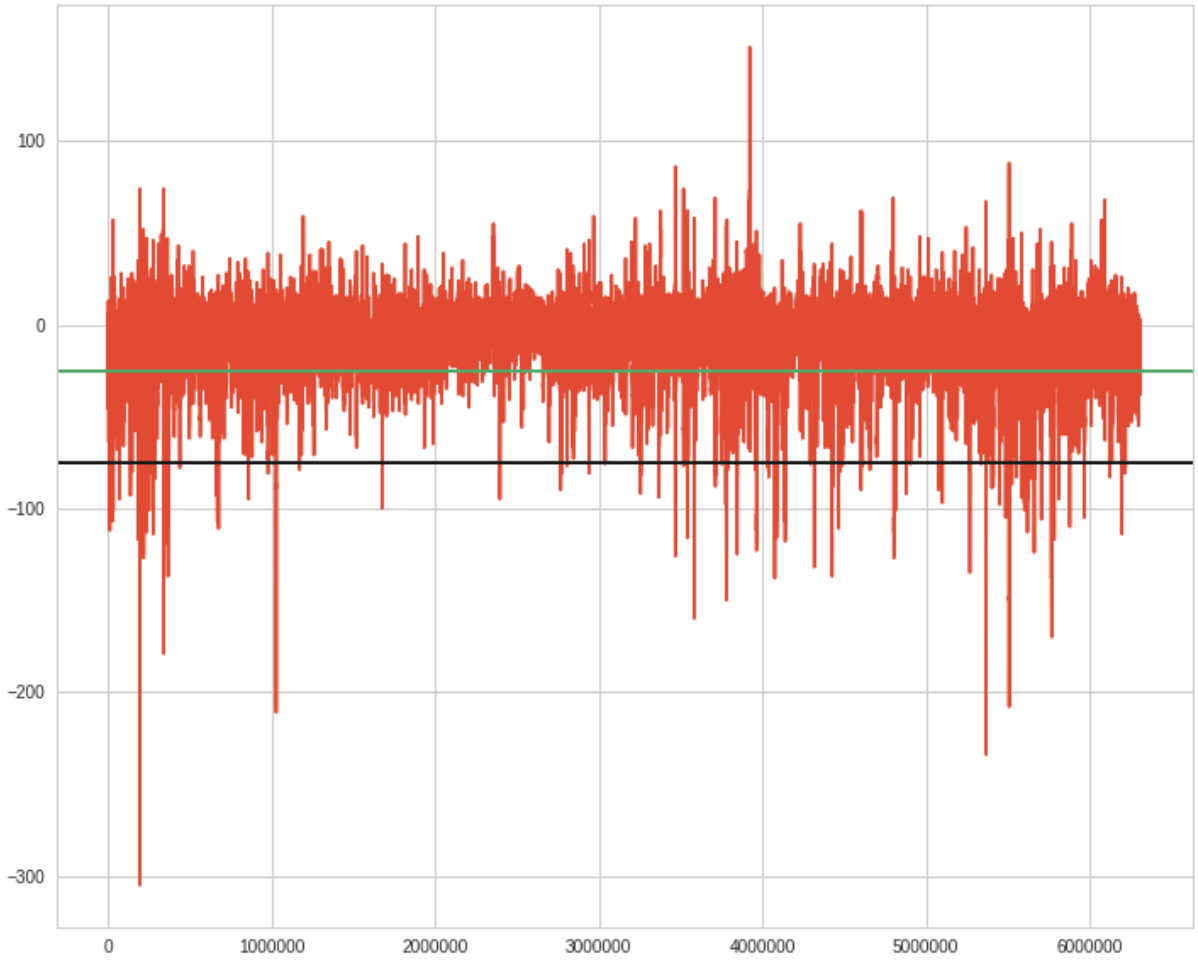


Figura 13 - Il dataset dopo il sottocampionamento, la riga nera segnala la soglia dei -75nT e quella verde la soglia dei -25nT

Il problema della previsione degli eventi significativi può essere anche visto da un'altra prospettiva, cioè quella del rilevamento di anomalie.

L'utilizzo di reti neurali per il rilevamento delle anomalie è un campo piuttosto nuovo, in cui si inserisce questo studio.

Tra i metodi di studio dei casi di rilevamento delle anomalie si possono citare la STL, decomposizione della'andamento stagionale, e il CART, una tecnica di machine learning che sfrutta la classificazione e gli alberi decisionali (regression trees).

6.1.1 Le soglie

Le soglie significative sono essenzialmente due: -25nT e -75nT.
Come è stato detto in precedenza, i valori dell'indice tendono a oscillare tra gli 0 e i -20nT in caso di attività normale.

Tra i -25nT e i -75nT si è in presenza di un'attività significativa del vento solare, ma non ancora classificabile come tempesta solare, che viene invece individuata oltre la soglia dei -75nT.

6.1.2 Input

Per provare ad andare a correggere parzialmente il problema dello sbilanciamento delle classi ho applicato una logica di sotto-campionamento del dataset, andando a prendere un campione ogni 8 dal dataset.

Il dataset sotto-campionato è stato quindi diviso in training e test set, utilizzando una partizione di 0,67.

Come dati di input ho quindi generato *n* vettori di 420 elementi, ciascuno sfalsato di uno rispetto al precedente fino a prendere l'intero dataset.
Per generare le etichette da usare per la classificazione ho usato lo stesso metodo, sfalsato però i vettori di 30 elementi.

Considerando che gli elementi utilizzati hanno una distanza di 8 minuti l'uno dall'altro a causa del sotto-campionamento iniziale, le etichette si trovano ad essere avanti di 4 ore rispetto ai corrispettivi valori nei vettori di input.

I vettori sono composti dai valori dell'indice SYM nelle ultime 56 ore.
Ho provato a dare alla rete più informazioni possibili, in modo da avere la possibilità di raggiungere una precisione più alta avendo più elementi a disposizione per apprendere.

Riporto inoltre per completezza una "sezione" del dataset, in modo da poter visualizzare più da vicino l'indice e il suo andamento nel tempo.
Nell'esempio in questione sono riportati 500 valori non normalizzati estratti dal dataset.

Figura 13 - Una sezione dell'indice SYM

6.1.3 Normalizzazione

Sin dall'inizio dello studio sulle reti LSTM e le prime fasi di implementazione e test, è emersa l'importanza della normalizzazione dei dati.

Come già ampiamente trattato in precedenza, il dataset ha valori che vanno dai +150nT ai -300nT concentrati intorno allo zero.

6.2 I TEST

Le reti permettono di lavorare con facilità sia in regressione che in classificazione.
Ho dunque lavorato a tre script che permettessero di implementare la stessa rete in modalità diversa:

1. Regressione
2. Regressione + Classificazione
3. Classificazione

La versione 1 implementa la rete neurale lavorando in regressione pura, mentre la versione 3 in classificazione pure.

La versione 2 invece è un misto delle due modalità di analisi: la rete viene addestrata in regressione e i risultati vengono trasformati in etichette corrispondenti alle categorie.
Sue questo risultato trasformato viene calcolata l'accuratezza della rete, in modo da avere un risultato confrontabile con il metodo 3.

6.2.1 Regressione

Metriche

La rete è stata addestrata sui dati normalizzati usando come funzione di loss l'MSE (Mean Squared Error).

Come metriche aggiuntive per valutare le prestazioni della rete ho utilizzato il MAE (Mean Absolute Errore), calcolato durante l'addestramento e definito come segue:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Sul risultato denormalizzato della rete ho infine calcolato l'RMSE (Root Mean Squared error) a titolo di confronto. I risultati sono mostrati nella tabella e nei grafici che seguono.

Risultato regressione in training su dati normalizzati:

MAE	**MSE** (loss)
0,013	4,062*10^-4

RMSE calcolato su dati denormalizzati

	RMSE
Train	8.95
Test	10.24

Il risultato complessivo della rete nel predire l'andamento dell'indice si può osservare nel grafico seguente, in cui il dataset viene rappresentato in blu, la predizione sul train set in arancione e quella sul test set in verde.

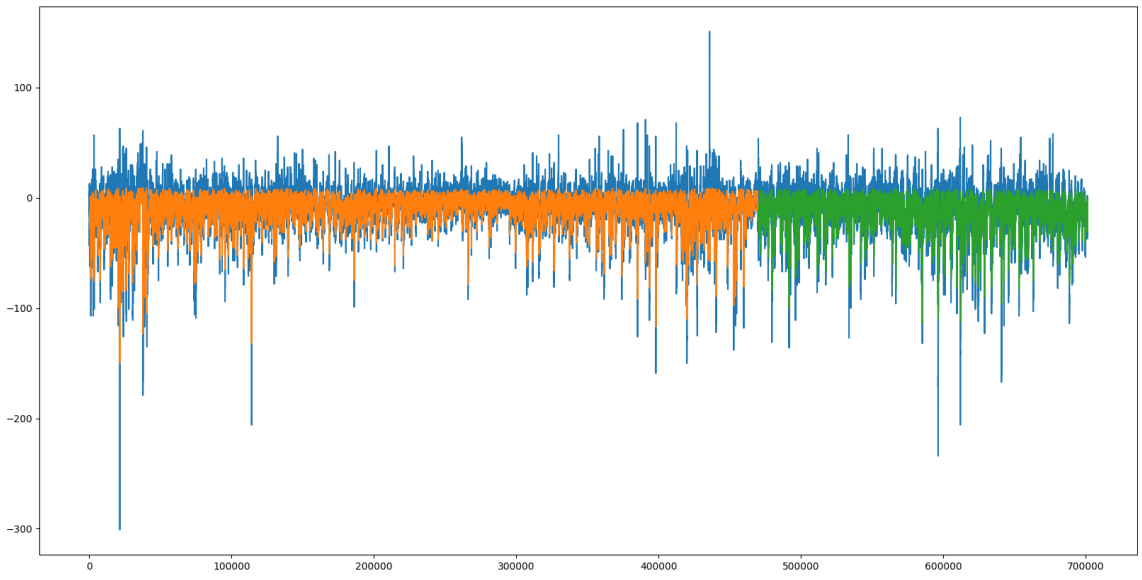


Figura 14 - Predizione su tutto il dataset

Da questo grafico si evince come la rete sia riuscita a prevedere con molta precisione l'andamento della rete, soprattutto nel caso degli eventi significativi.
È soprattutto visibile come la rete sia influenzata dalla forte regolarità dell'indice, risultando in stime quasi sempre conservative per gli eventi negativi e una generale difficoltà di previsione per la parte positiva del dataset, forse causa della normalizzazione.

Se la mancata dei valori positivi non crea alcun problema ai fini dello studio, valori troppo conservativi nel caso degli eventi negativi potrebbero essere causa di un mancato allarme. Per tentare di risolvere questo problema si è passati in una fase successiva dello studio al secondo metodo di implementazione (regressione + classificazione).

Nello specifico:

Nel grafico seguente invece ho riportato i primi 6420 valori del dataset per mostrare visivamente il risultato del training.

La linea rossa rappresenta i dati su cui è stato effettuato l'addestramento, mentre la linea verde rappresenta il valore che si sta cercando di prevedere in un determinato momento a partire dal train set.
La predizione quindi, di colore arancione, dovrebbe idealmente cercare di ricalcare il più accuratamente possibile il grafico verde.

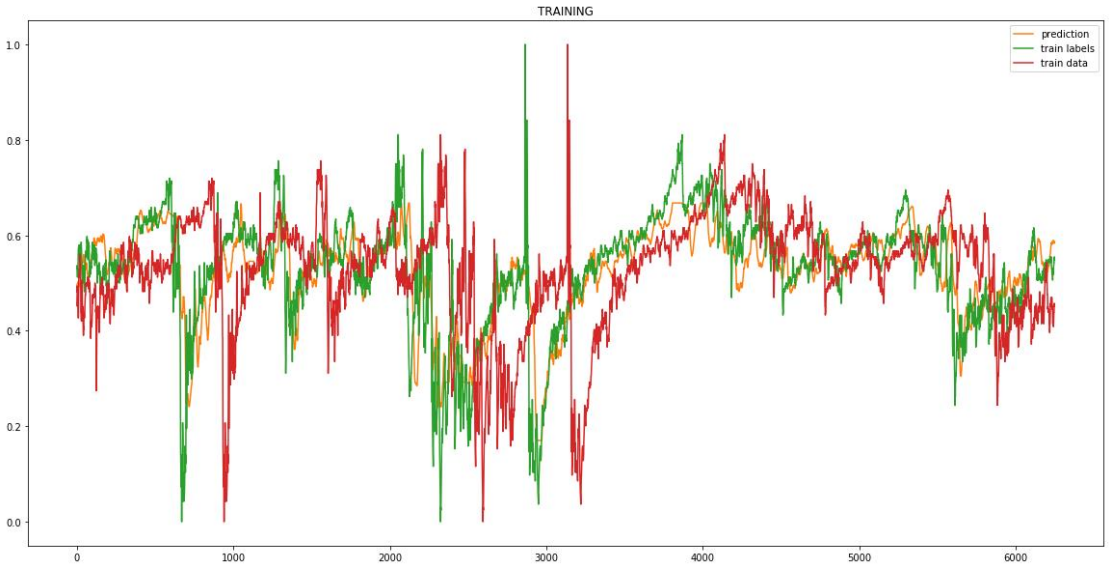
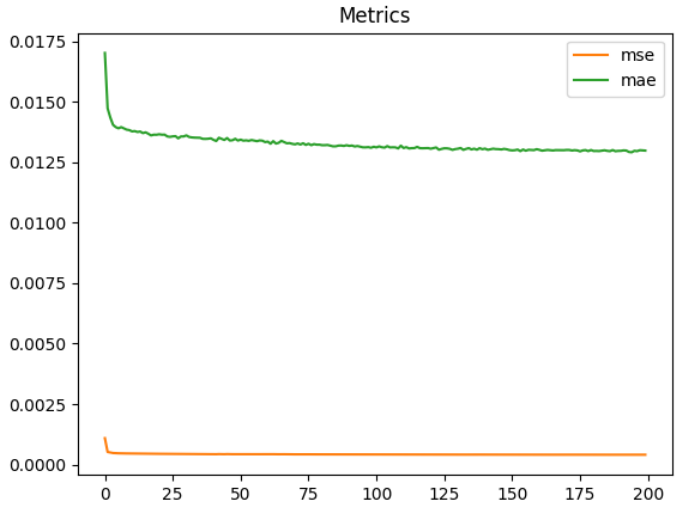


Figura 15 - Visualizzazione congiunta test, training e prediction

Stando al grafico e il valore della funzione di loss esposto in precedenza è possibile affermare che la rete sia riuscita a comprendere il problema da risolvere e sia quindi in grado di fornire un modello adeguato ad effettuare una predizione.

Di seguito, le prestazioni d'apprendimento della rete.



6.2.2 Regressione + Classificazione

Il secondo metodo che ho utilizzato per cercare di predire l'andamento dell'indice consiste nel riutilizzare la rete addestrata in regressione trattata in precedenza, andando però a "classificare" il risultato.

I valori reali presenti nel dataset e i valori della predizione sono stati assimilati alla propria categoria in tre diversi tentativi di classificazione: classificazione binaria con soglia a -75, classificazione binaria con soglia a -25, classificazione multiclasse con soglie a -25 e -75.

Per ciascun tentativo, riporto l'RMSE e l'accuratezza ottenuta in regressione, a titolo di confronto con il primo tentativo.

		RMSE	**Accuratezza**	**Errore**
Due classi (Soglia a -75)	Train	8.95		
	Test	10.24		
			99.28%	0.72%
Due classi (Soglia a -25)	Train	9.03		
	Test	10.28		
			90.5%	9.5%
Tre classi (Soglie a -75/-25)	Train	9.14		
	Test	10.38		
			89.94%	10.06%

A seguire riporto i valori di precisione per classi, ottenuti al termine dell'addestramento e della fase di test, misurata sulle tre classi:

	Train	**Test**
≥ -25 (R)	418954	191783
≥ -25 (P)	439467	206061
-75 <x< -25 (R)	48110	37189
-75< x <-25 (P)	29084	24344
≤- 75 (R)	2343	2001
≤- 75 (P)	916	568

R = Rilevato, P = Predetto

Dalla tabella si evince come la rete sia riuscita ad imparare a distinguere parzialmente le varie classi, non riuscendo ad eliminare del tutto l'effetto dello sbilanciamento dei valori nel dataset.

La classe *"≥ -25"* in particolare, sia nel training che nel test, tende ad assorbire valori in più dalle altre due classi, che sono invece state predette meno, in misura compatibile con la presenza delle rispettive classi all'interno del dataset.

La classe *"≤-75"*, quella più significativa per lo studio, è stata predetta correttamente soltanto al 39% nel training set, scendendo al 28% sul test set.

6.2.3 Classificazione "pura"

La terza possibilità è consistita nell'affrontare il caso come un problema di classificazione.

Per addestrare la rete ho modificato il dataset, trasformando ogni valore nella propria corrispondente categoria secondo le soglie descritte sopra.

Ai fini dello studio prestazionale della rete, questa è stata addestrata in tre modi diversi:

- Classificazione binaria con soglia a -25nT
- Classificazione binaria con soglia a -75nT
- Classificazione multiclasse con soglie a -25nT a -75nT

Metriche

Le funzioni di loss utilizzate sono *binary_crossentropy* per la classificazione binaria, *categorical_crossentropy* per la classificazione multiclasse.

Come riferimento, riporto i dettagli delle funzioni di loss di tipo crossentropy usate per la classificazione.

Per classificazioni binarie, la funzione di loss è definibile come segue:

$$-(y\log(p)+(1-y)\log(1-p))$$

Per classificazioni multiclasse invece, la funzione è la seguente:

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

6.2.3.1 Classificazione binaria (-25nT)

I risultati della classificazione binaria con soglia a -25nT sono i seguenti:

	Loss	**Accuratezza**
Train	0,13	81%
Test	0,71	73%

Il risultato di questa rete fanno intendere che la rete sia anche in questo caso ad imparare e a creare un modello piuttosto efficiente.

I risultati riguardanti l'accuratezza media per classe ci danno un'indicazione ulteriore delle predizioni svolte dalla rete.

	Train	**Test**
≥-25 (R)	423198	194700
≥- 25 (P)	428843	201163
≤- 25 (R)	46209	36273
≤- 25 (P)	40564	29810

La funzione di attivazione utilizzata nel layer Dense in uscita è "*sigmoid*".

Per quanto riguarda la precisione nella previsione dei valori al di sotto della soglia dei -25nT, la rete riesce a raggiungere l'87,7% sul training set e 82,1% sul test.

6.2.3.2 Classificazione binaria (-75nT)

Il risultato dell'accuratezza generale della rete va confrontato con il risultato dell'accuratezza media per classe.

	Loss	**Accuratezza**
Train	0,036	95,2%
Test	0,092	92%

	Train	Test
≥-75 (R)	467064	228972
≥- 75 (P)	467933	229650
≤- 75 (R)	2343	2001
≤- 75 (P)	1474	1323

La funzione di attivazione utilizzata nel layer Dense in uscita è "*sigmoid*".

La precisione nel classificare i valori inferiori alla soglia dei -75nT come tali è del 62,9% nella fase di train e 66,1% nel test.

6.2.3.3 Classificazione multiclasse (-25nT, -75nT)

A differenza delle due reti precedenti, la rete multiclasse ha avuto bisogno di un tipo di attivazione dello strato Dense "fully connected" di tipo Softmax, per evitasse che la funzione di loss desse come risultato NAN.

Di seguito sono riportati i dati relativi alla classificazione multiclasse.

	Loss	Accuratezza
Train	0,1193	96,4%
Test	0,1194	96,5%

	Train	Test
≥ -25 (R)	423198	194700
≥ -25 (P)	431051	204011
-75 <x< -25 (R)	43751	34198
-75< x <-25 (P)	36559	25187
≤- 75 (R)	2458	2075
≤- 75 (P)	1797	1775

La classificazione multiclasse raggiunge un'altezza molto alta sia sul train set che sul test set, riuscendo ad avere risultati migliori sulla classe *"≤- 75"* rispetto ai test in classificazione binaria, riuscendo infatti a riconoscere come significativi il 73% dei valori nel train e l'85% nel test.

7 MIGLIORAMENTI FUTURI

Durante lo svolgimento della tesi ho imparato a conoscere le reti neurali di tipo LSTM, le loro caratteristiche, le peculiarità e le differenze rispetto alle tradizionali reti RNN. Ho imparato inoltre ad implementarle e ad utilizzarle in Python.
Ho studiato, seppur in maniera superficiale, il fenomeno delle tempeste solari, il loro effetto sulla Terra e le tecniche di misurazione che sono state inventate durante il secolo scorso.

Credo che il lavoro svolto dimostri la possibilità concreta di utilizzare le reti neurali per analizzare l'indice SYM e prevederne l'andamento.

Ritengo che la difficoltà maggiore consista nell'utilizzare nel modo corretto i dati, modificandoli prima in modo da rendere più efficiente il processo di apprendimento della rete.

Ritengo che il pre-processing del dataset costituisca la parte più consistente in molte applicazioni di machine learning e di intelligenza artificiale.
In questo caso, la complessità del dataset risiedeva nel forte sbilanciamento presente nel dataset tra dati significativi e non.

Il campionamento

Per contrastare questo fenomeno, la tecnica che ho utilizzato è stata quella del sotto-campionamento del dataset, in modo da ridurre l'effetto della grande quantità di valori ritenuti non significativi.
Per fare ciò ho scelto, a seguito di alcuni tentativi, una risoluzione standard che ho mantenuto nelle varie implementazioni della rete LSTM.

Una delle prime modifiche che potrebbero essere fatte al mio lavoro è sicuramente quello di provare a modificare questa risoluzione, variando il range di campionatura.

Input della rete

A partire dai dati dell'indice, ho generato una matrice di vettori composti da 420 elementi. La rete impara dall'andamento dell'indice nella finestra temporale rappresentata da questi elementi. Anche in questo caso ho scelto questo numero di valori a seguito di alcuni tentativi e l'ho mantenuto nelle varie prove.

Una modifica che potrebbe essere interessante effettuare sarebbe modificare la dimensione dei vettori, andando ad osservare come cambino le prestazioni della rete aumentando o diminuendo la finestra temporale da dare in input alla rete.

Modifiche al dataset

Un'ultima modifica che potrebbe essere utile ai fini di una maggiore accuratezza di predizione è l'aggiunta di rumore al dataset, incrementando il numero di eventi significativi presenti nel dataset.

Nonostante le reti LSTM siano intrinsecamente pensate per ricordare gli eventi significativi anche a grande distanza l'uno dall'altro, ritengo che aumentarne il numero possa essere un buon modo per far sì che la rete impari meglio a riconoscere gli eventi "veri" in fase di test.

8 BIBLIOGRAFIA

[1] Mid latitude Geomagnetic Indices “ASY”and “SYM” for 2009 (Provisional) - T. Iyemori, M. Takeda, M. Nose, Y. Odagi and H. Toh

[2] Iaga Bulletin N°40 – M.Sugiura, T.Kamei

[3] Hourly Values of Equatorial DST For the IGY – Masahisa Sugiura

[4] High-resolution global storm index: Dst versus SYM-H - James A. Wanliss1 and Kristin M. Showalter1

[5] DeepZip: Lossless Compression using Recurrent Networks - Kedar Tatwawadi

[6] Deep Residual Learning for Image Recognition - Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun

[7] Geomagnetic Storms and Their Impacts on the U.S. Power Grid (Meta-R-319), John Kappenman, Metatech Corporation

[8] Evaluating Preprocessing Strategies for TimeSeries Prediction Using Deep Learning Architectures

LSTM - Sajitha Naduvil-Vadukootu, Rafal A. Angryk, Pete Riley