ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Scuola di Ingegneria e Architettura

Dipartimento di Informatica – Scienza e Ingegneria

Corso di Laurea Magistrale in INGEGNERIA INFORMATICA

Tesi di Laurea in SISTEMI INTELLIGENTI

Analisi di serie temporali riguardanti dati energetici mediante architetture neurali profonde

RELATORE
Chiar.ma Prof.ssa Michela Milano

LAUREANDO
Luca Di Ielsi

CORRELATORI
Dott. Marco Lippi
Ing. Piergiorgio Testi

Anno Accademico 2014/2015 – Sessione III

Indice

Elenco delle figure

Elenco delle tabelle

INTRODUZIONE

Lo scenario nel quale è stato sviluppato il presente lavoro di tesi è quello della gestione dell'energia elettrica.

In Italia, negli ultimi anni sono sorte molte società il cui obiettivo principale è proprio quello di fornire diagnosi, certificazioni, servizi energetici o misure di miglioramento dell'efficienza energetica, andando ad agire direttamente nelle installazioni o nei locali del cliente attraverso interventi di riqualificazione edilizia (ad esempio sostituendo i materiali con altri a minor impatto energetico) o di processo (ad esempio riorganizzando i turni di produzione in orari in cui l'energia costi meno). Nel 2008 sono state riconosciute tramite il Decreto Legislativo 115/2008 le ESCo (Energy Service Company), cioè società che forniscono ai propri clienti (i quali hanno, generalmente ma non necessariamente, un consumo di energia significativo) un insieme di servizi integrati per la realizzazione e la gestione di interventi mirati ad ottenere un risparmio energetico, garantendone i risultati ed i risparmi promessi. Tra le varie fasi in cui questi interventi si articolano, si riconosce anche quella di monitoraggio continuo degli impianti e verifica delle prestazioni e risultati ottenuti. Il monitoraggio viene effettuato tramite la dislocazione, direttamente nei luoghi interessati, di sensori per la misurazione di alcune metriche inerenti al consumo o alla produzione di energia. I dati rilevati dai sensori sono inviati con dei particolari protocolli ad un sistema informatico gestito dalle società, che oltre a conservarli, creando di fatto uno storico dell'evoluzione dell'impianto monitorato, li esaminano cercando di estrarne informazioni utili.

Seaside è una di queste società, che ha sede a Bologna ed è nata nel 2010 dall'unione di realtà consolidate nel settore della sustainability, dell'implementazione di impianti e della finanza, integrando aspetti di ingegneria e di economia. L'esigenza da parte sua di migliorare il proprio reparto di *Business Intelligence* – in particolar modo rispetto all'analisi e alla predizione dell'andamento di serie temporali – in aggiunta al suo spiccato orientamento verso l'innovazione tecnologica e al mio interesse per determinati strumenti nell'ambito del *Machine Learning* pertinenti al caso in questione, hanno suggerito una collaborazione per applicare a dati industriali in loro possesso nuove tecniche di elaborazione, sulla base dello stato dell'arte.

Con il presente lavoro, si è cercato di studiare l'apporto che può fornire l'applicazione di un'architettura neurale profonda ai dati utilizzati per effettuare la previsione di una serie temporale.

Dopo aver esaminato quali fossero i dati a disposizione e, in collaborazione con Seaside, aver definito un obiettivo che prevedeva la predizione della produzione di energia elettrica da un impianto fotovoltaico con un orizzonte temporale di un'ora, è stata effettuata una rassegna di quale sia lo stato dell'arte relativo ad alcune delle tecniche neurali profonde attualmente di maggior interesse. Si è proceduto quindi all'implementazione di una di queste, il *denoising auto-encoder* a più livelli, sia per come questo si possa adattare al problema della previsione di serie temporali (dove la cardinalità dei dati è, generalmente, molto elevata), sia a scopo di ricerca, in quanto rappresenta un tipo di rete neurale le cui prestazioni rispetto all'applicazione in ambito predittivo sono ancora scarsamente studiate. Il lavoro ha visto la sua conclusione con una parte di test sperimentali in cui si sono effettuati dei confronti tra previsioni con o senza l'impiego della sottostruttura creata, variando diverse configurazioni.

Tali esperimenti hanno fornito risultati apprezzabili già con l'utilizzo di un solo livello, e sono stati migliorati aggiungendo profondità all'architettura tramite l'aggiunta di un secondo livello.

Sebbene i risultati ottenuti ad oggi siano più che soddisfacenti, il lavoro svolto durante questa tesi può essere esteso in molte direzioni, con l'obiettivo di migliorare la qualità delle predizioni e di estendere il modello allo studio di altri casi simili.

1. ANALISI DI SERIE TEMPORALI

1.1 Serie temporali

Una serie temporale (o storica) consiste nell'insieme di un certo numero di osservazioni riguardanti un determinato fenomeno, ordinate cronologicamente. Usualmente tali osservazioni sono misurate ad intervalli di tempo costanti, ad esempio con cadenza oraria, giornaliera, settimanale, mensile, ecc., a seconda della natura del fenomeno oggetto di studio; tuttavia questo non è necessario.

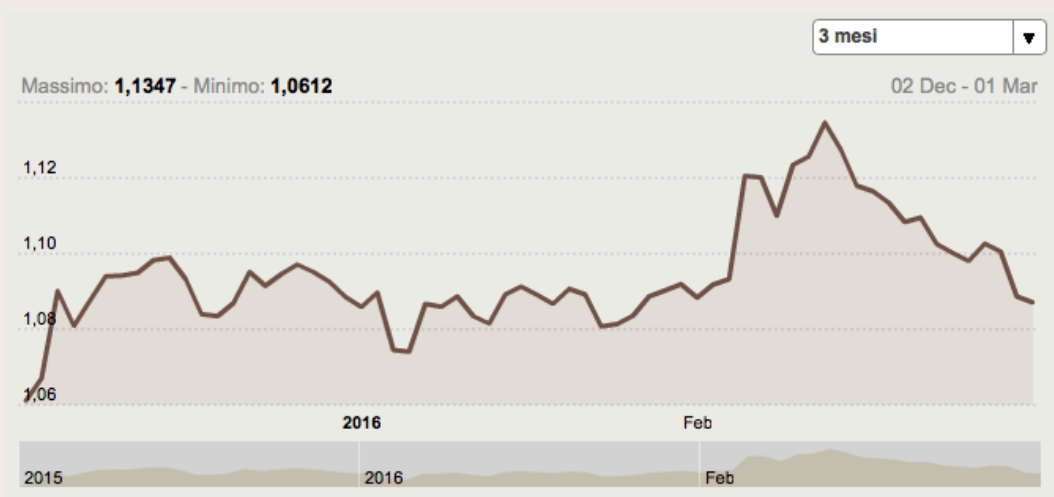

Figura 1.1 - Esempio di una serie temporale: andamento euro/dollaro durante l'ultimo trimestre (fonte http://www.ilsole24ore.com/finanza-e-mercati.shtml)

In una serie temporale, ogni valore può rappresentare una quantità rilevata in un istante di tempo, ad esempio la temperatura atmosferica esterna misurata ogni quarto d'ora, il prezzo di un'azione registrato settimanalmente alla chiusura della Borsa, oppure può derivare

dall'accumulo di quantità rilevate su un intervallo temporale, ad esempio il consumo mensile di energia elettrica di uno stabilimento industriale, il totale delle vendite di un esercizio commerciale durante lo scorso trimestre. Alcuni tipici esempi di serie temporale potrebbero essere i dati registrati quotidianamente relativi al cambio euro/dollaro, o quelli relativi ad un particolare indicatore socioeconomico, come il tasso di disoccupazione nazionale, verificato con frequenza trimestrale, o ancora l'attività del traffico Internet in entrata e in uscita, monitorato all'interno di una rete aziendale.

Dal punto di vista statistico, le osservazioni che formano una serie temporale sono tra loro dipendenti, ed è proprio la natura di questa dipendenza a costituire l'oggetto di interesse. Gli n valori osservati sono realizzazioni di n variabili aleatorie $X_1, \dots, X_n$ fra loro dipendenti. L'obiettivo dell'analisi di una serie temporale è quello di studiare le relazioni tra tali variabili casuali attraverso la costruzione e lo sviluppo di modelli che mirino o all'ottenimento di previsioni di breve periodo o alla scomposizione della serie temporale in un insieme di componenti latenti.

Il presente lavoro di tesi ha concentrato l'attenzione sul primo scopo piuttosto che sul secondo. Al giorno d'oggi, infatti, l'analisi di una serie temporale a scopo predittivo riveste un ruolo di primaria importanza in molteplici ambiti nei quali è ragionevole pensare che siano disponibili grandi moli di dati:

- Operations management. Le aziende abitualmente utilizzano le previsioni delle vendite di prodotti o della domanda di servizi al fine di programmare la produzione, controllare le scorte, gestire il processo di distribuzione, determinare il fabbisogno di personale e pianificare la capacità. Le previsioni possono anche essere impiegate per determinare il mix dei prodotti o dei servizi da offrire e i posti in cui risulti più conveniente la produzione.

- Marketing. Effettuare previsioni è importante in molte decisioni di marketing. Le previsioni di come le vendite rispondano ad una campagna pubblicitaria, al lancio di nuove offerte o ai cambiamenti di prezzo permettono alle aziende di valutarne l'efficacia, di determinare se gli obiettivi siano stati raggiunti ed, eventualmente, di apportare aggiustamenti alla strategia.
- Finanza e gestione del rischio. Chi investe in attività finanziarie è interessato a prevedere il rendimento dei suoi investimenti. Tali attività includono – ma non sono limitate a – azioni, obbligazioni e materie prime; altre decisioni di investimento possono essere fatte in relazione alle previsioni dei tassi di interesse, delle opzioni e dei tassi di cambio. La gestione del rischio finanziario richiede previsioni sulla mutevolezza dei rendimenti delle attività, così da poter valutare ed assicurare i rischi associati al portafoglio di investimenti e cosicché i derivati finanziari possano essere correttamente prezzati.
- Economia. I governi, le istituzioni finanziarie e le organizzazioni politiche richiedono previsioni delle principali variabili economiche, come ad esempio il tasso del prodotto interno lordo, della crescita demografica, della disoccupazione, della produzione e del consumo. Queste previsioni sono una parte integrante dell'orientamento che sta alle spalle della politica monetaria ed economica, dei piani di bilancio e delle decisioni assunte dei governi.
- Controllo dei processi industriali. Le previsioni dei valori futuri di caratteristiche critiche per un processo di produzione possono aiutare a capire quando variabili controllabili importanti dovrebbero essere cambiate, o se il processo dovrebbe essere arrestato e revisionato. Schemi di controllo *feedback* o *feedforward* sono largamente utilizzati nel monitoraggio e nella regolazione dei processi industriali, e la previsione dell'output del processo è una parte integrante di tali schemi.

- Demografia. Le previsioni sulla popolazione di un Paese o di una regione sono effettuate abitualmente, spesso suddivise per variabili come il sesso, l'età e l'etnia. I demografi predicono anche l'andamento delle nascite, delle morti e dei modelli di migrazione delle popolazioni. I governi successivamente adoperano queste previsioni per pianificare azioni politiche e sociali, come la spesa nella sanità pubblica, i programmi di pensionamento e quelli anti-povertà. Molte aziende utilizzano le previsioni sulla popolazione suddivisa per fasce d'età per redigere piani strategici riguardanti lo sviluppo di nuove linee di prodotti o tipologie di servizi da offrire.

1.2 Processo di analisi predittiva

Come per la gran parte delle attività nel campo scientifico, anche per l'analisi predittiva è possibile formalizzare un processo di lavoro, cioè un insieme di attività interconnesse che da uno o più dati in ingresso producano uno o più dati in uscita.

Si individuano principalmente 7 passi:

1. Definizione del problema
2. Raccolta dei dati
3. Analisi dei dati
4. Scelta del modello
5. Validazione del modello
6. Allestimento del modello di predizione
7. Monitoraggio delle prestazioni

La definizione del problema è forse la fase più delicata e complessa dell'intera attività. Richiede di comprendere le modalità in cui le previsioni verranno utilizzate e le aspettative di chi ne farà uso. Ciò che ci si deve chiedere durante questa fase è quale sia la forma desiderata della previsione e il suo orizzonte temporale (ad esempio, se è richiesta su un intervallo settimanale, o mensile, ecc.), quanto spesso la previsione debba essere rivista e quale sia il livello minimo di precisione necessario affinché si possano considerare accettabili le scelte effettuate sulla base di tale previsione. È fondamentale discutere in maniera approfondita con i vari soggetti coinvolti nella raccolta dei dati, nella gestione del database e nell'utilizzo del modello per la pianificazione per non tralasciare alcun aspetto e comprendere le caratteristiche produttive dell'impresa, dei vari processi interni e della domanda servita.

Durante la raccolta dei dati si ottengono le informazioni riguardanti la storia delle variabili da predire, incluse quelle sui possibili predittori. In questo momento è utile iniziare a pensare a come saranno gestiti in futuro i dati che si stanno raccogliendo e a come saranno risolti eventuali problemi di dati mancanti o *outlier*.

L'analisi dei dati è la fase preliminare alla selezione del modello da usare. A questo punto potrebbe essere utile produrre dei grafici dei dati precedentemente raccolti ed effettuare una prima semplice ispezione visiva, che spesso permette di individuare immediatamente eventuali *trend*, stagionalità o altre componenti cicliche. Il *trend* rappresenta la tendenza di fondo del fenomeno considerato, che può essere crescente o decrescente, e di solito può essere espresso mediante una funzione polinomiale di grado relativamente basso. La stagionalità è costituita da variazioni che si riscontrano con analoga intensità all'interno di uguali periodi, come ad esempio lo stesso anno. Oltre a queste caratteristiche si dovrebbero identificare valori insoliti e potenziali *outlier*, e approfondirne lo studio per correggere eventuali errori di misura.

Si arriva dunque alla scelta di uno o più modelli di predizione e, conseguentemente, alla loro configurazione. La configurazione consiste nel calcolare, per ciascun modello scelto, i suoi parametri ignoti, solitamente con il metodo dei minimi quadrati, in modo che sia minimizzato l'errore di predizione.

Implementati i modelli è necessaria la loro validazione, cioè la valutazione delle prestazioni. Tale valutazione deve andare oltre il semplice studio di come questi si adattino ai dati storici da cui sono stati ricavati, ma dovrebbe essere applicata a dati nuovi. Per fare ciò, un metodo largamente impiegato suggerisce di dividere il *data set* iniziale in due, un *training set* ed un *test set*. I modelli vengono allenati sul solo *training set*, e da questo vengono ricavati i parametri che sono poi testati sul *test set*, un insieme di dati che è stato tenuto separato e che il sistema non ha mai visto. Di norma, per quanto riguarda il *Machine Learning*, la suddivisione del *data set* iniziale prevede anche di ricavare un *validation set* impiegato per determinare il numero ottimo di epoche durante l'apprendimento della rete, in modo da evitare l'*overfitting*, cioè la tendenza del modello ad imparare in modo troppo specifico, senza la capacità di generalizzazione sufficiente per adattarsi con successo anche ai nuovi casi non visionati.

L'allestimento del modello "vincitore" consiste nel mettere a disposizione dell'utilizzatore finale il sistema di predizione. È importante che costui comprenda come utilizzare lo strumento proposto e che la generazione delle previsioni diventi un'operazione il più abituale possibile. La manutenzione del sistema, così come la sicurezza che i dati in ingresso continuino ad arrivare con la frequenza prevista sono due aspetti che impattano in modo incisivo sulla puntualità e sulla bontà delle previsioni.

È naturale, nel campo delle previsioni, che le condizioni possano cambiare nel tempo. Così, un modello che poteva essere performante in

passato può deteriorare le proprie prestazioni e non adattarsi più alla situazione presente. Solitamente, in queste occasioni, gli errori di predizione diventano via via maggiori, oppure cresce la loro frequenza. Per questo motivo, monitorare la performance del sistema dovrebbe essere un'attività continuativa fin dal principio del ciclo di vita.

1.3 Modelli per l'analisi predittiva

Ognuno degli scenari descritti precedentemente si differenzia dagli altri per via dell'orizzonte temporale su cui si va ad effettuare la previsione, per i risultati desiderati su di essa e, soprattutto, per le caratteristiche dei dati disponibili. Si individuano sostanzialmente due grandi tipologie di previsioni: quantitative e qualitative.

Nella prima categoria rientrano quelle in cui l'informazione riguardante il passato è disponibile, può essere rappresentata sotto forma di dato numerico ed è presumibile che alcune delle caratteristiche della struttura dei dati si possano ravvisare anche nei periodi successivi a quello di osservazione. Quest'ultima condizione è una caratteristica fondamentale di tutte le tecniche quantitative (e anche di una parte di quelle qualitative) indipendentemente dal tipo e dalla complessità del modello utilizzato. I modelli che fanno riferimento a questa classe sintetizzano dai dati gli schemi ricorrenti ed esprimono una dipendenza statistica tra i valori passati e quello corrente. Dopodiché tali modelli sono usati per effettuare una previsione, proiettando nel futuro gli schemi individuati in precedenza. I tre modelli di questo tipo più utilizzati sono modelli di regressione, modelli di *smoothing* e modelli generali per serie temporali.

La seconda categoria è invece caratterizzata dal fatto che i dati in ingresso richiesti dipendono dal metodo utilizzato ed in gran parte da attività legate fortemente al giudizio dell'analista e alla conoscenza accumulata. Le previsioni qualitative sono spesso usate in situazioni dove vi è poca disponibilità di dati sul passato, o addirittura non ve ne è affatto. Un esempio potrebbe essere l'introduzione di un nuovo prodotto per il quale non esiste uno storico a riguardo. In questo caso l'azienda potrebbe sfruttare l'opinione esperta dei propri addetti al marketing per stimare (in modo soggettivo) come andranno le vendite del nuovo prodotto durante la sua fase di introduzione nel mercato. Rientrano in questa categoria un'ampia gamma di metodologie che si differenziano per costi, complessità ed accuratezza, e che, quando e nella misura in cui risulti possibile, sono affiancate a quelle quantitative. Probabilmente la tecnica di previsione qualitativa più formale e più largamente diffusa è il metodo d'indagine iterativo conosciuto come *Metodo Delphi*, che si svolge attraverso più fasi di espressione e valutazione delle opinioni di un gruppo di esperti o attori sociali ed ha l'obiettivo di far convergere l'opinione più completa e condivisa verso un'unica espressione.

Segue un approfondimento sui due modelli che sono stati utilizzati per la previsione durante il lavoro di tesi. La scelta è ricaduta sulla regressione lineare multipla, prevalentemente sulla base della possibilità di ottimizzarne, per mezzo di una rete neurale, i dati in ingresso. I risultati del *random walk*, invece, sono stati utilizzati come metro di riferimento per il confronto.

1.3.1 Random walk

In generale, una serie temporale può essere descritta come la traiettoria di un processo stocastico. Semplificando molto, si può definire un

processo stocastico come una forma di rappresentazione di una grandezza che varia nel tempo in maniera casuale (come ad esempio un segnale elettrico, o il numero di persone in coda allo sportello postale) e con determinate caratteristiche. Tra i processi stocastici coinvolti nello studio delle serie temporali, una classe molto interessante è quella dei processi integrati. Un processo stocastico $\{y_t\}$ si dice integrato di ordine 1 se y_t non è stazionario, mentre risulta stazionaria (cioè con media e varianza costante) la differenza $\Delta y = y_t - y_{t-1}$.

Il processo integrato più semplice è il processo che prende il nome di *random walk plus drift*, che dato un valore iniziale y_0 è generato da

$$y_t = y_{t-1} + \delta + \varepsilon_t,$$

dove ε_t è una sequenza di variabili indipendenti e identicamente distribuite (*i.i.d.*) a media nulla e varianza σ_ε^2 (*white noise*), e δ è una costante, detta deriva (*drift*). Se tale costante è nulla il processo è detto semplicemente *random walk*, se $\delta > 0$ il processo ha una tendenza a crescere, se viceversa $\delta < 0$ il processo tende a decrescere. Tale processo è integrato di ordine 1, infatti se si esprime la differenza

$$y_t - y_{t-1} = \delta + \varepsilon_t$$

si ottiene la variabile casuale ε_t (più, eventualmente, il *drift* δ) che, essendo una sequenza *i.i.d.*, è un processo stazionario. Infine, avendo ε_t media nulla, il valore atteso di y_t è il primo valore della sequenza y_0, mentre la varianza è pari a t volte la varianza di ε_t.

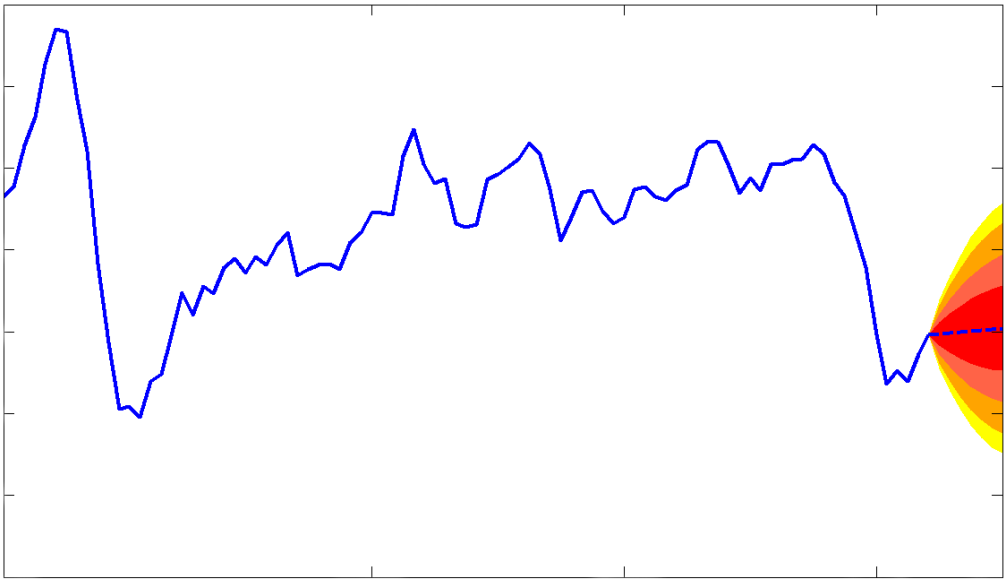

Figura 1.2 - Esempio di random walk per l'analisi predittiva

Per quanto riguarda gli esperimenti utilizzati durante questo lavoro di tesi, è stato considerato come modello di confronto il processo *random walk* nella sua versione più semplice, cioè senza deriva e con ε_t nullo, ossia predicendo il valore di y_t all'istante t come identico di quello all'istante precedente y_{t-1}.

1.3.2 Regressione lineare

Il modello di regressione ha lo scopo di stimare i valori di una variabile quantitativa a partire da quelli osservati di una o più altre variabili, o da quelli che la variabile stessa ha assunto in istanti passati. Nell'analisi di regressione la variabile i cui valori sono da stimare viene detta variabile dipendente (o variabile risposta), mentre quella da cui questi valori in certa misura dipendono viene chiamata variabile indipendente (o regressore, o variabile esplicativa). Un modello nel quale si faccia ricorso a più di una variabile esplicativa per effettuare previsioni su una

variabile dipendente prende il nome di modello di regressione multipla; viceversa, se la variabile esplicativa è unica, si parla di modello di regressione semplice.

Oltre ad ottenere valori numerici per la variabile dipendente, il modello di regressione permette altresì di identificare il tipo di relazione matematica che intercorre tra la variabile indipendente e la variabile dipendente. La natura di tale relazione può essere rappresentata da funzioni matematiche di svariato tipo, da alcune semplici fino ad altre estremamente complesse.

La tipologia di relazioni più semplice è quella lineare, che per l'appunto viene descritta da una funzione lineare e che, nel caso generale di p variabili esplicative, assume l'espressione

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \varepsilon_i,$$

dove:

β_0 = intercetta
β_1 = inclinazione di y rispetto alla variabile x_1 tenendo costanti le variabili $x_2, x_3, \ldots, x_p$
β_2 = inclinazione di y rispetto alla variabile x_2 tenendo costanti le variabili $x_1, x_3, \ldots, x_p$
β_p = inclinazione di y rispetto alla variabile x_p tenendo costanti le variabili $x_1, x_2, \ldots, x_{p-1}$
ε_i = errore nel valore della y per la i-esima osservazione
y_i = valore osservato della variabile dipendente per la i-esima osservazione
x_i = valore osservato della variabile indipendente per la i-esima osservazione

La componente ε_i è detta anche residuo del modello ed è il termine che trasforma il modello di regressione da modello matematico a statistico-probabilistico.

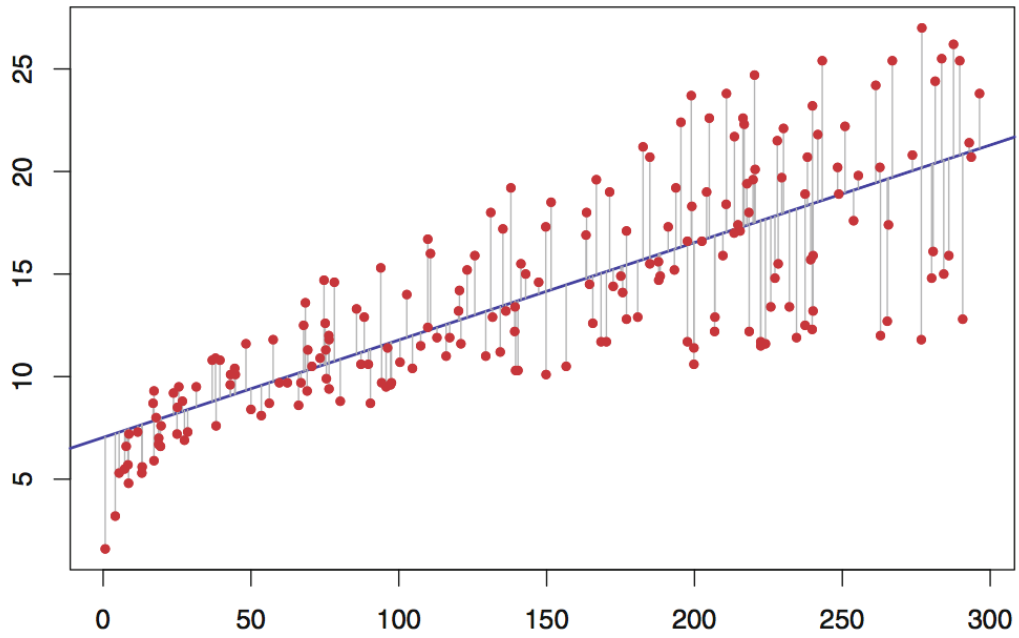

Figura 1.3 - Esempio di una regressione lineare semplice con una sola variabile predittrice: in blu è evidenziata la retta di regressione, in grigio i residui

Chiaramente, l'individuazione del modello matematico più adatto a descrivere la relazione tra le p variabili x_i e y è influenzata dalla distribuzione congiunta delle variabili stesse, la quale si può riconoscere ed analizzare attraverso un diagramma di dispersione o per mezzo di strumenti più sofisticati e oggettivi, come ad esempio il metodo dei minimi quadrati. Tale metodo consiste nel minimizzare la somma dei quadrati degli scarti tra i valori osservati di y (y_i) e quelli stimati $(\tilde{y}_i)$ dalla funzione di regressione, di equazione

$$\tilde{y}_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + \cdots + b_p x_{pi},$$

dove $b_0, b_1, b_2, \ldots, b_p$ sono i coefficienti di regressione campionari e vengono usati come stimatori dei corrispondenti parametri della popolazione $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$.

La somma da minimizzare è esprimibile come

$$\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

e, sostituendo $\tilde{y}_i$, si ha

$$\sum_{i=1}^{n} [y_i - (b_0 + b_1 x_{1i} + b_2 x_{2i} + \cdots + b_p x_{pi})]^2.$$

Il metodo dei minimi quadrati consiste esattamente nel determinare quei valori dei parametri $b_0, b_1, b_2, \ldots, b_p$ tali da minimizzare la somma dei quadrati degli scarti dei valori osservati da quelli stimati dalla funzione di regressione. Si può dimostrare che qualunque altro valore per i coefficienti, diverso da quello ottenuto con il metodo dei minimi quadrati, porta ad ottenere un valore della somma dei quadrati degli scarti certamente maggiore.

Dopo aver determinato i parametri del modello, al fine di valutare quanto il modello si adatti ai dati, è necessario calcolare tre importanti indici di variabilità. Il primo è la devianza totale (o somma complessiva degli scarti), denotata con SST (*Total Sum of Squares*), che è una misura della variabilità complessiva delle y_i rispetto alla media generale $\bar{y}$:

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

La devianza totale si suddivide in devianza spiegata (o devianza di regressione) e devianza residua (o devianza dalla regressione o, semplicemente, residuo). La devianza spiegata o devianza di regressione (SSR, *Residual Sum of Squares*) è quella parte di variabilità dovuta alla presenza di una relazione tra le variabili x e y:

$$SSR = \sum_{i=1}^{n} (\tilde{y}_i - \bar{y})^2,$$

mentre la devianza residua o devianza dalla regressione (SSE, *Explained Sum of Squares*) è dovuta ad altri fattori che non siano relativi alla relazione tra x e y, ed è la parte erratica del modello:

$$SSE = \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

La devianza totale è uguale alla devianza di regressione a cui si deve sommare la devianza residua:

$$SST = SSR + SSE.$$

A questo punto, per valutare la bontà di adattamento del modello costruito ai dati forniti, solitamente si calcola un altro indicatore, definito come il rapporto tra la devianza di regressione (SSR) e la devianza totale (SST), che misura la proporzione di variabilità della y spiegata dalla relazione con le variabili x all'interno del modello di regressione. Questo rapporto è un numero compreso tra 0 e 1, ed è detto coefficiente di determinazione R^2:

$$R^2 = \frac{devianza\ di\ regressione}{devianza\ totale} = \frac{SSR}{SST}.$$

Più R^2 è vicino a 1, più si può affermare che il modello spieghi bene i dati. Ad esempio, un valore di $R^2 = 0{,}9237$ significa che il 92,37% della variabilità della y è spiegato dalle x ed un valore così elevato suggerisce una significativa relazione tra le variabili scelte, lasciando intuire che solo il 7,36% della variabilità contenuta nei dati campionari è dovuto ad altri fattori che non dipendono da tale relazione. Nel caso di regressione lineare multipla, inoltre, si può utilizzare un nuovo indice che tenga conto anche del numero di variabili esplicative incluse nel modello e dell'ampiezza del campione, l'$R^2\ corretto$:

$$R^2_{adj} = 1 - \left[(1 - R^2)\frac{n-1}{n-p-1}\right].$$

dove n è la dimensione dello spazio campionario e p il numero delle variabili esplicative incluse nel modello. Il ricorso a questo tipo di indice si rende necessario soprattutto qualora si vogliano confrontare modelli di regressione che intendono spiegare la stessa variabile dipendente impiegando un numero diverso di variabili esplicative. In modo del tutto identico ad R^2, il significato di R^2_{adj} è la percentuale della variabilità della y che può essere spiegata dal modello proposto, considerato il numero di predittori e la dimensione dell'ampiezza campionaria.

2. DEEP LEARNING

2.1 Reti neurali artificiali

Le reti neurali artificiali sono sistemi di elaborazione dell'informazione che simulano, tramite un modello matematico, la struttura ed il comportamento del cervello umano al fine di riprodurne il meccanismo con cui questo svolge le attività di ragionamento e apprendimento.

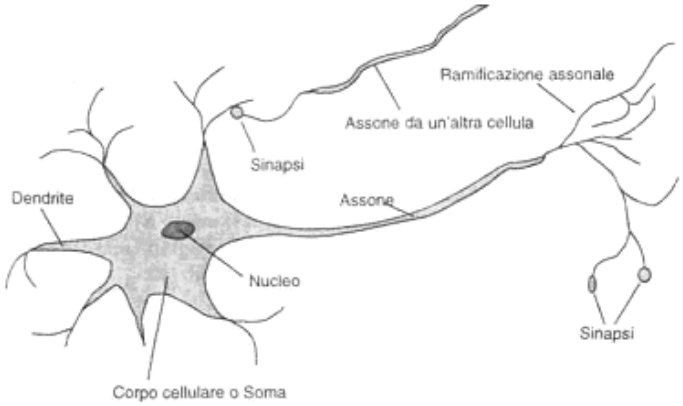

Figura 2.1 - Il neurone biologico

Il nostro sistema nervoso è composto da miliardi di neuroni, o cellule nervose. Un neurone è formato da un corpo cellulare (soma), da diversi prolungamenti ramificati, detti dendriti, attraverso cui riceve segnali elettrici da altri neuroni afferenti e da un prolungamento filamentoso chiamato assone, la cui lunghezza varia da circa un centimetro a qualche metro. Alla propria estremità, l'assone si ramifica formando terminali attraverso i quali i segnali elettrici vengono trasmessi ad altre cellule (ad esempio ai dendriti di altri neuroni). Tali segnali superano il

punto di congiunzione tra il terminale di un assone e la cellula ricevente (sinapsi) grazie a sostanze chimiche, dette neurotrasmettitori. Un neurone si attiva, cioè trasmette un impulso elettrico lungo il relativo assone, quando si verifica una differenza di potenziale elettrico tra l'interno e l'esterno della cellula. L'impulso elettrico provoca la liberazione di un neurotrasmettitore dai terminali dell'assone, che a loro volta possono, ad esempio, influenzare altri neuroni.

Pur avendo alla base un meccanismo piuttosto semplice come quello appena descritto, il cervello umano si rivela un calcolatore complesso, non lineare e parallelo, in grado di eseguire attività articolate come il riconoscimento, la percezione e il controllo del movimento, in modo spesso più rapido degli attuali calcolatori (che tuttavia utilizzano componenti elettronici molto più veloci dei neuroni biologici). Oltre a questo, il cervello è privo di un controllo centralizzato perché tutte le sue zone funzionano insieme, influenzandosi reciprocamente e apportando un certo contributo alla realizzazione di uno specifico compito. Se un neurone o una delle sue connessioni si danneggiano, le prestazioni del processo cerebrale degradano gradualmente, ma l'attività non viene meno. L'aspetto decisamente più interessante è rappresentato dal fatto che il cervello è in grado di modificare continuamente le connessioni tra neuroni in base all'esperienza acquisita: è capace, cioè, di imparare.

Una rete neurale artificiale tenta di replicare il comportamento di questa macchina potentissima creando un gran numero di connessioni tra elementi molto semplici, i neuroni artificiali, e realizzando di fatto una struttura distribuita caratterizzata da un elevato grado di parallelismo e, soprattutto, in grado di apprendere (e generalizzare), cioè fornire una risposta in corrispondenza di una situazione di partenza mai vista prima.

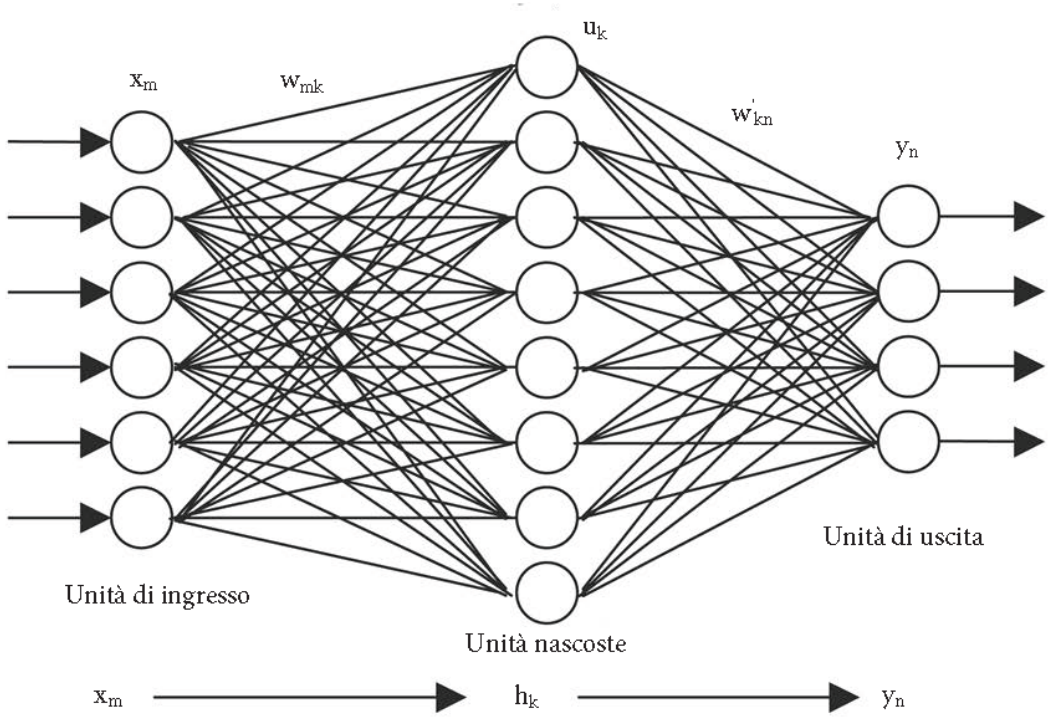

Figura 2.2 - Schema di una rete neurale artificiale

Esattamente come nel cervello umano, ogni neurone artificiale è connesso con altri neuroni e il comportamento intelligente emerge proprio dalle numerosissime interazioni tra le unità interconnesse. Alcune di queste unità ricevono informazioni dall'esterno, altre emettono risposte verso l'esterno e altre ancora comunicano solo ed esclusivamente con le unità all'interno della rete. Si individuano così tre livelli: quello delle unità di ingresso (*input*), di uscita (*output*) e nascoste (*hidden*). Ciascun neurone svolge un'unica operazione elementare: si attiva se la quantità totale di segnale che riceve supera una certa soglia di attivazione e, una volta attivo, emette a sua volta un segnale che viene trasmesso lungo i canali di comunicazione fino alle altre unità a cui è connesso. Ogni singolo segnale proveniente dai dendriti e diretto verso il nucleo viene pesato in base alla sua importanza e l'apporto che fornisce all'attivazione del neurone a cui è destinato è relazionato al suo peso.

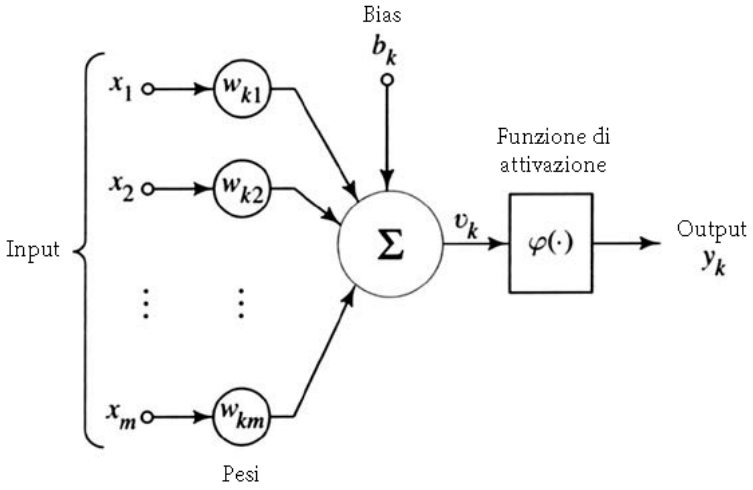

Figura 2.3 - Il neurone artificiale

La funzione di attivazione è una funzione matematica che può assumere varie forme, le più utilizzate sono la funzione gradino, la funzione segno, la funzione logistica e la funzione tangente iperbolica. Il legame tra l'input e l'output, cioè la funzione di trasferimento della rete, non viene programmato bensì risulta semplicemente da un processo di apprendimento basato su dati empirici, di cui se ne distinguono sostanzialmente tre tipi:

- Supervisionato (*supervised learning*), qualora si disponga di un insieme di dati per l'addestramento (o *training set*) comprendente esempi tipici di ingressi con le relative uscite ad essi associate: in questo modo la rete può imparare ad inferire la relazione che li lega. Successivamente la rete è addestrata mediante un opportuno algoritmo (di solito il *backpropagation algorithm*), il quale usa tali dati allo scopo di modificare i pesi ed altri parametri della rete stessa in modo da minimizzare l'errore di previsione relativo all'insieme di addestramento; i risultati sono testati su un insieme di dati che la rete non ha mai

analizzato, detto test set. Se l'addestramento ha successo, la rete impara a cogliere la relazione incognita che lega le variabili d'ingresso a quelle d'uscita, ed è quindi in grado di fare previsioni anche quando l'uscita non sia nota a priori. Per consentire ciò, è necessario che il sistema possieda un'adeguata capacità di generalizzazione in riferimento a casi ignoti e mai visti. Questo procedimento è utile nell'affrontare problemi di regressione o classificazione.

- Non supervisionato (*unsupervised learning*), basato su algoritmi d'addestramento che modificano i pesi della rete facendo esclusivamente riferimento ad un insieme di dati che comprenda solo le variabili d'ingresso. Tali algoritmi tentano di raggruppare i dati d'ingresso e di individuare quindi determinati cluster rappresentativi dei dati stessi, facendo uso tipicamente di metodi topologici o probabilistici. L'apprendimento non supervisionato è anche impiegato per sviluppare tecniche di compressione dei dati.

- Per rinforzo (*reinforcement learning*), in cui un opportuno algoritmo si prefigge lo scopo di individuare un modus operandi valido, a partire da un processo di osservazione dell'ambiente esterno; ogni azione ha un impatto sull'ambiente e, per contro, l'ambiente produce una retroazione che guida l'algoritmo stesso durante processo di apprendimento. Tale classe di problemi prevede l'impiego di un agente, dotato di capacità percettiva, che esplori un ambiente in cui intraprende una serie di azioni, e che via via modifichi tali azioni in base alle conseguenze prodotte. L'ambiente stesso fornisce in risposta un incentivo o un disincentivo, a seconda dei casi. Gli algoritmi per il *reinforcement learning* tentano, quindi, di determinare una politica volta a massimizzare la somma degli incentivi ricevuti dall'agente nel corso della sua esplorazione del problema.

> L'apprendimento con rinforzo si discosta da quello supervisionato perché non fa mai riferimento a coppie input-output di esempi noti, né procede alla correzione esplicita di azioni che non siano ottime. Inoltre, l'algoritmo è focalizzato sulla prestazione in linea, la quale implica un bilanciamento tra esplorazione di situazioni ignote e sfruttamento della conoscenza attuale.

Da quando fu teorizzato il primo modello di neurone artificiale (nel 1943 da McCulloch e Pitts), e dopo un periodo iniziale di grande entusiasmo attorno all'argomento, la fortuna delle reti neurali declinò nel decennio a cavallo tra il 1970 e il 1980, anche per via di alcuni limiti messi in luce dagli studi di Minsky e Papert. Fino al 1982, quando dapprima Hopfield e Kohonen, e successivamente Rumelhard, Hinton e Williams proposero nuovi modelli e riportarono in auge questa tecnologia. Una decina di anni fa alcuni ricercatori tra cui Hinton, Bengio e LeCun, hanno proposto un paradigma di apprendimento automatico innovativo, che prende il nome di *Deep Learning* e che ha conseguito risultati inimmaginabili fino a pochi anni prima in svariate applicazioni affrontabili con le tecniche di *Machine Learning*, contribuendo ad enfatizzarne l'utilizzo. Attualmente la comunità scientifica mondiale considera le reti neurali (nelle loro molteplici implementazioni) una tecnologia viva e di interesse certo, per via del loro eclettismo che permette una trasversalità rispetto ai diversi campi d'utilizzo e anche per le prestazioni che si riescono ad ottenere sfruttandole nella risoluzione di determinati problemi. Gli ambiti dove sono maggiormente utilizzate sono il riconoscimento di forme, la comprensione del linguaggio naturale, la traduzione automatica, i problemi di classificazione e di previsione, il *Data Mining*, l'ottimizzazione di risultati ed il *pattern recognition* in campo bioinformatico.

Tra i molti aspetti decisamente positivi dell'impiego delle reti neurali, ve ne sono anche di meno favorevoli. Il più evidente è che tutti i modelli prodotti da reti neurali, nonostante siano molto efficaci, non sono spiegabili in linguaggio simbolico: i risultati vanno dunque accettati così come sono, come se derivassero da una *black box*. Rimangono, quindi, uno strumento da utilizzare se l'attenzione si pone sul fine, piuttosto che sul mezzo. Inoltre, come per qualsiasi oggetto di modellazione, anche le reti neurali fanno dipendere la loro efficienza dalla cura con cui si scelgono le variabili predittive: ad esempio non sono in grado di trattare variabili di tipo categorico (come i nomi di persona) con molti valori differenti. È possibile che la fase di addestramento del sistema, durante la determinazione dei pesi dei singoli neuroni, richieda molto tempo, a seconda della cardinalità dei dati implicati. Infine, non esistono (ancora) teoremi o formalismi che permettano di definire i parametri ottimi per la rete: la bontà del modello dipende spesso dall'esperienza di chi l'ha creato e dal numero di prove che sono state eseguite per realizzarlo.

2.2 Modelli per il Deep Learning

Se si volesse dare una definizione di *Deep Learning*, si potrebbe dire che consiste in «una classe di tecniche di *Machine Learning*, le quali sfruttano l'elaborazione non lineare dei dati su diversi livelli, allo scopo di estrarre e trasformare *feature*, effettuare classificazione ed analisi di *pattern* ricorrenti, in modo supervisionato o non supervisionato»[1]. Una definizione che assume una sfumatura meno tecnica e forse più filosofica, è la seguente: «Il *Deep Learning* è una nuova area di ricerca

[1] Li Deng, Dong Yu - Deep Learning, Methods and Applications (originariamente pubblicato come Foundations and Trends in Signal Processing, Volume 7, Issues 3-4, pp 197-387), 2014.

del *Machine Learning*, che è stata introdotta con l'obiettivo di avvicinare il *Machine Learning* ad una delle sue finalità originali: l'Intelligenza Artificiale. Il *Deep Learning* riguarda l'apprendimento multi-livello della rappresentazione e dell'astrazione, che aiuta a dare un senso a dati come immagini, suoni e testi»[2].

Esistono in letteratura svariati tipi di architetture profonde, di cui segue una rapida rassgna. Le *Deep Belief Networks*, uno dei primi modelli introdotti da Geoffrey E. Hinton e R. R. Salakhutdinov nel loro lavoro del 2006[3], sono un modello probabilistico generativo, costituito da diversi livelli di unità nascoste, e possono essere viste come una composizione di semplici moduli che rappresentano il singolo livello. Ciascun modulo può essere realizzato tramite una *Restricted Boltzmann Machines*, un modello grafico probabilistico (rappresentabile tramite un grafo bipartito che connette unità *visible* a unità *hidden*) di tipo *energy-based* (dove il *learning* corrisponde a modificare la funzione energia, un valore scalare associato ad ogni configurazione delle variabili) utilizzato per evidenziare determinate caratteristiche e regolarità sui dati che riceve in ingresso. L'aggettivo *'restricted'* evidenzia il fatto che tra unità dello stesso livello non esistono collegamenti. Altri esemplari di reti neurali profonde sono gli auto-encoder, un modello di apprendimento non supervisionato, architetturalmente simile al *multilayer perceptron*, con un livello di input, uno di output e, nel mezzo, uno o più livelli nascosti interconnessi tra loro. Differentemente dal *multilayer perceptron*, nell'*auto-encoder* il livello di output ha lo stesso numero di nodi di quello di input e, inoltre, da un insieme di dati d'ingresso X, non viene addestrato con lo scopo di predire un insieme di dati target Y, bensì con quello di

[2] http://deeplearning.net/tutorial/
[3] Hinton, Slakhutdinov – Reducing the Dimensionality of Data with Neural Networks, Science, Volume 313, pp 504-507, 2006

ricostruire lo stesso insieme X. Inoltre, degne di nota sono le *Convolutional Neural Networks*, che hanno una struttura che si adatta al trattamento di immagini e video e le *Recurrent Neural Networks*, impiegate per lo più nel riconoscimento della scrittura a mano o in quello del parlato.

Per il problema affrontato durante il presente lavoro di tesi, si è scelto lo studio e l'approfondimento di uno di questi, ovvero sia degli *auto-encoder*, in una loro variante denominata *stacked denoising auto-encoder*. I motivi che hanno portato a questa scelta sono stati molteplici. Innanzitutto, si tratta di uno dei modelli attualmente allo stato dell'arte in numerose applicazioni. Inoltre, si è pensato che tale soluzione potesse bene adattarsi e risultare utile alla situazione analizzata, dove si richiedeva l'estrazione di *feature* tra cui era possibile un legame non lineare difficilmente individuabile a priori, e avendo a che fare con un gran numero di dati non etichettati, che quindi han suggerito l'impiego di una tecnica di apprendimento semi-supervisionata. Per ultimo, il fatto che, nella letteratura analizzata, l'impiego di questa struttura sia scarsamente messo alla prova in problemi come la previsione dei dati di serie temporali.

2.3 Auto-encoder

L'*auto-encoder*, chiamato anche auto-associatore o *Diabolo Network*, è un'architettura allenata per codificare l'input x in una certa sua rappresentazione $c(x)$, in modo tale che successivamente lo stesso input possa essere ricostruito da questa rappresentazione. In questo modo l'output dell'*auto-encoder* risulta essere lo stesso input. Se la funzione di attivazione che regola l'attivazione dei neuroni del livello nascosto è lineare, e per allenare la rete si adopera il criterio dell'errore

quadratico medio, allora le k unità nascoste imparano a proiettare l'input nell'intervallo dei primi k componenti principali dei dati. Se, invece, al livello nascosto si conferisce in qualche modo la caratteristica di non linearità, l'*auto-encoder* acquisisce la capacità di catturare aspetti multi-modali relativamente alla distribuzione dell'input. Volendo schematizzare, si può dire che l'*auto-encoder* riceve come input un vettore $x \in [0,1]^d$, mappandolo (per mezzo di un *encoder*) in una rappresentazione nascosta $h \in [0,1]^{d'}$ tramite un *mapping* deterministico definito come

$$h = s(Wx + b),$$

dove i parametri W e b rappresentano, rispettivamente, la matrice dei pesi e il vettore dei *bias* (definiti dalla rete) ed s una funzione non lineare come, ad esempio, la funzione sigmoide. Successivamente, h è ri-mappata (grazie ad un *decoder*) in una ricostruzione y, che ha la stessa forma di x. Anche questo *mapping* avviene tramite una trasformazione simile

$$y = s(W'x + b').$$

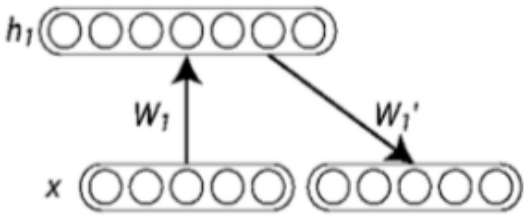

Figura 2.4 - L'auto-encoder (fonte http://www.iro.umontreal.ca/~bengioy/talks/Google230909.pdf)

Una formulazione interessante generalizza il criterio dell'errore quadratico medio alla minimizzazione della *log-likelihood negativa* della ricostruzione, data la codifica $c(x)$, definita come

$$-logP(x|c(x)).$$

Se $x|c(x)$ è gaussiano, ritroviamo il classico errore quadratico. Se gli input x_i sono o funzioni binarie, o hanno distribuzione di probabilità binomiale, allora la *loss function* diventa

$$-logP\big(x\big|c(x)\big) = -\sum_i x_i log f_i\big(c(x)\big) + (1-x_i)\log\Big(1-f_i\big(c(x)\big)\Big),$$

dove $f(\cdot)$ è detta *decoder* e $f(c(x))$ è la ricostruzione prodotta dalla rete che, in questo caso, è un vettore di numeri compresi tra 0 e 1 e ottenuti, ad esempio, con una funzione di attivazione sigmoidale.

L'aspettativa è che la codifica $c(x)$ sia una rappresentazione distribuita che catturi i principali fattori di variazione all'interno dei dati: poiché $c(x)$ è vista come una compressione con perdita di x, non può essere una buona compressione (con una perdita piccola) per tutti gli ingressi x in modo uniforme, così l'apprendimento porta ad averne una migliore specialmente per gli esempi trattati durante l'addestramento e, sperabilmente, anche per altri casi particolari (questo è il modo in cui l'*auto-encoder* generalizza), ma non per tutti in modo arbitrario. Un aspetto importante da sottolineare riguarda il fatto che con questo approccio, un *auto-encoder* con una dimensione dell'input non superiore rispetto a quella della funzione di codifica potrebbe apprendere banalmente la funzione identità, senza essere di alcuna utilità. Sorprendentemente, alcuni esperimenti evidenziano che, nella pratica, quando addestrati con il metodo stocastico della discesa del

gradiente, gli *auto-encoder* non lineari con più unità nascoste rispetto agli input forniscono rappresentazioni utili (in termini di errore di classificazione misurato su una rete che prenda in input la stessa rappresentazione).

Per raggiungere una ricostruzione accurata di input continui, un *auto-encoder* con un livello nascosto e unità nascoste non lineari ha bisogno di pesi molto piccoli nel primo livello (per portare a regime la non linearità delle sue unità nascoste) e pesi molto grandi nel secondo livello. Dal momento che la regolarizzazione implicita o esplicita rende difficile raggiungere le soluzioni con i pesi più grandi, l'algoritmo di ottimizzazione trova esclusivamente quelle codifiche che lavorano bene per gli esempi simili a quelli nel *training set*, piuttosto che apprendere la semplice ricostruzione della funzione identità.

Esistono diversi modi in cui si può prevenire che un *auto-encoder* con più unità nascoste impari solo la funzione identità, e far sì quindi che riesca a catturare qualcosa di utile riguardo la rappresentazione dell'input. Una strategia spesso vincente è introdurre del rumore nella codifica, per migliorare la robustezza. Il *denoising auto-encoder*, infatti, minimizza l'errore in ricostruzione a partire da un input stocasticamente corrotto. Può essere dimostrato che massimizza il *lower bound* della *log-likelihood* di un modello generativo.

2.3.1 Denoising auto-encoder

Il *denoising auto-encoder* è una versione stocastica dell'*auto-encoder*, nella quale l'input è corrotto stocasticamente, ma la versione non corrotta dell'input stesso viene usata come target per la ricostruzione.

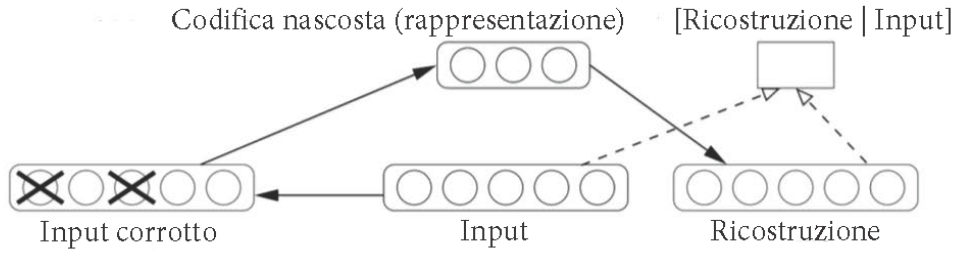

Figura 2.5 - Il denoising auto-encoder (fonte http://www.iro.umontreal.ca/~bengioy/talks/Google230909.pdf)

Intuitivamente, un *denoising auto-encoder* fa due cose: cerca di codificare l'input (preservandone le informazioni che lo riguardano) e cerca di annullare l'effetto del processo di corruzione stocasticamente applicato all'input medesimo. Quest'ultima operazione può essere compiuta esclusivamente catturando le dipendenze statistiche tra i valori in ingresso e, nella pratica, il processo di corruzione stocastica consiste nel modificare portando a zero alcuni di questi valori. Per cui il *denoising auto-encoder* proverà a predire i valori mancanti da quelli presenti, per sottoinsiemi dei pattern mancati, aleatoriamente selezionati. Il criterio di training può essere espresso tramite la *log-likelihood* della ricostruzione

$$-logP\left(x\middle|c(\tilde{x})\right),$$

dove x è l'input non corrotto, $\tilde{x}$ quello stocasticamente corrotto e $c(\tilde{x})$ la codifica ottenuta da $\tilde{x}$. Nel complesso, questa strategia ha un grande successo nel pre-training non supervisionato di architetture profonde, soprattutto se poi le si fa seguito con un modello generativo.

2.3.2 Stacked denoising auto-encoder

I *denoising auto-encoder* possono essere impilati uno sull'altro per formare ed inizializzare una rete neurale profonda multi-livello.

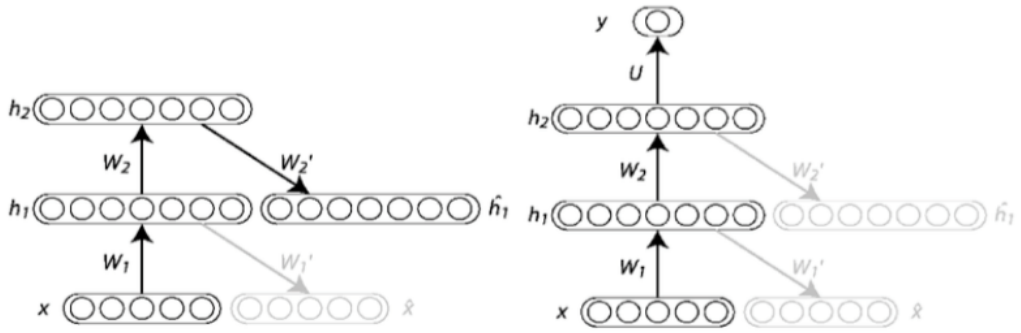

Figura 2.6 - Lo stacked denoising auto-encoder (fonte http://www.iro.umontreal.ca/~bengioy/talks/Google230909.pdf)

La procedura di training è simile a quella delle *Deep Belief Network* e consiste in 5 passi:

1. Si addestra il primo livello come semplice *denoising auto-encoder*, per minimizzare l'errore di ricostruzione dell'input grezzo. È una fase puramente non supervisionata.
2. L'output delle unità nascoste (cioè le codifiche) del *denoising auto-encoder* viene usato come input del livello successivo, addestrato a sua volta come un *denoising auto-encoder*. Anche questa è una fase dove non è richiesto l'etichettamento dei dati.
3. Si itera come nel passo (2) per inizializzare un numero arbitrario di livelli nascosti.
4. Si considera l'output dell'ultimo livello nascosto come input di un top-level supervisionato e se ne inizializzano i parametri (in modo aleatorio oppure tramite un addestramento supervisionato, mantenendo il resto della rete fisso).

5. Si effettua il cosiddetto *fine-tuning* della rete, settando i parametri relativamente al criterio di supervisione scelto. Alternativamente, si uniscono i *denoising auto-encoder* in uno molto profondo e si effettua il *fine-tuning* dell'errore globale di ricostruzione.

3. CASO DI STUDIO

3.1 Analisi dei dati

Seaside dispone di un database in cui memorizza diversi dati provenienti da sensori dislocati negli impianti che gestisce, con una frequenza di trasmissione che, di solito, è di un pacchetto di misurazioni ogni 15 minuti. Una volta importati nel database, le informazioni possono essere visualizzate e analizzate tramite SSRS (SQLServer Reporting Services) e un software che gestisce il supporto alla *Business Intelligence* e alla visualizzazione interattiva, e che permette la creazione dinamica di *dashboard*. Il database è stato progettato ed implementato secondo il modello dimensionale descritto da Kimball, in particolare utilizzando il modello a stella (altrimenti detto *star schema*) come riferimento. Tale modello prevede la creazione delle tabelle dei fatti al centro dello schema, circondate dalle tabelle delle dimensioni. Attualmente vi sono presenti due tabelle dei fatti: EnergyData, contenente i dati in arrivo dai sistemi di monitoraggio e EnergyPrice, contenente i prezzi dell'energia per mercato, data e ora.

In Tabella 3.1 sono descritte invece le tabelle delle dimensioni.

Nome	Descrizione
Fasce	Vi sono definite le fasce orarie per impianto. Per ogni impianto, giorno della settimana, ora, viene assegnata la fascia oraria di appartenenza.
Calendario	Vi è stato definito il calendario dal 01/01/2000 al 31/12/2030. Per ogni giorno, la tabella contiene la descrizione di settimana,

	mese, trimestre, anno di appartenenza, la descrizione in lingua italiana e inglese del giorno della settimana, e se si tratta di un giorno festivo o lavorativo secondo il calendario italiano.
Dispositivi	Contiene la definizione dei dispositivi usati per il monitoraggio. In particolare viene definita sia la descrizione del dispositivo, sia la gerarchia rispetto al gateway di riferimento e rispetto ad altri device. Esistono inoltre altri campi della tabella usati per descrivere i dispositivi legati ad impianti fotovoltaici. La dimensione è gestita come una SCD (Slowly Changing Dimension).
Gateway	Contiene la definizione dei gateway e la loro gerarchia, il riferimento all'impianto di appartenenza e la sua posizione all'interno dell'impianto.
Metriche	Contiene tutte le metriche che è possibile utilizzare. Per ogni metrica, viene definito un nome, una unità di misura e se in fase di ETL occorre calcolare il dato puntuale a partire dal progressivo, fornito dallo strumento di lettura.
Organization	Vi sono definite le proprietà degli impianti. È composta da due livelli (Organizzazione e Owner) in modo da poter gestire il caso delle Società Veicolo per gli impianti fotovoltaici.
Impianto	Contiene l'anagrafica degli impianti. Oltre ai campi inerenti alla localizzazione geografica, contiene alcuni attributi usati per caratterizzare l'impianto: • se si tratta di un impianto fotovoltaico o di una struttura energivora • il cluster di appartenenza dell'impianto • la potenze nominale • TEP: rappresenta il fattore di conversione per il calcolo delle Tonnellate di Petrolio Equivalenti • CO2: rappresenta il fattore di conversione per il calcolo delle tonnellate di CO2 non emesse
Product	Contiene l'anagrafica dei prodotti. Un prodotto è la generalizzazione di un dispositivo: in pratica un dispositivo può essere definito come l'istanza di un prodotto.
Time	Contiene una riga per ogni coppia ora-minuto dalle 00:00 alle 23:59.

Tabella 3.1 - DB, tabelle delle dimensioni

3.2 Scelta del caso di studio

Dopo un'analisi della qualità e quantità dei dati a disposizione, che ha permesso di definire due macro-classi di realtà monitorate, cioè quella delle strutture che consumano energia elettrica e quella degli impianti di produzione, la scelta, effettuata sotto la guida di un responsabile dell'azienda, è ricaduta su uno di questi ultimi. Il caso infatti è di particolare interesse per l'azienda stessa e i dati disponibili sono risultati adeguati per approfondirne la trattazione. Tuttavia, la scelta di un particolare caso di studio non è assolutamente restrittiva: sarà possibile, in futuro, trasferire le metodologie studiate e sviluppate a qualsiasi altra situazione in cui lo scopo sia identico a quello preso in esame, cioè la previsione di valori in una serie temporale, *mutatis mutandis*.

Quella selezionata è un'azienda di produzione industriale, che sopperisce ad una parte del fabbisogno di energia elettrica interno tramite un impianto di produzione fotovoltaico. In particolare, è stato monitorato uno degli impianti di cui dispone tale azienda, in cui la produzione avviene per mezzo di 108 moduli fotovoltaici ciascuno con una potenza di 230W, per un totale di 24,84kW di potenza complessiva, installati sopra il tetto di un capannone situato a pochi chilometri di distanza da Bologna. L'impianto è composto da 6 stringhe di moduli fotovoltaici collegate in parallelo, ognuna composta da 18 unità collegate in serie. Le stringhe sono connesse in seguito ad un contatore e, successivamente, ad un inverter, per la trasformazione della corrente da continua ad alternata, prima dell'immissione in rete. Il contatore, dal quale sono state rilevate le misurazioni riguardanti la produzione, è un contatore di impulsi, in cui il valore in kWh dell'energia prodotta si ricava dividendo per 100 la misura da esso fornita.

3.3 Pulizia dei dati e normalizzazione

Una volta scelti l'impianto e la metrica da analizzare, si sono estratti i dati dal database online tramite interrogazioni SQL, prima in un foglio Excel e, successivamente, in un file *.csv* in cui ogni riga rappresentasse una coppia di valori [data-ora, misurazione], tramite la scrittura di uno *script* in Python che eseguisse automaticamente l'operazione, accettando come parametri di ingresso la finestra temporale desiderata, l'impianto e la metrica di misurazione. Quasi immediatamente è emerso che in alcuni (relativamente pochi, a dire il vero) casi la qualità non era sufficiente, prevalentemente per quattro motivi: la mancanza di dati in determinati *slot* temporali, la presenza di valori decisamente al di sopra della media, la replicazione di alcune registrazioni o l'errore sull'orario della misurazione.

In realtà, più che sulla misurazione, tutti questi quattro errori sono probabilmente dovuti alla trasmissione dei valori misurati. La prima e la seconda problematica, infatti, sono tra loro legate: poiché lo strumento di misura è un contatore cumulativo incrementale, se una o più trasmissioni non vengono effettuate nell'istante previsto, quando la comunicazione riprende, la differenza tra il valore comunicato e l'ultimo valore noto risulta essere la somma di tutti gli slot temporali non trasmessi. Dal momento che l'analisi si basa sul valore istantaneo, cioè sulla differenza, e non su quello incrementale, è evidente che, a fronte di trasmissioni non effettuate, ci siano dei valori esageratamente grandi da spalmare sulle misurazioni degli istanti precedenti. Anche la replicazione dell'invio della stessa misurazione e gli errori sul *timestamp* sono, con grande probabilità, imputabili ad un fallimento del sistema di trasmissione.

Pur esulando per certi versi dagli scopi di questa tesi (ma essendo ad ogni modo un'operazione necessaria, e sui cui metodi implementativi si potrebbe approfondire di molto lo studio), è stato effettuato un processo di pulizia dei dati (in gran parte in modo automatico attraverso uno *script*, per il resto manualmente) che provvedesse a:

- Individuare e correggere gli errori sui *timestamp* sbagliati (cioè tutte le misurazioni devono pervenire in un orario multiplo di 15 minuti).
- Individuare ed eliminare i valori delle misurazioni oltre una certa soglia (600 impulsi).
- Individuare ed eliminare le misurazioni replicate.
- Individuare e correggere gli *slot* in cui la misurazione è mancante (ricopiando il valore dello stesso slot temporale del giorno prima, oppure del primo giorno passato in cui è presente).

A questo punto, come ultima operazione sui dati prima del loro trattamento, è stata effettuata una normalizzazione, dividendo ciascuna misurazione per il massimo valore registrato, e scalandole quindi all'interno dell'intervallo tra 0 e 1. Questo è servito per facilitare il lavoro degli strumenti utilizzati in seguito.

3.4 Modelli utilizzati

I modelli impiegati per effettuare la predizione sono quelli già descritti nei capitoli precedenti: il *random walk* e la regressione lineare multipla. Per quanto riguarda il primo, si tratta di un modello del tutto privo di comportamento "intelligente" che non ha bisogno di particolari manipolazioni dei dati in ingresso: semplicemente il valore predetto $v_{t+\Delta t}$ è identico a quello presente, v_t.

Il secondo modello invece, quello di regressione lineare multipla, vuole in input un *data set* costituito da tanti vettori quante sono le osservazioni, ognuno dei quali formato da un valore target (il primo elemento) $v_{t+\Delta t}$ che rappresenta il valore predetto, seguito da una coda di altri valori v_t, v_{t-1}, …, v_{t-w}. In questa notazione, Δt rappresenta l'orizzonte temporale in cui si andrà ad effettuare la previsione e w il numero di variabili esplicative tenute in considerazione per la modellazione.

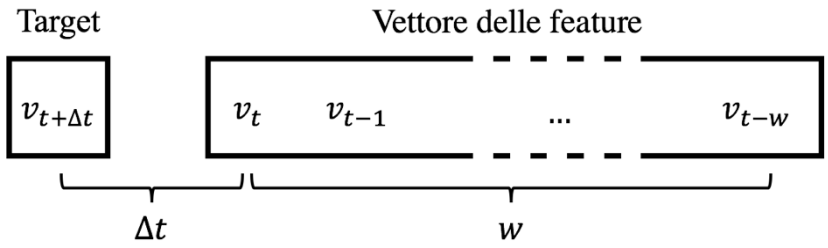

Figura 3.1 - Formato di input dei dati

Fissato un valore per Δt (che in questo caso è stato posto a un'ora, poiché tale è l'orizzonte temporale in cui l'azienda è interessata ad ottenere la previsione) e scelto accuratamente un valore w adeguato, la regressione lineare multipla approssima già con una certa precisione l'andamento della curva di produzione.

Per tentare di migliorare tale risultato, si è progettata un'estensione del modello appena descritto, in cui prima del sistema che effettua la regressione lineare multipla viene inserita una rete neurale profonda formata da *stacked denoising auto-encoder*, con l'intento di cogliere ed evidenziare a priori eventuali dipendenze non lineari tra i dati. Quindi il vettore di dati che in precedenza era l'input della regressione, in questa nuova configurazione diventa l'input dello *stacked denoising auto-encoder*, il quale ne fornisce uno perfettamente identico nella forma e nelle dimensioni, con lo stesso target $v_{t+\Delta t}$, ma con i valori v'_t,

$v'_{t-1}, \ldots, v'_{t-w}$ modificati a seconda del peso che ciascun valore ha nella predizione del valore all'istante t.

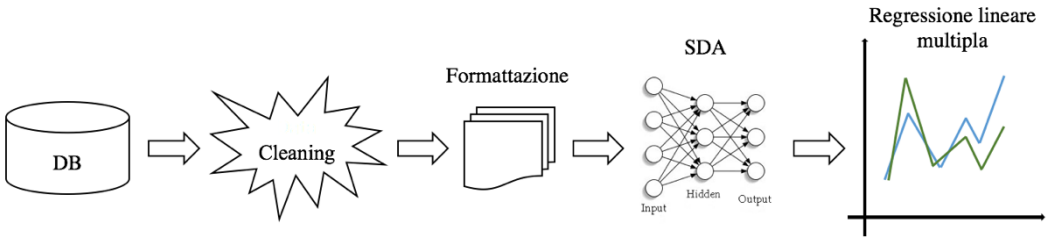

Figura 3.2 - Modello realizzato

Infine, si è costruito un *framework* per il testing dei modelli con diverse combinazioni di parametri, in modo da provare quali di queste restituissero i risultati migliori. I parametri modificabili sono stati:

- Dimensione del *data set* (quanto larga è la finestra temporale su cui si costruisce il modello)
- Numero di variabili esplicative considerate (quanto larga è la finestra temporale dei dati che vanno ad influire sul valore predetto)
- Numero di strati nascosti della rete (da quanti *denoising auto-encoder* è formata la rete)
- Numero di neuroni di ciascuno strato nascosto della rete (da quanti neuroni è formato ciascun *denoising auto-encoder*)
- Livello di corruzione per ciascun *denoising auto-encoder*
- Valore del $learning\_rate$ (parametro che regola quanto velocemente vengono variati i pesi della rete nel corso delle epoche)
- Grandezza di un $mini\_batch$ (parametro implementativo del training di ciascun *denoising auto-encoder* che corrisponde al numero di esempi processati alla volta)

Il testing è stato seguito da una fase di confronto dei risultati sperimentali ottenuti, durante la quale si sono assunti come riferimento gli indicatori introdotti in precedenza, cioè l'RMSE (*Root Mean Square Error*) definito come

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2},$$

la MAD (Mean Absolute Deviation) definita come

$$MAD = \frac{1}{n}\sum_{i=1}^{n}|\tilde{y}_i - y_i|$$

e, per i modelli di regressione lineare, il coefficiente di determinazione R^2 nell'implementazione fornita della libreria Python scikit-learn 0.17[4].

3.5 Cenni sull'implementazione

Una parte sostanziale del lavoro di tesi descritto in questo documento è consistita nell'implementazione e lo sviluppo, tramite il linguaggio di programmazione Python versione 2.7 e la libreria di supporto Theano versione 0.7, di uno *stacked denoising auto-encoder*. Anche se poi di fatto non si è sfruttata questa caratteristica per via della configurazione hardware su cui si sono svolti i test, l'aver scritto il codice con Theano offrirebbe la possibilità di sfruttare il calcolo parallelo su GPU, diminuendo di molto il *runtime* del programma. Nulla vieta, in futuro,

[4] http://scikit-learn.org/stable/modules/model_evaluation.html#r2-score-the-coefficient-of-determination

di adottare una piattaforma che supporti tale modalità e, nel caso in cui si volessero effettuare ulteriori test, lo si potrebbe fare in modo più veloce ed efficace.

Nel dettaglio, è stata implementata una classe che simuli un singolo *denoising auto-encoder*, specificando il numero di unità visibili (la dimensione d dell'input), il numero di unità nascoste (la dimensione d' del livello nascosto) ed il $corruption\_level$ (cioè la percentuale di corruzione dell'input). La classe rappresenta la rete tramite le variabili W, b_{vis} e b_{hid}, cioè la matrice dei pesi di ciascun collegamento tra ogni neurone di un livello con ogni neurone di un livello successivo e i due vettori dei *bias*, quello dello strato visibile e quello dello strato nascosto. La corruzione è introdotta mantenendo $(1 - corruption\_level)$ ingressi uguali a se stessi e ponendo $(corruption\_level)$ ingressi pari a 0, in modo aleatorio. È possibile, inoltre, settare i parametri del $learning\_rate$ e della dimensione dei $mini\_batch$, cioè la velocità con cui la rete apprende durante le varie epoche e quanti esempi alla volta processa.

Da qui si è proceduto implementando il modello *stacked*, formato da più oggetti *denoising auto-encoder* idealmente impilati l'uno sull'altro, in cui l'output del livello i-esimo (cioè la matrice dei pesi W e il vettore dei *bias* b_{hid}) diventa l'input del livello $(i + 1)$-esimo.

Come ultima operazione, preliminare rispetto alla fase di testing, è stato scelto un sottoinsieme del periodo temporale all'interno del quale fossero disponibili i dati (in realtà più di uno: la realizzazione è stata tale che questa scelta fosse una variabile implementativa, con la possibilità di stabilirne la grandezza durante la fase di test) sul quale effettuare l'apprendimento. Successivamente, questo *data set* è stato suddiviso in tre parti: un *training set* (55% del *data set* iniziale), un *validation set* (15% del *data set* iniziale) e un *test set* (40% del *data set*

iniziale), in modo da effettuare l'apprendimento della rete sul *training set* e testarne i risultati sul *test set*. Il *validation set* è servito per realizzare il meccanismo di *early stopping*: la rete termina l'apprendimento dopo un certo numero di epoche in cui l'errore sul *validation set* (con i parametri imparati sul *training set*) non diminuisce (nei test effettuati questo numero è stato fissato a 5).

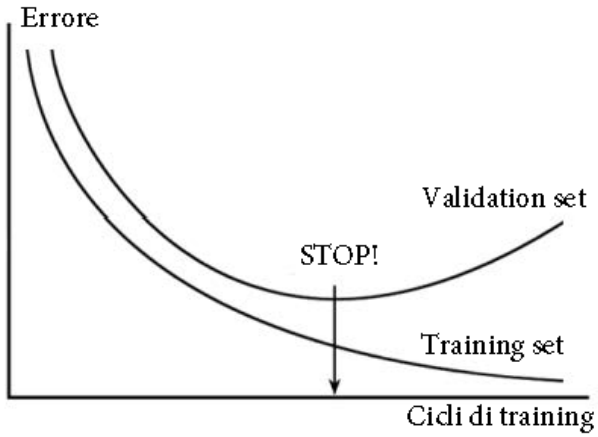

Figura 3.3 - Early-stopping per evitare overfitting

Questo meccanismo è uno di quelli utili ad evitare l'*overfitting* della rete, ovverosia la tendenza che ha il modello ad adattare i propri parametri in maniera eccessiva rispetto ai soli casi visionati, perdendo di generalità quando viene messo alla prova sui casi il cui output sia ignoto.

Per quanto riguarda il metodo scelto per l'ottimizzazione, si è optato per quello di discesa stocastica del gradiente: una tecnica di ottimizzazione che, data una funzione matematica a più dimensioni, permette di trovarne un minimo. L'implementazione consiste nel valutare inizialmente sia la funzione che il suo gradiente, a partire da una configurazione scelta casualmente nello spazio delle dimensioni. Da qui, si cerca di spostarsi nella direzione indicata dal gradiente (che stabilisce un verso di discesa in cui la funzione tende ad un minimo) e

si esamina se la funzione assume in effetti un valore inferiore a quello calcolato nella configurazione precedente. Se così fosse, il procedimento proseguirebbe iterativamente, ricalcolando il nuovo gradiente (che potrebbe essere totalmente differente dal precedente) e riprendendo alla ricerca di un nuovo minimo.

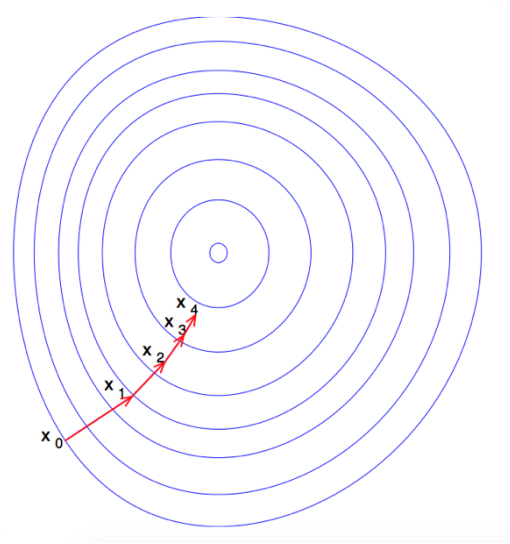

Figura 3.4 - Metodo di discesa stocastica del gradiente

Nel caso in questione, la funzione da minimizzare è rappresentata dalla funzione di costo, definita come la media di tutti i costi di *cross-entropy* (vedi paragrafo 2.3) della ricostruzione dell'esempio corrispondente, e la cui variazione dipende dall'efficacia dei pesi (della matrice W) che permettono alla rete di giungere all'apprendimento durante la fase di training. Per questo motivo avanzare di un passo durante il cammino di discesa equivale a modificare la matrice W dei pesi, cambiando la configurazione della rete. Rispetto al metodo di discesa del gradiente standard, in cui i pesi sono aggiornati dopo aver calcolato il gradiente per l'intero *training set*, in quello stocastico (a *mini-batch*) i pesi vengono aggiornati dopo un certo numero di esempi trattati, scelti

casualmente, per velocizzare il processo e tentare di evitare eventuali situazioni di minimo locale.

La rapidità con cui si effettuano questi passi è definita dal $learning\_rate$: un tasso di apprendimento maggiore equivale ad una granularità più grossa (l'algoritmo è più veloce ma meno preciso), viceversa con un tasso di apprendimento più basso la granularità sarà via via più sottile (cioè la differenza tra un passo ed il successivo sarà meno marcata, l'algoritmo sarà più lento ma avrà più probabilità di trovare un minimo migliore).

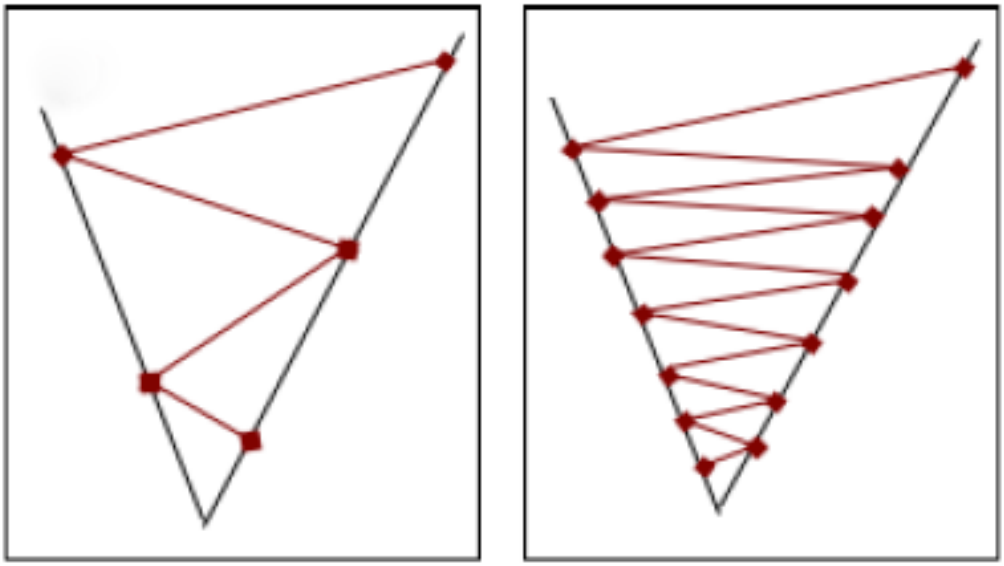

Figura 3.5 - Esempi di SGD con differenti learning rate

4. RISULTATI SPERIMENTALI

4.1 Configurazione dei test

Una rete neurale funziona più o meno bene a seconda di come vengono configurati i parametri che la caratterizzano. Purtroppo, effettuare il *tuning* di un'architettura di questo tipo, cioè assegnare a ciascun parametro il valore migliore, in modo da ottimizzare i risultati prodotti dalla rete, è una procedura scarsamente formalizzabile, e in gran parte dipendente dall'esperienza di chi la crea e dal numero di prove che si effettuano. Nel caso in questione sono state effettuate, compatibilmente con le risorse a disposizione (di tempo e di calcolo), diverse centinaia di test. I risultati esposti in questo capitolo evidenziano quelli che sono gli aspetti più interessanti e maggiormente apprezzabili dal punto di vista delle prestazioni del modello implementato. Ciò nonostante, considerato che le combinazioni di parametri possibili sono pressoché infinite, tali evidenze dovrebbero essere valutate come un indirizzo – seppur relativamente preciso in alcuni casi – rispetto alla direzione in cui dirigersi, e non un risultato definitivo. Sarà possibile, in futuro, approfondire ulteriormente lo studio delle combinazioni più promettenti, o provarne di nuove, per tentare di ottenere un rendimento superiore.

Tutti i test sono stati effettuati su due computer:

- Apple MacBook Pro 15", 2.2GHz Intel Core i7, 4GB DDR3 1333MHz, SSD 256GB

- Apple MacBook Pro 15”, 2.2GHz Intel Core i7, 16GB DDR3 1600MHz, SSD 256GB

I parametri considerati e modificati durante i test sono stati:

- Data d’inizio e data di fine delle osservazioni
- Numero di variabili esplicative per la regressione, corrispondente al numero di valori in ingresso ed in uscita della rete neurale (w)
- Numero di livelli nascosti
- Numero di neuroni per ciascun livello nascosto
- Livello di corruzione
- Tasso di apprendimento
- Grandezza dei *mini-batch*

I modelli con cui è stato confrontato il modello sono stati:

- *Random walk*
- Regressione lineare senza l’uso della rete neurale

Le metriche di confronto considerate sono state:

- Valore del RMSE
- Valore del MAD
- Valore del coefficiente R^2 della regressione lineare

Quella che viene presentata di seguito è una rassegna degli esempi ottenuti con le configurazioni più interessanti, dopo aver analizzato i vari risultati e avendo esclusi i peggiori.

4.2 Auto-encoder

Già con l'introduzione di un singolo livello e senza alcuna percentuale di corruzione dell'input, le prestazioni rispetto alla regressione lineare ed al *random walk* migliorano, in generale. Nella seguente tabella sono riportati alcuni dei risultati migliori ottenuti dai test effettuati con un solo auto-encoder inserito subito prima della regressione lineare multipla.

REGRESSIONE LINEARE			SDA			RW	w	LR	# hid	Corr	MB	Epoche	ΔRMSE	ΔMAD	ΔR^2	SDA	
RMSE	MAD	R^2	RMSE	MAD	r^2	RMSE										RMSE_valid	MAD_valid
0,07596	0,04302	0,84584	0,07124	0,03277	0,88030	0,08205	5	0,01	50	0	10	6	0,00473	0,01025	0,03446	0,11109	0,05692
0,07596	0,04302	0,84584	0,07213	0,03161	0,89083	0,08205	5	0,01	180	0	10	10	0,00383	0,01142	0,04499	0,11212	0,05419
0,07596	0,04302	0,84584	0,07158	0,03185	0,89109	0,08205	5	0,01	240	0	10	7	0,00438	0,01117	0,04525	0,11046	0,05392
0,07596	0,04302	0,84584	0,07356	0,03269	0,89085	0,08205	5	0,01	500	0	10	12	0,00240	0,01033	0,04501	0,11046	0,05543
0,07705	0,04977	0,85986	0,07510	0,03789	0,89561	0,08205	15	0,05	60	0	40	219	0,00195	0,01188	0,03575	0,10323	0,05377
0,07705	0,04977	0,85986	0,07259	0,03652	0,89893	0,08205	15	0,05	120	0	40	24	0,00446	0,01324	0,03907	0,10534	0,05569
0,07705	0,04977	0,85986	0,07201	0,03531	0,90293	0,08205	15	0,05	200	0	40	91	0,00504	0,01446	0,04307	0,10473	0,05347
0,07791	0,05174	0,86071	0,07823	0,04402	0,88604	0,08205	30	0,05	60	0	40	10	-0,00033	0,00773	0,02533	0,10310	0,05957
0,07791	0,05174	0,86071	0,07746	0,03959	0,90031	0,08205	30	0,05	120	0	40	1214	0,00045	0,01215	0,03960	0,09837	0,05258
0,07791	0,05174	0,86071	0,07464	0,04047	0,89914	0,08205	30	0,05	200	0	40	36	0,00327	0,01127	0,03843	0,10075	0,05563
0,08140	0,05758	0,86718	0,07509	0,03984	0,89882	0,08205	60	0,05	200	0	10	104	0,00631	0,01774	0,03164	0,09875	0,05203
0,08140	0,05758	0,86718	0,07480	0,03899	0,90100	0,08205	60	0,05	300	0	10	114	0,00660	0,01859	0,03382	0,09745	0,05072
0,08140	0,05758	0,86718	0,07463	0,03946	0,90137	0,08205	60	0,05	400	0	10	66	0,00677	0,01812	0,03419	0,09866	0,05185
0,08140	0,05758	0,86718	0,07349	0,03903	0,91043	0,08205	60	0,05	1500	0	15	6	0,00791	0,01855	0,04325	0,10362	0,05757
0,06697	0,03453	0,89130	0,06225	0,02857	0,91661	0,08205	90	0,01	500	0	10	51	0,00472	0,00596	0,02532	0,09571	0,04839
0,06690	0,03499	0,89349	0,06393	0,02959	0,91027	0,08205	100	0,05	200	0	10	11	0,00297	0,00540	0,01678	0,09539	0,04771
0,06690	0,03499	0,89349	0,06371	0,03048	0,91318	0,08205	100	0,05	300	0	10	7	0,00319	0,00452	0,01968	0,09513	0,04751
0,06690	0,03499	0,89349	0,06386	0,03075	0,91390	0,08205	100	0,05	400	0	10	6	0,00304	0,00425	0,02040	0,09560	0,04900
0,06690	0,03499	0,89349	0,06235	0,02875	0,92920	0,08205	100	0,05	1500	0	15	6	0,00456	0,00624	0,03571	0,10262	0,05409
0,06685	0,03534	0,89421	0,06521	0,03199	0,91171	0,08205	120	0,01	180	0	10	641	0,00165	0,00335	0,01750	0,09475	0,04758
0,06685	0,03534	0,89421	0,06403	0,03089	0,91696	0,08205	120	0,01	500	0	10	8	0,00283	0,00445	0,02275	0,09686	0,04982
0,06685	0,03534	0,89421	0,06343	0,02998	0,91942	0,08205	120	0,1	500	0	10	20	0,00342	0,00535	0,02520	0,09625	0,04780

Tabella 4.1 - Risultati test con auto-encoder (in giallo i risultati migliori)

4.2.1 Best RMSE e MAD sul validation set

Si può notare che il miglior risultato per il RMSE (0,09475) sul *validation set* corrisponde alla configurazione che prevede 120 *feature* in input e uno strato nascosto di 180 neuroni, con un *learning rate* di 0,01 e *mini batch* di 10 elementi. Un aspetto decisamente interessante

da sottolineare è il fatto che, in questo esempio, il numero delle epoche di apprendimento sia stato piuttosto elevato (641), vale a dire che la rete ha in effetti svolto un considerevole lavoro di apprendimento.

Per quanto riguarda il MAD sul *validation set*, il valore migliore è 0,04751 ed è ottenuto con 100 *feature* in input e uno strato nascosto di 300 neuroni, con un *learning rate* di 0,05 e *mini batch* di 10 elementi. In questo caso però il numero delle epoche è basso, ed è quindi possibile che la stessa configurazione di rete possa essere migliorata con un più accurato *tuning* dei parametri.

4.2.2 Best R^2

Il valore di R^2 più alto riscontrato nei test, con l'impiego di un *auto-encoder* singolo, risulta 0,92920 ed è ottenuto con 100 *feature* in input alle quali segue uno strato nascosto di 1500 neuroni, con *mini batch* di 15 elementi.

4.2.3 Best ΔR^2 e ΔRMSE

Volendo effettuare un confronto con la regressione lineare ed il *random walk*, si può notare che le configurazioni della rete che maggiormente contribuiscono al miglioramento dei risultati sono quella con 5 *feature* in input, uno strato nascosto di 140 neuroni, *learning rate* 0,01 e *mini batch* di 10 elementi (che fa incrementare R^2 di 0,04525) e quella con 60 *feature* in input, uno strato nascosto di 1500 neuroni, *learning rate* 0,05 e *mini batch* di 15 elementi (che fa incrementare il RMSE di 0,00791).

4.3 Denoising auto-encoder

In alcuni casi, l'effetto della corruzione dell'input è positivo sulle prestazioni del sistema. Di seguito è riportata una tabella in cui si evidenziano i risultati migliori.

REGRESSIONE LINEARE			SDA			RW	w	LR	# hid	Corr	MB	Epoche	ΔRMSE	ΔMAD	ΔR^2	SDA	
RMSE	MAD	R^2	RMSE	MAD	r^2	RMSE										RMSE_valid	MAD_valid
0,07596	0,04302	0,84584	0,07183	0,03226	0,88046	0,08205	5	0,01	50	0,2	10	35	0,00413	0,01076	0,03462	0,11009	0,05511
0,07596	0,04302	0,84584	0,07189	0,03214	0,88547	0,08205	5	0,01	180	0,2	10	10	0,00407	0,01088	0,03963	0,10954	0,05441
0,07596	0,04302	0,84584	0,07177	0,03208	0,88534	0,08205	5	0,01	240	0,2	10	9	0,00419	0,01094	0,03950	0,10939	0,05447
0,07596	0,04302	0,84584	0,07311	0,03264	0,88714	0,08205	5	0,01	500	0,2	10	16	0,00285	0,01038	0,04130	0,11045	0,05520
0,07705	0,04977	0,85986	0,07283	0,03623	0,89479	0,08205	15	0,05	60	0,3	40	28	0,00422	0,01354	0,03493	0,10425	0,05487
0,07705	0,04977	0,85986	0,07305	0,03567	0,90056	0,08205	15	0,05	120	0,3	40	36	0,00400	0,01410	0,04070	0,10318	0,05347
0,07705	0,04977	0,85986	0,07217	0,03544	0,90369	0,08205	15	0,05	200	0,3	40	24	0,00488	0,01433	0,04383	0,10358	0,05340
0,07791	0,05174	0,86071	0,07787	0,04115	0,89163	0,08205	30	0,05	60	0,3	40	36	0,00003	0,01059	0,03092	0,09900	0,05471
0,07791	0,05174	0,86071	0,07491	0,03839	0,89998	0,08205	30	0,05	120	0,3	40	50	0,00300	0,01335	0,03927	0,09904	0,05191
0,07791	0,05174	0,86071	0,07470	0,03896	0,90189	0,08205	30	0,05	200	0,3	40	16	0,00320	0,01278	0,04117	0,09999	0,05335
0,08140	0,05758	0,86718	0,07506	0,03879	0,90094	0,08205	60	0,05	200	0,3	10	62	0,00634	0,01879	0,03376	0,09698	0,04932
0,08140	0,05758	0,86718	0,07495	0,03948	0,90182	0,08205	60	0,05	300	0,3	10	56	0,00645	0,01811	0,03464	0,09671	0,05002
0,08140	0,05758	0,86718	0,07530	0,03992	0,90261	0,08205	60	0,05	400	0,3	10	27	0,00610	0,01766	0,03543	0,09801	0,05074
0,08140	0,05758	0,86718	0,07396	0,03798	0,91223	0,08205	60	0,05	1500	0,3	15	8	0,00744	0,01960	0,04505	0,10098	0,05416
0,06697	0,03453	0,89130	0,06154	0,02825	0,91906	0,08205	90	0,01	500	0,2	10	26	0,00543	0,00628	0,02776	0,09538	0,04861
0,06690	0,03499	0,89349	0,06247	0,02868	0,91422	0,08205	100	0,05	200	0,3	10	9	0,00443	0,00631	0,02073	0,09426	0,04804
0,06690	0,03499	0,89349	0,06307	0,02994	0,91467	0,08205	100	0,05	300	0,3	10	6	0,00383	0,00506	0,02118	0,09503	0,04719
0,06690	0,03499	0,89349	0,06289	0,03005	0,91645	0,08205	100	0,05	400	0,3	10	6	0,00402	0,00494	0,02296	0,09503	0,04867
0,06690	0,03499	0,89349	0,06180	0,02850	0,93121	0,08205	100	0,05	1500	0,3	15	6	0,00511	0,00650	0,03772	0,10253	0,05308
0,06685	0,03534	0,89421	0,06221	0,02848	0,92293	0,08205	120	0,1	500	0,2	10	13	0,00465	0,00686	0,02872	0,09684	0,04837
0,06685	0,03534	0,89421	0,06387	0,03134	0,91754	0,08205	120	0,01	500	0,2	10	6	0,00298	0,00400	0,02332	0,09736	0,05008

Tabella 4.2 - Risultati test con denoising auto-encoder (in giallo i risultati migliori)

4.3.1 Best RMSE e MAD sul validation set

Introducendo una percentuale di corruzione, i valori del RMSE e del MAD sul *validation set* tendono a diminuire, anche se di uno scarto molto ridotto. Il valore più basso per il RMSE sul *validation set* è raggiunto nel test dove la rete è configurata con 100 *feature* in input, 200 neuroni nello strato nascosto, *learning rate* 0,05, *mini batch* di 10 elementi e una percentuale di corruzione del 30% e vale 0,09426. Si noti che la stessa configurazione senza corruzione dava un valore di

0,09539, cioè peggiore. Il MAD migliore sul *validation set* invece si ottiene con 100 *feature* in input, 300 neuroni nello strato nascosto, *learning rate* 0,05, *mini batch* di 10 elementi e una percentuale di corruzione del 30% e vale 0,04719. La stessa configurazione senza la corruzione dell'input dava un valore di 0,04751, che pur essendo il migliore di tutti i test effettuati con *auto-encoder* senza corruzione, risulta peggiore.

4.3.2 Best R^2

Anche il coefficiente di determinazione R^2 migliora sensibilmente se l'input è pretrattato tramite un *denoising auto-encoder*. Il valore migliore infatti risulta 0,93121 nella configurazione con 100 *feature* in input, 1500 neuroni nello strato nascosto, *learning rate* 0,1, *mini batch* di 15 elementi ed il 30% di corruzione (migliore rispetto a 0,92920 della stessa configurazione senza corruzione).

4.3.3 Best ΔR^2 e ΔRMSE

Se si valutano gli incrementi che la rete neurale ha favorito rispetto alla regressione lineare multipla applicata da sola, la configurazione che ha funzionato meglio è indubbiamente quella con 60 *feature* in input, 1500 neuroni nello strato nascosto, *learning rate* 0,05, *mini batch* di 15 elementi e un tasso di corruzione del 30%. Infatti, trattare i dati con un *denoising auto-encoder* così costruito consente di avere un incremento ΔR^2 di 0,04505 e un incremento ΔRMSE di 0,00744 (valori simili alle migliori configurazioni senza corruzione).

4.4 Stacked denoising auto-encoder

Per creare un'architettura profonda, si fa seguire ad un primo livello di neuroni nascosti un secondo strato, che riceve in input l'output del precedente, creando così uno *stacked denoising auto-encoder*. In generale, si è notato che gli esempi migliori sono quelli in cui il livello successivo ha un numero di neuroni minore o uguale di quello precedente. In determinate situazioni, i risultati di una rete profonda di questo tipo sono molto interessanti. Nella seguente tabella vengono evidenziati i più promettenti.

REGRESSIONE LINEARE			SDA			RW	w	LR	# hid	Corr	MB	Epoche	ΔRMSE	ΔMAD	ΔR^2	SDA	
RMSE	MAD	R^2	RMSE	MAD	r^2	RMSE										RMSE_valid	MAD_valid
0,07836	0,05296	0,86276	0,07573	0,03948	0,90029	0,08205	40	0,05	80-160	0,0-0,0	15	64	0,00263	0,01348	0,03753	0,09804	0,05041
0,07006	0,03961	0,88463	0,06878	0,03420	0,89766	0,08205	80	0,05	80-160	0,0-0,0	15	18	0,00129	0,00541	0,01303	0,09914	0,05197
0,07006	0,03961	0,88463	0,06802	0,03504	0,90131	0,08205	80	0,05	80-160	0,3-0,0	15	30	0,00204	0,00457	0,01668	0,09833	0,05006
0,07006	0,03961	0,88463	0,06451	0,03136	0,91123	0,08205	80	0,05	240-480	0,0-0,0	15	23	0,00555	0,00825	0,02660	0,09806	0,05011
0,08140	0,05758	0,86718	0,07460	0,03886	0,90357	0,08205	60	0,05	400-1000	0,3-0,0	15	6	0,00680	0,01872	0,03639	0,09764	0,05066
0,08140	0,05758	0,86718	0,07446	0,03829	0,90554	0,08205	60	0,05	600-1200	0,3-0,0	15	6	0,00694	0,01929	0,03836	0,09766	0,05133
0,08140	0,05758	0,86718	0,07232	0,03581	0,91772	0,08205	60	0,05	1000-500	0,3-0,0	15	6	0,00908	0,02177	0,05053	0,09786	0,05071
0,08140	0,05758	0,86718	0,07418	0,03812	0,92530	0,08205	60	0,05	1500-1500	0,3-0,0	15	12	0,00722	0,01946	0,05812	0,09904	0,05210
0,06690	0,03499	0,89349	0,06301	0,03006	0,91598	0,08205	100	0,05	400-1000	0,3-0,0	15	6	0,00390	0,00494	0,02249	0,09507	0,04888
0,06690	0,03499	0,89349	0,06142	0,02855	0,92345	0,08205	100	0,05	600-1200	0,3-0,0	15	6	0,00548	0,00645	0,02996	0,09625	0,04858
0,06690	0,03499	0,89349	0,06412	0,03107	0,91241	0,08205	100	0,1	1000-300	0,0-0,0	15	73	0,00278	0,00392	0,01891	0,09545	0,04759
0,06690	0,03499	0,89349	0,06057	0,02763	0,92034	0,08205	100	0,1	1000-300	0,3-0,0	15	7	0,00633	0,00736	0,02685	0,09535	0,04724
0,06690	0,03499	0,89349	0,06216	0,02756	0,93581	0,08205	100	0,05	1000-500	0,3-0,0	15	7	0,00474	0,00743	0,04231	0,09675	0,04803
0,06690	0,03499	0,89349	0,06153	0,02860	0,92495	0,08205	100	0,05	1000-750	0,0-0,0	15	6	0,00537	0,00640	0,03146	0,09780	0,05034
0,06690	0,03499	0,89349	0,06174	0,02867	0,92550	0,08205	100	0,05	1000-750	0,3-0,0	15	5	0,00516	0,00632	0,03201	0,09846	0,05040
0,06690	0,03499	0,89349	0,06210	0,03045	0,91873	0,08205	100	0,05	1500-500	0,0-0,0	15	22	0,00480	0,00454	0,02524	0,09723	0,04964
0,06690	0,03499	0,89349	0,06185	0,02906	0,92266	0,08205	100	0,05	1500-500	0,3-0,0	15	6	0,00505	0,00593	0,02917	0,09658	0,04824
0,06690	0,03499	0,89349	0,06476	0,03050	0,94576	0,08205	100	0,05	1500-1500	0,3-0,0	15	11	0,00214	0,00450	0,05227	0,10141	0,05099
0,06685	0,03534	0,89421	0,06386	0,03042	0,91345	0,08205	120	0,1	240-120	0,0-0,0	15	5	0,00299	0,00492	0,01924	0,09453	0,04749
0,06685	0,03534	0,89421	0,06329	0,02996	0,91637	0,08205	120	0,05	240-120	0,3-0,0	15	5	0,00357	0,00538	0,02215	0,09557	0,04956
0,06685	0,03534	0,89421	0,06357	0,03002	0,91541	0,08205	120	0,1	240-120	0,3-0,0	15	5	0,00329	0,00532	0,02120	0,09561	0,04973
0,06685	0,03534	0,89421	0,06526	0,03168	0,91207	0,08205	120	0,05	240-120	0,0-0,0	15	5	0,00160	0,00366	0,01785	0,09606	0,04971

Tabella 4.3 - Risultati test con stacked denoising auto-encoder (in giallo i risultati migliori)

4.4.1 Best RMSE e MAD sul validation set

Il valore più basso del RMSE corrisponde alla configurazione in cui la rete ha 120 *feature* in input, un primo strato nascosto di 240 neuroni

seguito da un secondo strato nascosto di 120 neuroni. In questo caso, l'input non viene corrotto, il tasso di apprendimento è fissato a 0,1 e il *mini batch* è di 15 elementi. Questo modello ha un RMSE sul *validation set* di 0,09453, più alto rispetto al migliore valore ottenuto con un *singolo denoising auto-encoder* ma minore di quello che si ha con l'*auto-encoder* senza corruzione.

4.4.2 Best R^2

Il risultato da sottolineare per quanto riguarda l'impiego dell'architettura su più livelli è sicuramente il valore di R^2 della regressione lineare che si è ottenuto durante questi test. Una rete con 100 *feature* in ingresso, 1500 neuroni nel primo strato nascosto e altrettanti nel secondo (*learning rate* = 0,05, *mini batch* = 15, 30% di corruzione nel primo livello, 0% nel secondo) ha infatti permesso di ottenere il valore di 0,94576, il migliore in assoluto.

4.4.3 Best ΔR^2 e ΔRMSE

Per quanto riguarda il guadagno su R^2 e sul RMSE, i risultati migliori si sono ottenuti rispettivamente grazie ad una rete con 60 *feature* in input, 1500 neuroni al primo strato e 1500 al secondo (*learning rate* = 0,05, *mini batch* = 15, 30% di corruzione al primo livello e 0% al secondo) con un $\Delta R^2 = 0{,}05812$ e ad una rete con 60 *feature* in input, 1000 neuroni al primo strato e 500 al secondo (*learning rate* = 0,05, *mini batch* = 15, 30% di corruzione al primo livello e 0% al secondo) con un ΔRMSE = 0,00908.

4.5 Comparazione tra i modelli

Nella seguente tabella vengono riassunti i valori migliori ottenuti con i modelli impiegati.

	RMSE_valid	MAD_valid	R^2	Δr^2	ΔRMSE
Auto-encoder	0,09475	0,04751	0,92920	0,04525	0,00791
D. auto-encoder	0,09426	0,04719	0,93121	0,04505	0,00744
S.D. auto-encoder	0,09453	0,04724	0,94576	0,05812	0,00908

Tabella 4.4 - Confronto tra i modelli (in giallo i risultati migliori)

Si può notare come, per quanto concerne gli esempi trattati, la rete neurale a più livelli ottenga prestazioni migliori rispetto ad un'architettura *shallow* relativamente al calcolo dei valori di R^2, ΔR^2 e ΔRMSE. Inoltre, l'introduzione della corruzione al primo strato e l'impiego del *denoising auto-encoder* permette di ottenere ottimi risultati sull'errore quadratico medio dopo che la rete ha fissato i pesi durante la fase di training.

4.6 Grafici

Di seguito sono riportati un paio di esempi dell'andamento dei valori reali (in blu) e di quelli predetti (in verde) in corrispondenza dell'ultima settimana del *test set*, di tutti i modelli impiegati durante i confronti: il *random walk*, la regressione lineare multipla e la regressione lineare multipla dopo aver trattato i dati con la rete neurale.

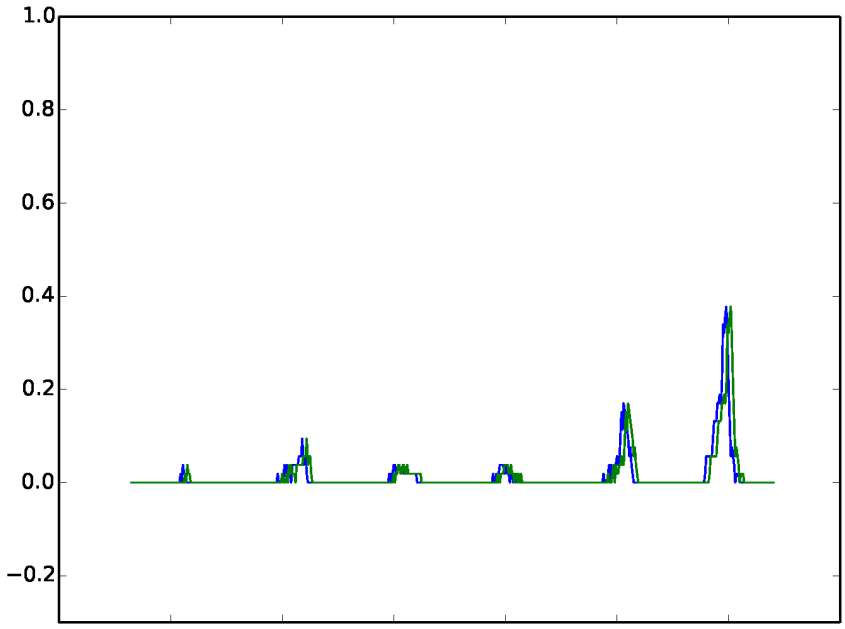

Figura 4.1 - Previsione random walk

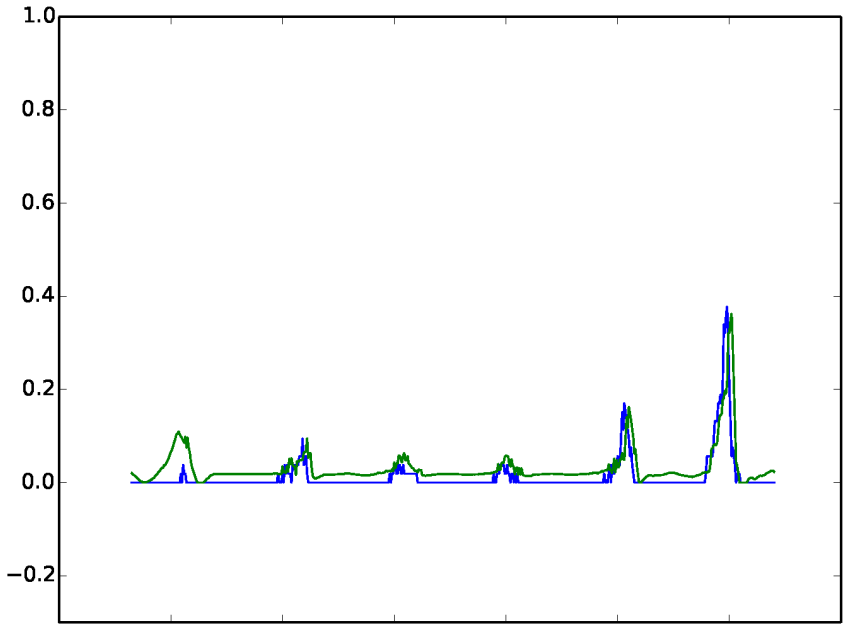

Figura 4.2 - Previsione reg lin con 100 feature

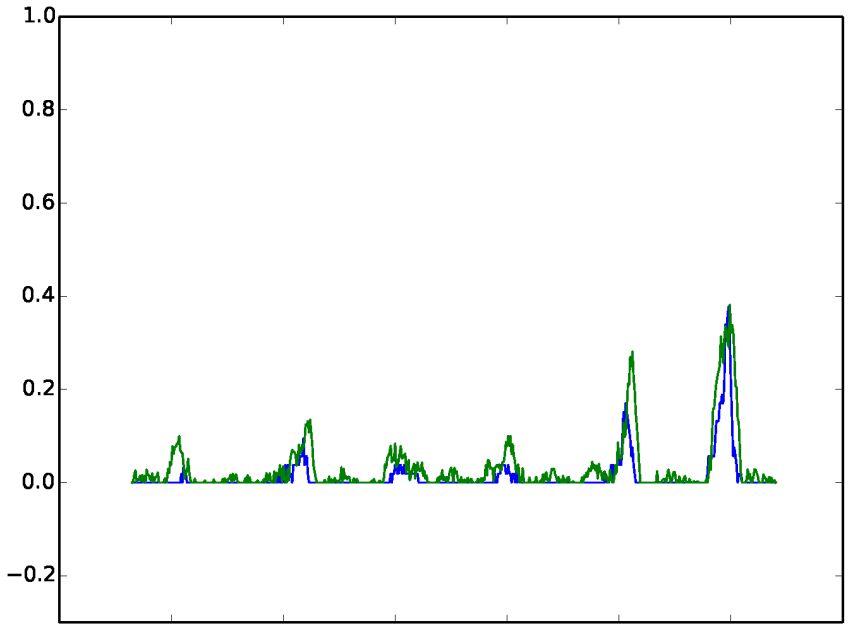

Figura 4.3 - Previsione SDA 100-1500-1500, corr 0,3-0,0

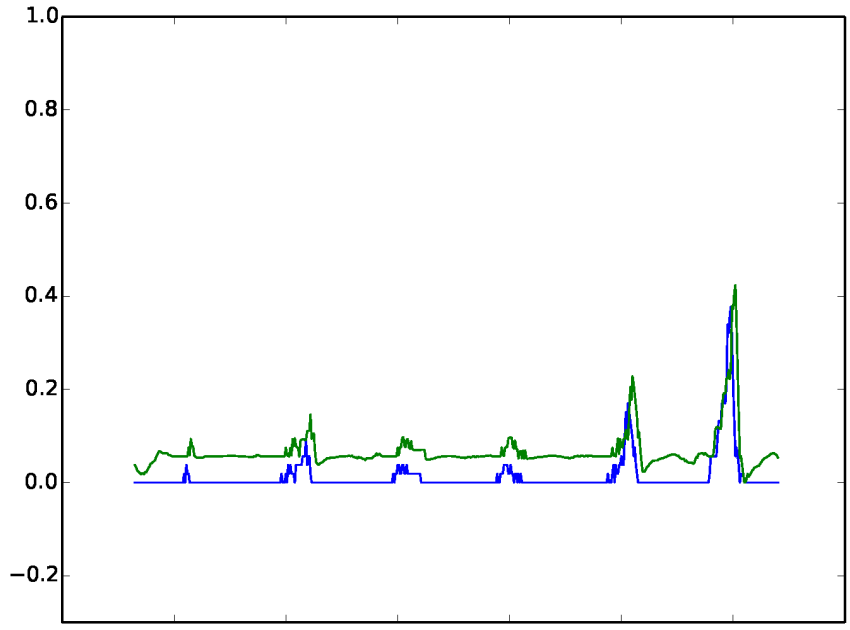

Figura 4.4 - Previsione regressione lineare con 60 feature

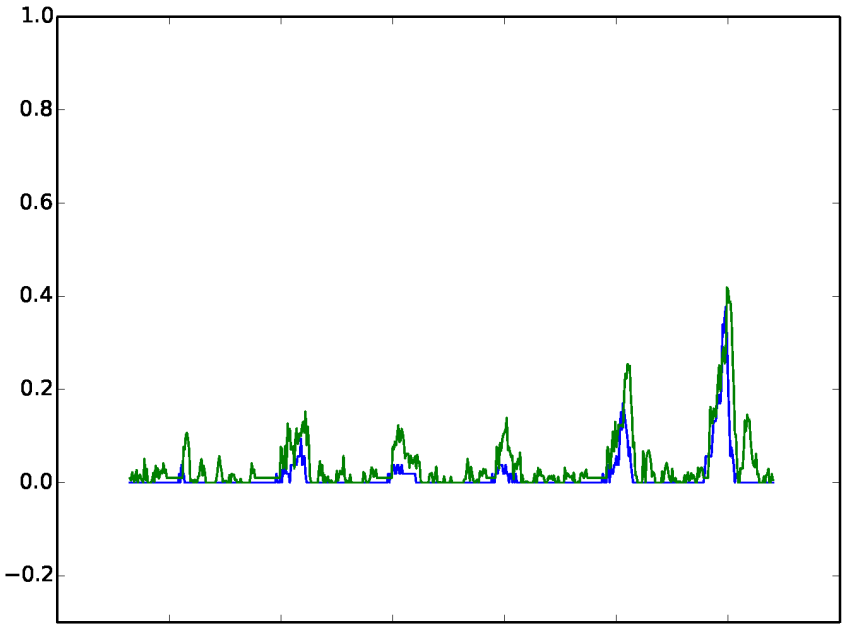

Figura 4.5 - Previsione SDA 60-1500-1500, corr 0,3-0,0

5. CONCLUSIONI

Dopo aver portato a termine un consistente numero di test, alcuni dei risultati emersi possono considerarsi interessanti. Innanzitutto, l'applicazione prima del trattamento dei dati di un'architettura intelligente, che sia o meno di tipo profondo, migliora in generale l'esito della previsione effettuata dalla regressione lineare nei termini evidenziati nel capitolo precedente. In secondo luogo, è da sottolineare come, in questo tipo di problemi, l'aggiunta di un secondo livello di profondità possa essere utile per un ulteriore miglioramento delle prestazioni.

Stabilita quindi l'efficacia della soluzione proposta, è doveroso soffermarsi a riflettere su quali possano essere gli sviluppi da qui in avanti. Non si deve dimenticare che ciò che si è ottenuto durante questo studio è frutto di sperimentazioni empiriche, nelle quali i parametri sono stati scelti senza una vera e propria teoria a supporto. Soltanto una vasta serie di prove (in aggiunta a quelle già svolte) potrà consolidare ulteriormente i risultati ottenuti, che in ogni caso non saranno mai da intendersi in senso assoluto.

Le direzioni verso cui indirizzare la ricerca in futuro sono molteplici, e probabilmente porteranno ad una messa a punto più raffinata dei parametri che garantirà ulteriori miglioramenti.

Come prima cosa, sarebbe interessante introdurre ulteriori variabili esplicative per quanto riguarda la regressione lineare, qualora fossero disponibili i dati relativi. Ad esempio, nel caso trattato della predizione dei valori di produzione di energia elettrica da fotovoltaico, si

potrebbero impiegare le informazioni sull'irraggiamento e sulla temperatura esterna, parametri che è ragionevole pensare possano influenzare il valore da predire.

Proseguendo, si potrebbe implementare un meccanismo per effettuare il *fine-tuning* della rete che, basandosi sull'errore misurato sulla regressione lineare a posteriori, aggiusti i pesi ad ogni iterazione.

Un altro sviluppo interessante riguarderebbe lo studio della robustezza della rete se si decide di allungare l'orizzonte temporale scelto per la predizione. Nel caso trattato, il lasso di tempo a cui l'azienda era interessata era relativamente breve (un'ora): si potrebbero eseguire dei test semplicemente aumentandolo, oppure usare le previsioni in maniera ricorsiva, sfruttando i valori predetti per compiere nuove previsioni a più ampio raggio.

In definitiva, dal momento che il modello è stato costruito lasciando ampio spazio di personalizzazione per quanto riguarda la scelta dei parametri, e che costituisce un'architettura generale per l'analisi e la previsione di un qualsiasi tipo di serie temporale, ci sarà la possibilità di estendere lo studio ad altri impianti simili oppure, sempre all'interno dell'ambito considerato, con metriche di misurazione diverse come ad esempio il consumo di energia elettrica.

BIBLIOGRAFIA

Estela Bee Dagum – Analisi delle serie storiche: modellistica, previsione e scomposizione, Milano, Springer (2002)

Douglas C. Montgomery, Cheryl L. Jennings, Murat Kulahci – Introduction to time series analysis and forecasting, Hoboken N.J., John Wiley (2008)

David M. Levine, Timothy C. Krehbiel, Mark L. Berenson – Statistica, 5° edizione, Milano, Pearson (2010)

David M. Levine, Timothy C. Krehbiel, Mark L. Berenson – Statistica, 2° edizione, Milano, Apogeo (2006) (capitolo 13, online all'indirizzo: http://www.apogeonline.com/2006/libri/88-503-2357-3/ebook/2357-cap13.pdf)

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel – Time series analysis, 3° ed, Englewood Cliffs N.J., Prentice Hall, 1994

Domenico Piccolo – Introduzione all'analisi delle serie storiche, Roma, La nuova Italia scientifica, 1990

Flavio Beretta, Filippo De Carlo, Vito Introna, Daniele Saccardi – Progettare e gestire l'efficienza energetica, Milano, McGraw-Hill, 2012

M. Längkvist, L. Karlsson, A. Loutfi – A review of unsupervised feature learning and deep learning for time series modeling. Pattern

Recognition Letters, 42(1): 11-24 (disponibile online all'indirizzo: http://dx.doi.org/10.1016/j.patrec.2014.01.008)

Deep learning tutorial, version 1.0, 2015 (disponibile online all'indirizzo: http://deeplearning.net/tutorial/deeplearning.pdf)

Li Deng – Three classes of deep learning architectures and their applications: a tutorial survey, APSIPA Transactions on Signal and Information Processing, 2012

Li Deng, Dong Yu – Deep Learning, Methods and Applications (originariamente pubblicato come Foundations and Trends in Signal Processing, Volume 7, Issues 3-4, pp 197-387), 2014

Yoshua Bengio – Learning Deep Architechture for AI, Foundations and Trends in Machine Learning 1(2), pp 1-127, 2009

Geoffrey E. Hinton, R.R. Slakhutdinov – Reducing the Dimensionality of Data with Neural Networks, Science, Volume 313, pp 504-507, 2006

Geoffrey E. Hinton – A Practical Guide to Training Restricted Boltzmann Machines, Version 1, 2010

Geoffrey E. Hinton – Learning multiple layers of representation, Trends in cognitive sciences, 11(10), pp 428-434, 2007

Ian Goodfellow, Yoshua Bengio, Aaron Courville – Deep Learning – An MIT Press book in preparation (disponibile online all'indirizzo: http://www.deeplearningbook.org)

Yann LeCun, Yoshua Bengio, Geoffrey E. Hinton – Deep Learning, Nature 521, pp 436-444, 2015

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol – Stacked Denoising Auto-encoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, Journal of Machine Learning Research 11, pp 3371-3408, 2010

https://www.python.org

http://deeplearning.net/software/theano/

http://deeplearning.net