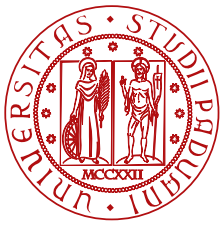


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

Reti Neurali e il fenomeno della doppia discesa: scenari innovativi per apprendere sistemi dinamici dai dati

Relatore
Prof. Pillonetto Gianluigi

Laureando
Marigo Andrea

ANNO ACCADEMICO 2023-2024

Data di laurea 27/09/2024

A Mamma e Papà,
che mi hanno sempre stimolato
a raggiungere i miei obiettivi.

Sommario

Le reti neurali hanno rivoluzionato numerosi campi dell'intelligenza artificiale grazie alla loro capacità di apprendimento automatico da grandi quantità di dati. Tuttavia, la complessità strutturale delle reti neurali e la loro natura non lineare rendono difficile comprendere a fondo il loro comportamento, soprattutto in termini di generalizzazione.

Un fenomeno di particolare interesse emerso recentemente è quello delle "curve a doppia discesa". Questo fenomeno descrive come l'errore di generalizzazione, oltre alla soglia di interpolazione sui dati di allenamento, subisca una ulteriore discesa. Questo comportamento sostituisce quello tradizionalmente conosciuto a forma di U, in cui i modelli con complessità molto elevata erano solitamente evitati. Infatti, questo fenomeno contraddice i principi classici su cui si fondava l'identificazione di sistemi, come il principio di Occam.

In questa tesi, si introduce il concetto di modello e il problema dell'identificazione di un sistema dinamico, trattando il principale procedimento di risoluzione. In seguito, si analizzano le reti neurali artificiali, approfondendo le strutture più comuni e il loro processo di apprendimento. Successivamente, si esamina il problema della regressione e come la flessibilità delle reti neurali viene sfruttata per ottimizzarlo, con particolare attenzione sui modelli basati sull'energia. Infine, è discusso il fenomeno delle curve a doppia discesa, con la presentazione di studi teorici e un'evidenza sperimentale.

Indice

Capitolo 1

Modellare sistemi dinamici

L'elemento fondamentale su cui si fondano la teoria dei sistemi e del controllo sono i sistemi dinamici. Essi sono caratterizzati dal cambiamento dinamico nel tempo delle loro uscite in risposta alle variazioni degli ingressi. Qualsiasi fenomeno fisico o naturale può essere considerato un sistema dinamico, poiché in risposta a vari fattori, produce un particolare comportamento. Alcuni esempi sono: il moto di un pendolo, le interazioni tra differenti specie nello stesso ambiente e la regolazione della produzione di insulina nel corpo umano.

Lo studio dei sistemi dinamici è essenziale per comprendere e controllare il comportamento di sistemi complessi nel mondo reale. Una rappresentazione matematica di un fenomeno permette analizzare le relazioni tra ingressi e uscite del sistema, e quindi di predire il suo comportamento futuro in condizioni non sperimentate. Queste strutture prendono il nome di modelli e trovano applicazioni in una vasta gamma di discipline, dalla robotica all'economia, dall'ingegneria dei trasporti alla biologia.

Il campo dell'ingegneria che si occupa di costruire un modello di un sistema dinamico a partire dai dati ottenuti da sue misurazioni è noto come *system identification*. Per ottenere ciò, è necessaria la selezione di un modello che rappresenti la dinamica del sistema, e il successivo *tuning* dei parametri per mezzo di algoritmi di *machine learning*.

1.1 Modello

Un modello è una rappresentazione matematica di un fenomeno, del quale riproduce alcune caratteristiche o comportamenti fondamentali. Il modello raggiunge il suo scopo quando è in grado di predire correttamente uscite future del sistema. I modelli matematici permettono:

- una descrizione sintetica e non ambigua di fenomeni complessi;

- partendo da qualsiasi condizioni iniziali impostate, la simulazione al computer del comportamento del sistema;

- l'analisi delle proprietà e risposte di un fenomeno.

La fase di progettazione di un modello comincia con l'analisi del sistema. Infatti una conoscenza iniziale del fenomeno, attraverso leggi fisiche o parametri noti, permette di formare già inizialmente una buona struttura per il modello matematico. A seconda della quantità e qualità di conoscenze pregresse riguardanti il fenomeno, si distinguono tre differenti approcci per costruire il modello:

(a) **Scatola trasparente**: se si conoscono le leggi che regolano il fenomeno ed il valore esatto dei parametri coinvolti, allora il modello deriva dalla risoluzione di queste leggi.

(b) **Scatola grigia**: se si conoscono le leggi ma non i parametri del sistema, allora il modello deriva comunque dalla risoluzione delle equazioni della dinamica, ma la stima dei parametri necessiterà anche di misurazioni del sistema dinamico e conseguentemente di algoritmi di *machine learning*.

(c) **Scatola nera**: se non si conosce nulla del fenomeno, allora la realizzazione del modello sarà complessa, e comprenderà per esempio l'uso di reti neurali e avanzati algoritmi di ottimizzazione.

1.1.1 Tipologie di modelli di stato

Ogni modello può essere espresso attraverso le variabili di stato $x(t)$, che rappresentano le condizioni del sistema in un dato istante di tempo. Esse tengono traccia delle informazioni rilevanti per l'evoluzione del sistema nel tempo. Ad esempio, le variabili di stato potrebbero includere parametri come la pressione atmosferica, la temperatura, la velocità del vento, e altre specifiche quantità. Esse sono utilizzate per predire il comportamento futuro del sistema sulla base delle relazioni dinamiche tra di esse.

Date le variabili di stato $x(t) = [x_1(t), x_2(t), ..., x_n(t)]^T$ e l'ingresso $u(t)$, un sistema è univocamente definito dalle seguenti equazioni:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right) \\ y(t) = h\left(x(t), u(t)\right) \end{cases}$$

a tempo continuo $t \in \mathbb{R}$

$$\begin{cases} x(t+1) = f\left(x(t), u(t)\right) \\ y(t) = h\left(x(t), u(t)\right) \end{cases}$$

a tempo discreto $t \in \mathbb{Z}$

Dunque, il cambiamento delle variabili di stato è determinato dalla funzione f, mentre l'uscita del sistema è influenzata direttamente da h e indirettamente da f.

Un'altra importante distinzione è tra modelli lineari, in cui f e h sono combinazioni lineari dei loro argomenti, e non lineari, dove f e h sono funzioni non lineari. In questo secondo caso, la trattazione risulta più difficile, ma il modello è solitamente in grado di approssimare sistemi molto più complessi.

1.2 Il problema dell'identificazione di sistemi

Si consideri ora un contesto in cui manca la conoscenza dettagliata della dinamica, delle costanti e dei parametri associati a un sistema. In questo scenario, si affronta il problema dell'identificazione trattandolo come un sistema a scatola nera, dove l'approccio si basa esclusivamente sull'osservazione dei suoi input e output senza una comprensione diretta dei meccanismi interni.

L'approccio classico per affrontare il problema si fonda sui principi della statistica matematica. Queste tecniche, basate sui metodi di minimizzazione dell'errore di predizione, si concentrano sulla determinazione dell'ordine del modello e sulla scelta di diverse strutture che dipendono da un vettore di parametri sconosciuto. La dimensione di questo vettore determina direttamente la complessità del modello.

Selezionare la struttura del modello più appropriata è fondamentale nella procedura di identificazione e si fa di solito seguendo il principio del rasoio di Occam. Tale principio afferma che tra più spiegazioni plausibili per un fenomeno, la più semplice tende ad essere la corretta. Ciò significa che non si dovrebbe introdurre complessità o assumere più fattori di quanto sia strettamente necessario per spiegare un dato fenomeno. Questo principio deriva dall'idea che la natura tende a essere parsimoniosa e che le spiegazioni più semplici sono spesso più verosimili e più facili da verificare. Tuttavia, ci possono essere casi in cui una spiegazione più complessa è necessaria per rendere conto di tutti i dati disponibili, quindi il principio di Occam dovrebbe essere considerato come un principio guida, non come una regola assoluta.

1.2.1 Procedimento classico

Il procedimento classicamente utilizzato per identificare un modello di un sistema dinamico è composto di tre passaggi.

a) Selezione della famiglia di modelli

L'individuazione della struttura di modelli che potrebbe rappresentare al meglio il fenomeno è il primo passaggio e anche il più cruciale per la costruzione di un modello matematico a partire dai dati. Ciascun modello induce una mappa $\mathrm{g}(\cdot)$ dai dati di input-output $Z(t)$ osservati fino al tempo t, alla predizione della futura uscita $\hat{y}(t+1) = \mathrm{g}(\theta, Z(t))$. I parametri θ possono variare

all'interno di un range $D_{\mathcal{M}}$ che quindi definisce una famiglia $\mathcal{M}$ di possibili modelli tra cui scegliere quello che predice al meglio uscite future.

Inoltre, nella teoria classica, è presente la concezione di ordine discreto del modello. Una famiglia di modelli $\mathcal{M}_1$ si dice di ordine inferiore a $\mathcal{M}_2$ se il range di parametri $D_{\mathcal{M}_1}$ è contenuto in $D_{\mathcal{M}_2}$. Tale ordine è ad esempio il grado di un polinomio nella regressione polinomiale, oppure il numero di neuroni in una rete neurale artificiale.

La scelta riguardante la struttura del modello generalmente deve soddisfare le seguenti caratteristiche:

- **Flessibilità:** il modello deve essere in grado di descrivere al meglio dati differenti. Tale proprietà può essere ottenuta semplicemente aumentando l'ordine del modello.

- **Parsimonia:** bisogna evitare di inserire parametri non necessari, mantenendo l'ordine basso. Un modello troppo flessibile potrebbe essere molto influenzato dal rumore sui dati di allenamento.

b) Stima dei parametri

Una volta scelta la struttura, si stimano i parametri utilizzando i dati a disposizione. Il metodo classico, e anche il più intuitivo, consiste nel determinare θ in modo tale che le sue predizioni differiscano il meno possibile dai dati del sistema. Si scelgono cioè i parametri che minimizzano l'errore tra gli output osservati e quelli predetti dal modello. Supponendo di avere dei dati di coppie ingresso-uscita $(x(t), y(t))$ del fenomeno per N istanti di tempo, si definisce l'errore:

$$V_N(\theta, Z) = \frac{1}{N} \sum_{t=1}^{N} \|y(t) - \hat{y}(t|\theta)\|^2 \tag{1.1}$$

Dunque la stima dei parametri, dato un insieme di dati $Z_e = [..., (x(i), y(i)), ...]$, si ottiene calcolando il valore dei parametri dove l'errore risulta minimo:

$$\hat{\theta}_N = \arg\min_{\theta \in D_\theta} V_N(\theta, Z_e) \tag{1.2}$$

Il valore minimo tra tutti i $V_N(\theta, Z_e)\ \forall\, \theta \in D_{\mathcal{M}}$ è chiamato *empirical fit* oppure errore di allenamento e si ottiene in questo modo:

$$V_{\text{emp}} = V_N(\hat{\theta}_N, Z_e) \tag{1.3}$$

c) Validazione

L'ultimo passaggio è quello della validazione, per mezzo della quale si decide se validare o falsificare le ipotesi inizialmente fatte sul modello.

Per applicarla è necessario disporre di un ulteriore insieme di dati Z_v, indipendente da quello usato per l'allenamento, dunque si calcola l'errore di validazione su questi dati:

$$V_{\text{valid}} = V_N(\hat{\theta}_N, Z_v) \tag{1.4}$$

Tale errore di validazione è un indicatore della capacità del modello di generalizzare in modo efficace. Se l'errore è troppo elevato per le richieste del problema, il modello non viene accettato. In questo caso, solitamente bisognerà modificare la struttura del modello per poi ripetere lo stesso procedimento.

Capitolo 2

Reti Neurali

La ricerca nell'ambito delle reti neurali artificiali trae ispirazione dalla straordinaria capacità del cervello umano di risolvere complesse attività in modo efficiente. Grazie ad una rete fortemente connessa di neuroni, esso è in grado di operare simultaneamente su molteplici livelli di elaborazione.

Si consideri ad esempio il processo di visione, che coinvolge il riconoscimento degli oggetti, la valutazione delle distanze e delle velocità. Il nostro cervello elabora queste informazioni in un intervallo di tempo straordinariamente breve, tra 100 e 200 millisecondi. Al contrario, un algoritmo convenzionale, seguendo un approccio sequenziale nell'esecuzione delle istruzioni, potrebbe richiedere molto più tempo, addirittura giorni, per compiere operazioni simili.

Le reti neurali artificiali, analogamente a quelle biologiche, sono costituite da elementi di base essenziali chiamati neuroni, che eseguono operazioni semplici. L'interconnessione tra questi elementi costitutivi permette di elaborare calcoli complessi sfruttando il contributo di ogni singolo neurone.

Inoltre, il cervello mostra una notevole flessibilità nel riorganizzare la sua struttura neurale per adattarsi meglio ai compiti specifici. Questa abilità è forse la più interessante dal punto di vista ingegneristico. Infatti, fornendo alla rete artificiale i dati relativi al mondo esterno, essa può imparare a risolvere un qualsiasi compito tramite elaborati algoritmi di apprendimento.

2.1 I vantaggi delle reti neurali

La potenza delle reti neurali artificiali risiede nella loro capacità di affrontare una vasta gamma di compiti, ispirandosi alle reti neurali presenti in natura. Nelle prime pagine del libro [5], viene fornita la seguente definizione formale di rete neurale, intesa come una macchina adattiva:

Def. *Una rete neurale è un processore distribuito massivamente in parallelo composto da unità di elaborazione semplici, che ha una propensione naturale per memorizzare conoscenze*

esperienziali e renderle disponibili per l'uso. Assomiglia al cervello per due aspetti:

1. *La rete acquisisce conoscenza dal suo ambiente attraverso un processo di apprendimento.*

2. *Le forze di connessione tra i neuroni, chiamate pesi sinaptici, vengono utilizzate per memorizzare la conoscenza acquisita.*

In seguito, la conoscenza appresa dalla rete è definita come segue:

Def. *Per conoscenza si intende l'informazione memorizzata o i modelli utilizzati da una persona o da una macchina per interpretare, prevedere e rispondere in modo appropriato al mondo esterno.*

La rete neurale può assimilare due tipi di conoscenza distinti. La prima è costituita dalle informazioni certe sull'ambiente nel suo stato iniziale, definite informazioni a priori. La seconda deriva dalle osservazioni del fenomeno, spesso ottenute tramite strumenti di misura suscettibili al rumore o ai disturbi. Le conoscenze acquisite possono essere applicate a dati nuovi e mai incontrati in precedenza. Grazie a questa capacità di generalizzazione, le reti neurali sono in grado di effettuare previsioni accurate e prendere decisioni affidabili su ingressi che non facevano parte del loro addestramento.

Le reti neurali dispongono delle seguenti caratteristiche:

- **Non linearità:** questo attributo è particolarmente rilevante quando si tratta di modellare sistemi dinamici molto complessi. Ad esempio, fenomeni fisici come la turbolenza nei fluidi, i valori di tensione e corrente nei circuiti elettrici e il comportamento dei materiali sotto stress seguono leggi non lineari.

- **Mappatura Input-Output:** la rete neurale, in seguito al processo di apprendimento, costruisce una mappa che lega i valori di ingresso a quelli di uscita. Tale mappatura è progettata in modo da essere valida per qualsiasi ingresso, anche quelli non inclusi nell'insieme di addestramento.

- **Adattabilità:** le reti neurali allenate per operare in un determinato ambiente possono essere facilmente riaddestrate per gestire cambiamenti minori nelle condizioni del sistema. Questa caratteristica le rende utili anche nei sistemi di controllo. Tuttavia, l'adattabilità non garantisce necessariamente robustezza. La costante di tempo con cui la rete neurale si adatta deve essere sufficientemente lunga da consentire di sopprimere disturbi nel sistema, ma anche sufficientemente breve da consentire una risposta efficace a cambiamenti significativi.

- **Informazioni di contesto:** le informazioni contestuali sono gestite in modo intrinseco grazie alla struttura e allo stato di attivazione della rete neurale. Ogni neurone all'interno della rete può essere influenzato dall'attività globale di tutti gli altri neuroni. Questo significa che la rete neurale è in grado di considerare dinamicamente il contesto delle informazioni, poiché ogni neurone risponde non solo ai suoi input diretti, ma anche all'interazione complessa con gli altri neuroni nella rete.

2.2 Struttura di una rete neurale

Una rete neurale artificiale ha come obiettivo l'emulazione dei processi cerebrali umani. Per fare ciò anche la struttura stessa deve assomigliare a quella naturale di un cervello biologico. Proprio come quest'ultimo è una rete intricata di cellule interconnesse capaci di trasmettere segnali elettrici e svolgere semplici operazioni, una rete neurale artificiale è costituita da un numero predeterminato di neuroni artificiali collegati tra loro tramite canali virtuali. Questi neuroni, organizzati in strati e dotati di componenti lineari e non lineari, elaborano le informazioni attraverso operazioni semplici per produrre un output, che può essere sia l'uscita finale del modello che l'ingresso per neuroni successivi.

2.2.1 Modello di un neurone

Ogni neurone artificiale deve essere in grado di ricevere in ingresso diversi segnali, compiere un elaborazione e restituire un uscita. In Figura 2.1 è rappresentata la sua struttura.

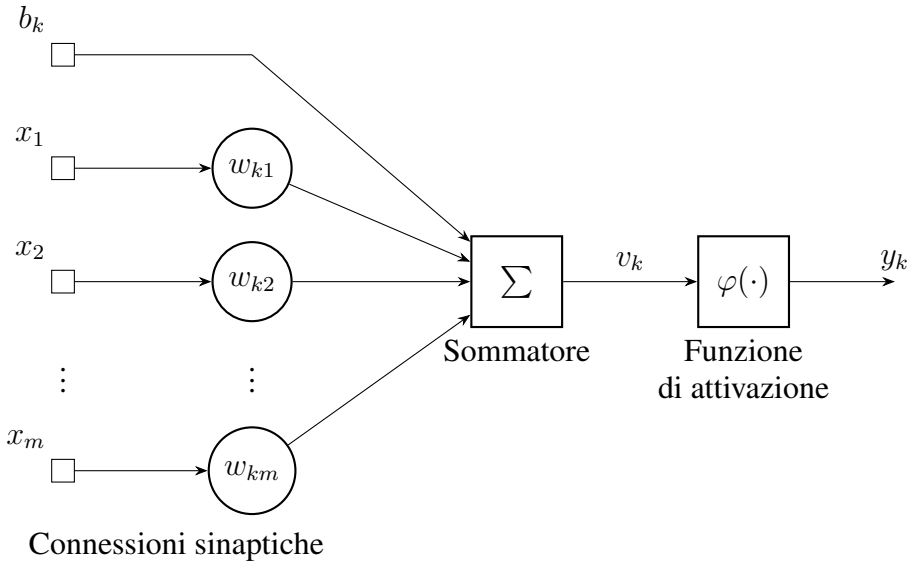


Figura 2.1: Rappresentazione schematica di un neurone artificiale.

Un neurone è dunque composto di tre elementi:

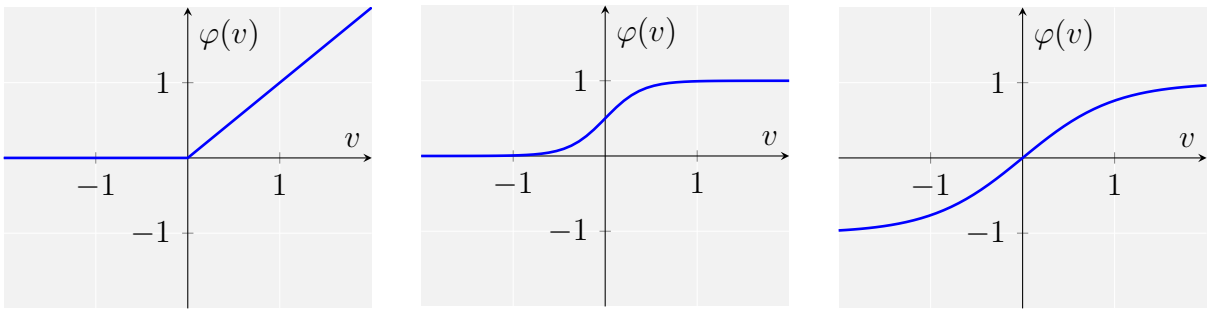
1. **Un insieme di connessioni sinaptiche.** Ogni segnale di ingresso x_i è moltiplicato dal peso w_{ki} della sinapsi. Tali pesi possono essere sia positivi che negativi. È anche presente un valore costante b_k detto *bias*, specifico del neurone k, che non costituisce un ingresso ma influenza l'uscita.
2. **Un sommatore** che calcola la somma $b_k + \sum_{i=1}^{m} w_{ki} x_i$.
3. **Una funzione di attivazione**, il cui scopo è quello di limitare i valori di uscita all'interno di un range specifico.

Se consideriamo il *bias* come un ingresso $x_0 = 1$ che è moltiplicato per il peso $w_{k0} = b_k$, si ottengono le seguenti equazioni che regolano la relazione ingresso-uscita in un neurone.

$$v_k = \sum_{i=0}^{m} w_{ki} x_i \tag{2.1}$$

$$y_k = \varphi(v_k) \tag{2.2}$$

La funzione di attivazione limita l'uscita solitamente negli intervalli $[-1, 1]$ oppure $[0, 1]$. In Figura 2.2 sono presentati 3 esempi comunemente utilizzati di funzione di attivazione.



(a) $\mathrm{ReLU}(x) = \max(x, 0)$ **(b)** Sigmoide $\frac{1}{1+e^{-av}}$ **(c)** Tangente iperbolica $\tanh(v)$

Figura 2.2: Funzioni di attivazione.

2.2.2 Architettura Feedforward

Le reti neurali sono tipicamente costruite su diversi strati, dove il primo strato riceve l'input del sistema e l'ultimo strato produce l'output del modello. Aggiungere più livelli comporta un aumento nella complessità del modello e nella sua capacità di adattarsi ai dati.

Una rete con L livelli può essere vista come una composizione di funzioni: $f^L \circ ... \circ f^2 \circ f^1$. Dove $f^L : \mathbb{R}^{n_L} \mapsto \mathbb{R}^{n_{out}}$ è detto livello di uscita, mentre tutti i precedenti sono noti come *livelli*

nascosti $f^{\ell} : \mathbb{R}^{n_\ell} \mapsto \mathbb{R}^{n_{\ell+1}}$, perché la loro uscita influenza indirettamente l'uscita dell'intera rete. Dunque, un generico neurone nello strato ℓ riceve in ingresso gli output dei neuroni nello strato $\ell - 1$ e la sua uscita è utilizzata come ingresso per il livello successivo.

Inoltre, per la struttura dei neuroni precedentemente esposta, dato il vettore di ingressi z, è possibile esprimere in forma matriciale la funzione di un intero strato come segue:

$$f^{\ell}(z) = \varphi\left(W_\ell\, z + b_\ell\right) \tag{2.3}$$

dove la matrice W_ℓ contiene sull'i-sima riga i pesi sinaptici dell'i-simo neurone di quel livello. Mentre b_ℓ è il vettore colonna di tutti i termini di bias.

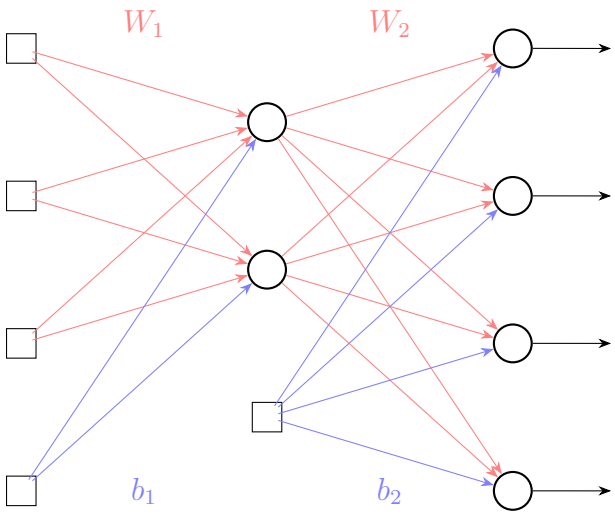


Figura 2.3: Grafico di una rete neurale artificiale.

Per esempio, data la rete neurale totalmente connessa a due strati rappresentato in Figura 2.3, l'uscita della rete è data da:

$$\hat{y}(t) = \varphi\left(W_2\, \varphi\left(W_1\, x(t) + b_1\right) + b_2\right) \tag{2.4}$$

In questo caso $f^1 : \mathbb{R}^3 \mapsto \mathbb{R}^2$, mentre $f^2 : \mathbb{R}^2 \mapsto \mathbb{R}^4$. Le matrici sinaptiche sono $W_1 \in \mathbb{R}^{2\times 3}$ e $W_2 \in \mathbb{R}^{4\times 2}$, mentre i vettori di *bias* sono $b_1 \in \mathbb{R}^2$ e $b_2 \in \mathbb{R}^4$.

I parametri che permettono a una rete neurale di risolvere differenti problemi sono i pesi sinaptici e i termini di *bias*. In un'architettura di rete *feedforward*, il numero di parametri è molto più elevato rispetto al numero di neuroni e cresce velocemente con l'aumentare dei neuroni per livello.

Nell'esempio di Figura 2.3 il vettore di parametri sconosciuti che si dovrà andare a stimare è $\theta = (W_1, b_1, W_2, b_2)$, per un totale di 20 incognite in questo caso.

La stima di questi parametri rappresenta quindi una sfida significativa per l'ottimizzazione.

2.3 Processo di apprendimento

La proprietà di principale importanza di una rete neurale è la sua capacità di apprendere delle informazioni dall'ambiente esterno riuscendo progressivamente a migliorare le proprie prestazioni. Tuttavia, la struttura precedentemente descritta non è sufficiente a questo scopo, ma è necessaria l'applicazione di un algoritmo detto di *learning*. È possibile definire il processo di apprendimento di una rete neurale [5] in questo modo:

Def. *L'apprendimento è un processo mediante il quale i parametri variabili di una rete neurale vengono adattati attraverso un processo di stimolazione da parte dell'ambiente in cui la rete è immersa. Il tipo di apprendimento è determinato dal modo in cui avvengono i cambiamenti dei parametri.*

Dunque questa definizione implica i seguenti passaggi:

1. La rete neurale deve essere stimolata. Ciò significa che necessita di dati appartenenti all'ambiente esterno.

2. Dall'analisi di questi dati, i parametri variabili della rete vengono ragionevolmente alterati.

3. In seguito a questi cambiamenti la rete neurale performerà differentemente, possibilmente meglio di prima.

2.3.1 Apprendimento tramite correzione dell'errore

La parte più complessa del procedimento di apprendimento è certamente la scelta della modalità di modifica dei parametri. Uno dei paradigmi principali che sono utilizzati in quest'ambito è quello chiamato *Error-Correction*.

Si suppone di avere a disposizione una serie di dati che associano al vettore di ingressi x le corrette uscite d che dovrebbe restituire la rete neurale. In questo caso è possibile definire un errore come la differenza tra l'uscita attesa e quella effettiva della rete. Dunque, le modifiche fatte sui parametri liberi sono proporzionali a questo errore.

Considero un neurone k appartenente allo strato di uscita. Sia $y_k(n)$ l'uscita effettiva di questo neurone dopo l'n-sima iterazione dell'algoritmo di aggiustamento dei pesi sinaptici. L'errore al nodo k è per definizione:

$$e_k(n) = d_k(n) - y_k(n) \tag{2.5}$$

Si definisce anche una funzione di costo $\mathcal{E}(n)$, che risulta essere l'energia istantanea dell'errore.

$$\mathcal{E}(n) = \frac{1}{2} e_k^2(n) \tag{2.6}$$

Gli aggiustamenti ad ogni iterazione dei pesi sinaptici $w_{ki}(n)$ associati agli ingressi $x_i(n)$ hanno l'obiettivo di minimizzare questa funzione di costo. Quindi, il procedimento di *learning* continua finché non si raggiunge una condizione di stabilità, ovvero $\mathcal{E}(n)$ non diminuisce più significativamente. La regola applicata per minimizzare la funzione di costo è detta *delta roule* ed è definita così:

$$\Delta w_{ki}(n) = \eta e_k(n) x_i(n) \tag{2.7}$$

dove η è una costante positiva che determina l'ampiezza di apprendimento tra uno step e il successivo. Dunque, i pesi sinaptici del nodo k vengono aggiornati in questo modo:

$$w_{ki}(n+1) = w_{ki}(n) + \Delta w_{ki}(n) \tag{2.8}$$

La tipologia di apprendimento *Error-Correction* è una sorta di sistema in retroazione. Dalla teoria del controllo sappiamo che la stabilità di tale sistema è particolarmente influenzata dalla scelta di η. Perciò il raggiungimento o meno dell'obiettivo di minimizzare la funzione di costo è direttamente collegato a questo parametro.

2.3.2 Algoritmo di propagazione all'indietro

Il procedimento descritto nella sezione precedente si applica esclusivamente ai neuroni dello strato di uscita. Tuttavia, le strutture di rete più complesse potrebbero disporre di alcuni livelli nascosti. È fondamentale che i pesi sinaptici dei neuroni in questi livelli vengano aggiornati in modo adeguato. Per fare ciò, si attribuisce ai neuroni nascosti una porzione dell'errore calcolato sulle uscite, utilizzando l'algoritmo di propagazione all'indietro.

Per questo algoritmo, la funzione di costo è definita come la somma dei quadrati degli errori sui neuroni dell'ultimo livello:

$$\mathcal{E}(n) = \frac{1}{2} \sum_{k \in L_{out}} e_k^2(n) \tag{2.9}$$

Si considera inizialmente un qualsiasi neurone $j \in L_{out}$. Le correzioni fatte sui pesi sinaptici $\Delta w_{ji}(n)$ sono proporzionali all'opposto della derivata parziale: $-\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$. In questo modo, i pesi sono aggiornati seguendo la decrescenza della funzione energia.

Per la regola della catena si ottiene:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.10}$$

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = \frac{\partial}{\partial e_j(n)}\left[\frac{1}{2}\sum_{k\,\in\, L_{out}} e_k^2(n)\right] = e_j(n) \qquad \frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)}\left[d_j(n) - y_j(n)\right] = -1$$
$$\frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\partial}{\partial v_j(n)}\left[\varphi(v_j(n))\right] = \varphi'(v_j(n)) \qquad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = \frac{\partial}{\partial w_{ji}(n)}\left[\sum_{i=0}^{m} w_{ji}(n)x_i(n)\right] = x_i(n) \tag{2.11}$$

Sostituendo 2.11 in 2.10 si ottiene:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'(v_j(n))x_i(n) \tag{2.12}$$

A questo punto la *delta roule* stabilisce le modifiche sui pesi.

$$\Delta w_{ji}(n) = \eta\delta_j(n)x_i(n) \tag{2.13}$$

dove $\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n)\varphi'(v_j(n))$ è detto gradiente locale, mentre η è il fattore di crescita strettamente positivo.

Sia ora j un neurone di uno strato più interno. Poiché $j \in L_{out-1}$ devo ricalcolare la seguente derivata parziale:

$$\begin{aligned}\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} &= \sum_{k\,\in\, L_{out}} e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)} = \sum_{k\,\in\, L_{out}} e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)} \\ &= \sum_{k\,\in\, L_{out}} e_k(n)\varphi'(v_k(n))w_{kj}(n) = -\sum_{k\,\in\, L_{out}} \delta_k(n)w_{kj}(n)\end{aligned} \tag{2.14}$$

Dunque, per il neurone interno j il gradiente locale è $\delta_j(n) = \varphi'(v_j(n))\sum_{k\,\in\, L_{out}} \delta_k(n)w_{kj}(n)$.

Ricorsivamente, per un neurone j appartenente ad un livello interno qualsiasi $\ell \neq L_{out}$, il gradiente locale si ottiene da:

$$\delta_j(n) = \varphi'(v_j(n))\sum_{i\,\in\,\ell} \delta_i(n)w_{ij}(n) \tag{2.15}$$

Riassumendo, l'algoritmo di propagazione all'indietro implementa le modifiche sui pesi sinaptici di neuroni a qualsiasi livello. La formula generale è la seguente, dove bisogna far attenzione ad inserire il corretto gradiente locale in base allo strato di appartenenza del neurone.

$$\begin{pmatrix}\text{Correzione}\\ \text{sui pesi}\\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\text{Fattore di}\\ \text{apprendimento}\\ \eta\end{pmatrix}\begin{pmatrix}\text{Gradiente}\\ \text{locale}\\ \delta_j(n)\end{pmatrix}\begin{pmatrix}\text{Segnale di ingresso } i\\ \text{al neurone } j\\ x_{ij}(n)\end{pmatrix}$$

2.4 Predizioni future a lungo termine

Fino a questo momento, non è stato definito in modo preciso il vettore di ingressi $x(t)$ della rete neurale. È importante notare che, nonostante venga denominato nello stesso modo, il vettore $x(t)$ è distinto dall'ingresso $u(t)$ del sistema dinamico. Infatti, la rete neurale potrebbe beneficiare dall'utilizzo di ulteriori segnali in input al tempo t.

Ad esempio, nelle strutture *feedforward* appena trattate, un metodo frequentemente applicato consiste nel definire $x(t)$ come una concatenazione di ingressi passati del sistema dinamico:

$$x(t) = [u(t), u(t-1), ..., u(t-m)] \tag{2.16}$$

In questo modo, l'uscita del modello dipende da una memoria di m ingressi precedenti. Questa architettura porta la rete neurale ad essere un modello chiamato *Nonlinear* FIR. Ciò significa che la risposta impulsiva del modello è a durata finita, ovvero si annulla in un tempo limitato, uguale alla lunghezza della memoria dell'ingresso.

È possibile arricchire ulteriormente il vettore di ingresso integrandolo con le uscite passate del modello:

$$x(t) = [y(t-1), ..., y(t-m), u(t), ..., u(t-m-d)] \tag{2.17}$$

dove m corrisponde ancora alla memoria della rete neurale, mentre d può essere utilizzato per considerare un ritardo nell'azione. Questa configurazione porta a un differente tipo di modello denominato *Nonlinear AutoRegressive with eXogenous input* (NARX). I modelli NARX combinano la dipendenza dalle uscite precedenti del modello (componente autoregressiva) con quella dagli ingressi esogeni passati, ovvero esterni al sistema e non influenzati dal sistema stesso.

2.4.1 Reti Neurali ricorrenti

Applicando le strategie descritte per l'apprendimento delle reti neurali con struttura *feedforward*, dato il vettore $x(t)$, è possibile predire le uscite future del sistema $\hat{y}(t) = f(x(t))$. Supponendo di conoscere gli ingressi al sistema $u(t+1), ..., u(t+k)$, per predire l'uscita al tempo $t+k$ si ottiene che:

- Con il modello FIR, conoscendo $x(t+k)$, è possibile ottenere $\hat{y}(t+k)$.
- Con il modello NARX non è possibile stimare $\hat{y}(t+k)$, poiché sarebbero necessarie le uscite del sistema $y(t), ..., y(t+k-1)$ per costruire l'ingresso $x(t+k)$.

La necessità di ottenere predizioni future più a lungo termine porta alla ricerca di una struttura differente per le reti neurali. Le reti neurali ricorrenti permettono di usare uscite precedentemente stimate per determinare predizioni future.

Ad esempio, dato $m = 2$ e $d = 0$, allora:

$$\begin{aligned}
\hat{y}(t) &= f\left(y(t-1), y(t-2), u(t), u(t-1), u(t-2)\right) \\
\hat{y}(t+1) &= f\left(\hat{y}(t), y(t-1), u(t+1), u(t), u(t-1)\right) \\
&\vdots \\
\hat{y}(t+k) &= f\left(\hat{y}(t+k-1), \hat{y}(t+k-2), u(t+k), u(t+k-1), u(t+k-2)\right)
\end{aligned} \tag{2.18}$$

In questo modo, una rete neurale ricorrente è in grado di compiere stime a lungo termine seguendo un approccio ricorsivo (Figura 2.4).

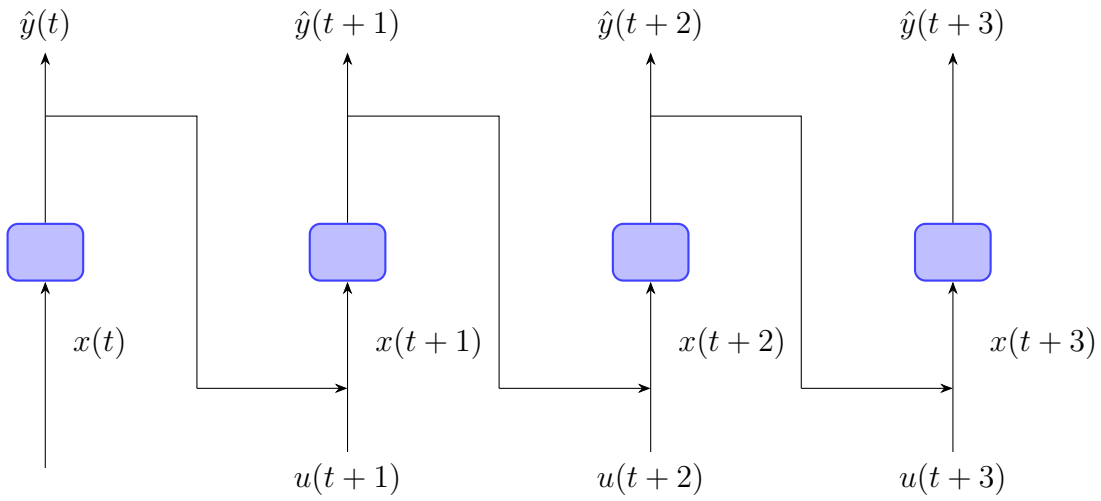


Figura 2.4: Predizione a lungo termine con rete neurale ricorrente.

2.5 Regressione nell'identificazione di sistemi

La regressione è uno dei problemi più comuni nell'identificazione di sistemi dinamici. L'obiettivo principale di un algoritmo di regressione è quello di imparare a predire il corretto valore $y \in \mathcal{Y}$ corrispondente ad un certo ingresso $x \in \mathcal{X}$. In contrapposizione al problema della classificazione, lo spazio delle uscite è continuo, ovvero $\mathcal{Y} = \mathbb{R}^k$. Per raggiungere questo scopo, si utilizza un insieme di coppie ingresso-uscita $(x(t), y(t)) \sim p(x(t), y(t))$ per $t = 0, 1, ...N$ che costituiscono l'insieme dei dati di allenamento. La risoluzione del problema consiste solitamente nel tracciare la curva che descrive al meglio la loro relazione.

Uno dei principali campi di applicazione degli algoritmi di regressione è la visione artificiale. In questo contesto, lo spazio degli ingressi $\mathcal{X}$ corrisponde allo spazio delle immagini, mentre quello delle uscite $\mathcal{Y}$ dipende dal tipo di problema specifico. Alcuni esempi includono:

- **Riconoscimento di oggetti:** individuare la posizione di un oggetto all'interno di un'immagine, solitamente disegnando un rettangolo di contorno $\mathcal{Y} \equiv \mathbb{R}^4$.
- **Stima dell'età:** riconoscere l'età di una persona da una foto del viso, $\mathcal{Y} \equiv \mathbb{R}_+$.
- **Tracciamento visivo di un oggetto in movimento:** seguire la posizione di un oggetto attraverso una sequenza di immagini nel tempo.

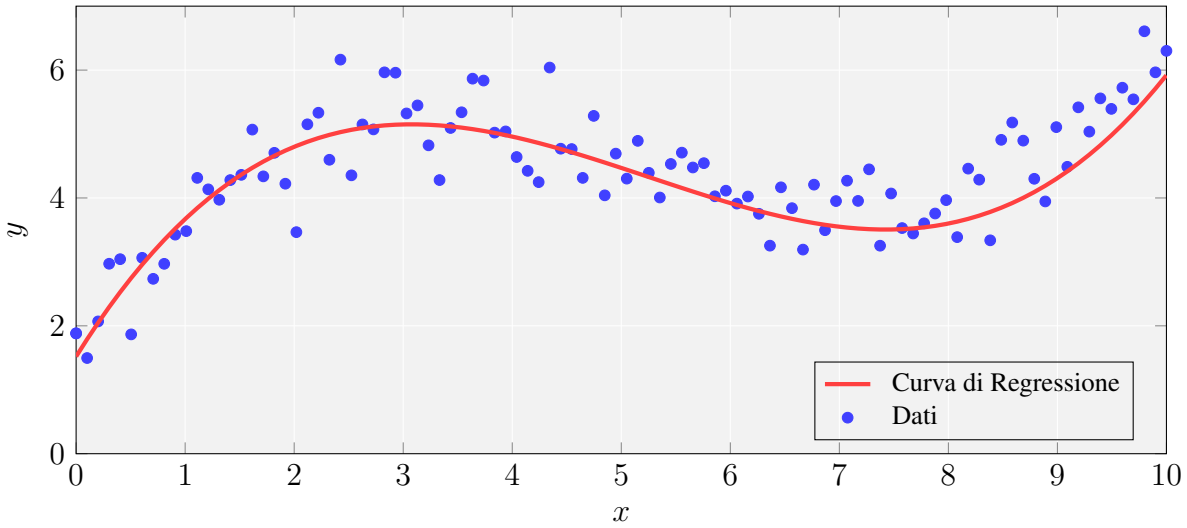


Figura 2.5: Regressione polinomiale. Il grafico rappresenta la regressione polinomiale di terzo grado ai dati, con $\mathcal{X} \equiv \mathbb{R}$ ed $\mathcal{Y} \equiv \mathbb{R}$. La funzione parametrizzata utilizzata è $f_\theta(x) = \theta_1 x^3 + \theta_2 x^2 + \theta_3 x + \theta_4$. I dati sono costituiti da 100 punti generati da una funzione vera, alla quale è stato aggiunto rumore gaussiano indipendente con deviazione standard $\sigma = 0.5$. La curva rossa rappresenta la funzione di regressione polinomiale ottenuta tramite la stima ai minimi quadrati, con i parametri stimati $\hat{\theta} = [0.04, -0.63, 2.74, 1.52]^T$, che differiscono leggermente dai valori veri $\theta^* = [0.04, -0.649, 3, 1]^T$.

Nella situazione più semplice, si assume che i dati siano generati da una funzione parametrizzata e perturbati da un rumore gaussiano:

$$y(t) = f_\theta(x(t)) + e(t), \qquad e(t) \sim \mathcal{N}(0, \sigma^2) \tag{2.19}$$

Per risolvere il valore dei parametri θ si definisce la funzione verosimiglianza, nota anche come *likelihood*:

$$\mathrm{L}(\theta|y) = p(y|\theta) \tag{2.20}$$

dove $p(y|\theta)$ indica la densità condizionata di y dato θ, ovvero è la probabilità di osservare l'uscita una volta fissati i parametri. Per facilitare i calcoli, si utilizza spesso una funzione denominata *negative log likelihood*:

$$\ell(\theta|y) = -\log \mathrm{L}(\theta|y) \tag{2.21}$$

Per calcolare una stima dei parametri si vuole massimizzare la verosimiglianza dei dati rispetto a θ:

$$\hat{\theta}_{ML} = \arg\max_{\theta} \mathrm{L}(\theta|y) \iff \hat{\theta}_{ML} = \arg\min_{\theta} \ell(\theta|y) \tag{2.22}$$

Se la funzione parametrizzata f_θ è lineare rispetto ai parametri e considerando che $e(t)$ è gaussiano, allora, in base al teorema di Gauss-Markov, la stima a massima verosimiglianza coincide con l'ottimizzazione ai minimi quadrati:

$$\hat{\theta} = \arg\max_{\theta} \sum_{t=1}^{N} \|y(t) - f_\theta(x(t))\|^2 \tag{2.23}$$

2.5.1 Regressione attraverso reti neurali

In scenari più complessi in cui f_θ non è nota, la relazione tra ingressi e uscite è completamente sconosciuta. In questi casi, è necessario adottare un approccio di modellazione a scatola nera, per il quale le reti neurali sono comunemente utilizzate. Grazie alla loro flessibilità e capacità di modellare relazioni non lineari, sono in grado di risolvere efficacemente il problema di regressione.

Regressione diretta

La strategia più convenzionale è quella di allenare la rete a predire direttamente y dato x. Questo approccio, noto come *regressione diretta*, prevede la stima dei parametri della rete neurale minimizzando $\ell(\theta|y)$. La rete neurale è vista come una funzione da ingresso a uscita $f_\theta : \mathcal{X} \to \mathcal{Y}$, quindi la stima si ottiene da:

$$\hat{y}(t) = f_\theta(x(t)) \tag{2.24}$$

Questo metodo equivale a trovare la distribuzione di probabilità dipendente dai parametri che lega le uscite agli ingressi $p(y(t)|x(t);\theta)$. Tuttavia, questo approccio limita la flessibilità della rete neurale, poiché assumendo che l'errore sia gaussiano si assume anche che la soluzione lo sia. In altre parole, si ipotizza che $p(y(t)|x(t);\theta) \sim \mathcal{N}\left(f_\theta(x(t)), \sigma^2\right)$, perciò la rete neurale è allenata unicamente per stimare la media di una distribuzione gaussiana.

Regressione probabilistica

Approfondendo la visione probabilistica del problema, un approccio alternativo è quello della *regressione probabilistica*. Le reti neurali, in questo caso, sono sfruttate per determinare i parametri di una distribuzione di probabilità specifica per $p(y(t)|x(t);\theta)$.

Ad esempio, una distribuzione gaussiana a una dimensione ha due parametri: media e varianza. La rete neurale, applicando la stima a massima verosimiglianza sui propri parametri θ, determinerà i parametri della distribuzione in questo modo: $f_\theta = [\mu_\theta(x), \log \sigma^2_\theta(x)]^T \in \mathbb{R}^2$.

Al momento di testare la rete, quindi di valutare l'uscita stimata corrispondente al valore $x(t)$, viene considerato il valore medio della distribuzione stimata per quell'ingresso:

$$\hat{y}(t) = \mathbb{E}\big[\mathcal{N}\left(\mu_\theta(x(t)), \sigma^2_\theta(x(t))\right)\big] = \mu_\theta(x(t)) \tag{2.25}$$

Lo scopo di questo secondo metodo non è unicamente quello di ottenere predizioni accurate, ma soprattutto di determinare una stima dell'incertezza della predizione. Tuttavia anche in questo caso, la rete neurale è allenata su un unica distribuzione che viene scelta all'inizio dell'allenamento.

Regressione basata sulla confidenza

Un'altra categoria è quella denominata *regressione basata sulla confidenza*. L'idea principale è quella di predire una quantità scalare $f_\theta(x, y)$ che dipende sia dell'ingresso x che dall'uscita y. Questo valore rispecchia la confidenza con cui la rete neurale associa y ad x.

Infine, la stima si ottiene calcolando per quale valore di y la verosimiglianza è massima:

$$\hat{y}(t) = \arg\max_y f_\theta(x(t), y) \tag{2.26}$$

2.5.2 Modelli basati sull'energia

Per sfruttare appieno la flessibilità che caratterizza le reti neurali, è fondamentale esprimere distribuzioni di probabilità generiche attraverso queste reti. Un metodo che si distingue da quelli sopra citati per questo aspetto è quello dei *modelli basati sull'energia*.

L'obiettivo è sempre quello di determinare il modello $p(y(t)|x(t);\theta)$, dove θ è ottenuto minimizzando la *negative log likelihood*. Tuttavia, la funzione della rete neurale non è limitata a trovare i parametri di una specifica distribuzione di probabilità. Infatti, questa distribuzione non viene definita a priori come nei precedenti casi, dove era ristretta alla sola famiglia delle gaussiane. Al contrario, la rete neurale possiede gli strumenti per modellare una propria distribuzione di probabilità.

Analogamente alla regressione basata sulla confidenza, anche in questo caso la rete neurale predice uno scalare in funzione delle coppie ingresso-uscita (x, y). Pertanto, la rete neurale è una funzione del tipo: $f_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ parametrizzata da $\theta \in \mathbb{R}^P$.

Il modello risultante è definito da:

$$p(y|x;\theta) = \frac{\mathrm{e}^{f_\theta(x,y)}}{Z(x,\theta)}, \qquad Z(x,\theta) = \int \mathrm{e}^{f_\theta(x,\tilde{y})}\, d\tilde{y} \tag{2.27}$$

dove la funzione $Z(x,\theta)$ è nota come funzione di normalizzazione dipendente dall'ingresso. Essa garantisce che la distribuzione di probabilità sia corretta, rispettando le seguenti proprietà:

- **Validità:** $\forall y \in \mathcal{Y} \quad p(y|x;\theta) \in [0,1]$
- **Normalizzazione:** $\int p(y|x;\theta)\, dy = 1$

Processo di addestramento

Per allenare il modello (2.27) si minimizza la *negative log likelihood*, che è espressa come:

$$\begin{aligned} \ell(\theta|y) &= \sum_{t=1}^{N} -\log p(y(t)|x(t);\theta) \\ &= \sum_{t=1}^{N} \log\left(\int \mathrm{e}^{f_\theta(x(t),y)}\, dy\right) - f_\theta(x(t),y(t)) \end{aligned} \tag{2.28}$$

Questo processo di addestramento richiede la valutazione dell'integrale $Z(x(t),\theta)$, che generalmente risulta essere intrattabile. Per questo motivo, si ricorre ad una approssimazione dell'integrale tramite il metodo di Monte Carlo.

Per applicarlo, è necessario definire una distribuzione di probabilità $q(y|y(t))$. Dato l'integrale $Z(x(t),\theta)$, si moltiplica e divide per la densità di probabilità $q(y|y(t))$.

$$\begin{aligned} \int \mathrm{e}^{f_\theta(x(t),y)}\, dy &= \int \frac{\mathrm{e}^{f_\theta(x(t),y)}}{q(y|y(t))}\, q(y|y(t))\, dy \\ &= \int h(y)\, q(y|y(t))\, dy = \mathbb{E}[h(y)] \end{aligned} \tag{2.29}$$

Riscrivendo l'integrale con $h(y)$, si nota come esso sia equivalente all'aspettazione della funzione appena introdotta.

Per la Legge dei Grandi Numeri, si ottiene una stima dell'integrale da:

$$\mathbb{E}[h(y)] \approx \frac{1}{M} \sum_{m=1}^{M} h(y^{(m)}) \tag{2.30}$$

dove i valori $\{y^{(m)}\}_{m=1}^{M}$ sono campionati dalla distribuzione di probabilità $q(y|y(t))$. In conclusione, l'approssimazione dell'integrale è:

$$Z(x(t),\theta) \approx \frac{1}{M}\sum_{m=1}^{M}\frac{\mathrm{e}^{f_\theta(x(t),y^{(m)})}}{q(y^{(m)}|y(t))} \tag{2.31}$$

Nella pratica, per allenare la rete neurale, si divide l'insieme dei dati in sottogruppi più piccoli in modo da incrementare l'efficienza computazionale e la stabilità dei risultati dell'addestramento. Per ognuno di questi gruppi si definisce una approssimazione di (2.28) da minimizzare:

$$J(\theta) = \frac{1}{n}\sum_{t=1}^{n}\log\left(\frac{1}{M}\sum_{m=1}^{M}\frac{\mathrm{e}^{f_\theta(x(t),y^{(m)})}}{q(y^{(m)}|y(t))}\right) - f_\theta(x(t),y(t)) \tag{2.32}$$

Minimizzando $J(\theta)$, si incoraggia la rete neurale a produrre un uscita elevata $f_\theta(x(t),y(t))$ corrispondente al dato osservato $y(t)$, attenuando allo stesso tempo le uscite $f_\theta(x(t),y)$ corrispondenti ad ogni altro y.

L'unica scelta di design nel processo di allenamento di questo modello è la scelta di $q(y|y(t))$. Tuttavia, siccome in (2.32) $J(\theta)$ si adatta automaticamente a $q(y|y(t))$, questa scelta influisce solamente sul numero e distribuzione dei campioni $y^{(m)}$.

In [129], si afferma che si ottengono sperimentalmente ottimi risultati utilizzando una media di gaussiane indipendenti, tutte centrate nel valore osservato $y(t)$.

$$q(y|y(t)) = \frac{1}{L}\sum_{i=1}^{L}\mathcal{N}\left(y(t),\sigma_i^2 I\right) \tag{2.33}$$

Predizione

Per testare il modello, dato l'ingresso $x(t)$, si calcola l'approssimazione di $Z(x(t),\theta)$ mentre la rete neurale determina $f_\theta(x(t),y)$ per ogni $y \in \mathcal{Y}$. In questo modo si ottiene la distribuzione di probabilità delle uscite condizionata dall'ingresso $p(y|x(t);\theta)$, da cui si possono stimare la media e la varianza del valore $\hat{y}(t)$.

Se si è solamente interessati alla stima dell'uscita, si considera il valore più probabile.

$$\hat{y}(t) = \arg\max_y p(y|x(t);\theta) = \arg\max_y f_\theta(x(t),y) \tag{2.34}$$

In questo caso, non è necessaria la stima di $Z(x(t),\theta)$.

Per trovare il massimo della funzione $f_\theta(x(t),y)$ si applica un algoritmo di salita del gradiente $\nabla_y f_\theta(x(t),y)$.

Partendo da un insieme di inizializzazione randomica $\{\hat{y}_k\}_{k=1}^{K}$, all'iterazione n si ha:

$$\hat{y}^{(n+1)} \leftarrow \hat{y}^{(n)} + \lambda \nabla_y f_\theta(x(t), \hat{y}^{(n)}) \tag{2.35}$$

Ad ogni iterazione dell'algoritmo, la stima corrente viene aggiornata seguendo la crescita del gradiente della funzione secondo il passo di apprendimento λ. L'algoritmo termina se:

- $\nabla_y f_\theta(x(t), \hat{y}^{(n)}) \leq \varepsilon$ quindi si raggiunge un massimo locale;
- oppure dopo un numero finito T di passi.

Poiché l'algoritmo potrebbe restituire la posizione di un massimo locale e per la casualità di $\hat{y}^{(0)}$, solitamente si applica lo stesso procedimento in parallelo su K inizializzazioni differenti. Al termine di ogni algoritmo, la stima finale sarà quella che massimizza $f_\theta(x(t), \hat{y})$.

Capitolo 3

Studi su modelli con molti parametri

Nel campo dell'identificazione di sistemi, la scelta iniziale della famiglia di modelli che potrebbero approssimare il fenomeno fisico è la più significativa di tutto il processo. Optare per un modello troppo "semplice" può portare a una descrizione inadeguata dei dati raccolti. Tuttavia, un'eccessiva complessità del modello può risultare altrettanto problematica, poiché può compromettere la sua capacità di generalizzare correttamente al di fuori dei dati di addestramento.

La complessità di un modello si riferisce alla misura della difficoltà o della sofisticazione del modello stesso, sia in termini di struttura che di funzionamento. Solitamente, il numero di parametri è utilizzato come indicatore della complessità. Ad esempio, una rete neurale con diversi neuroni, e quindi con molteplici pesi sinaptici, è in grado di riprodurre una grande varietà di sistemi.

Nella teoria classica, si tende a non complicare eccessivamente il modello, seguendo i principi di Occam, della parsimonia e della falsificabilità di Popper. Tuttavia, le recenti scoperte riguardanti le reti neurali di grandi dimensioni, dove il numero di parametri può superare persino il numero di dati disponibili per l'allenamento, hanno alterato questa prospettiva. Queste reti neurali, sebbene in grado di descrivere perfettamente i dati di addestramento, riescono comunque a ottenere buone prestazioni sui dati di validazione, dimostrando una capacità di generalizzazione sorprendente nonostante la loro complessità.

3.1 Teoria classica sulla selezione dell'ordine di un modello

Se si considera il numero di parametri, denotato con $m = \dim(\theta)$, come indicatore della complessità, si può selezionare l'ordine ottimale del modello testandolo progressivamente con valori crescenti di m. Per farlo, è necessario disporre di due insiemi di dati indipendenti dello stesso fenomeno: uno per l'allenamento Z_e e uno per la validazione Z_v, rispetto ai quali calcolare

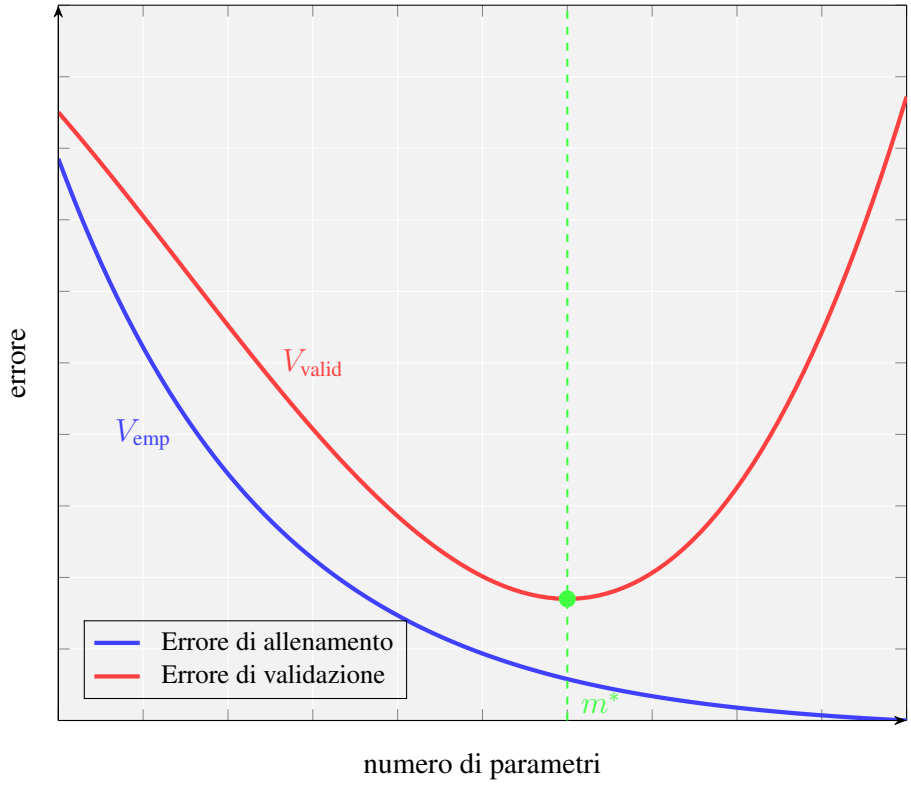


Figura 3.1: Andamento tipico degli errori al variare dell'ordine del modello.

i corrispondenti errori V_{emp} (1.3) e V_{valid} (1.4). Secondo la teoria classica, l'andamento tipico dell'errore di validazione segue una curva a "U", con una condizione ottimale in $m = m^*$ da trovare (Figura 3.1).

Fase di discesa dell'errore $(m \leq m^*)$

All'inizio, quando il modello ha pochi parametri, è troppo semplice per catturare le relazioni tra ingressi e uscite. In questa fase V_{emp} e V_{valid} sono entrambi elevati, di conseguenza il modello è sotto-adattato (*underfitting*) e caratterizzato da un grande *bias*. Con quest'ultimo termine si intende l'incapacità del modello di catturare la vera relazione nei dati, spesso perché dotato di una struttura troppo semplice.

Aumentare la complessità permette al modello di essere più flessibile e quindi di rappresentare più accuratamente i dati di allenamento. Per questo motivo, V_{emp} è monotonamente non-crescente in funzione di m.

Raggiungendo $m = m^*$, si arriva a una condizione tale in cui V_{valid} è al suo minimo, mentre V_{emp} è sufficientemente piccolo da conferire al modello un *bias* ridotto.

Fase di crescita dell'errore $(m > m^*)$

Superata la soglia ottimale, il modello continuerà a rappresentare sempre più precisamente i dati di allenamento, fino a raggiungere una condizione di interpolazione perfetta $V_{\text{emp}} = 0$, tipicamente ottenuta quando il numero di parametri è uguale al numero di dati a disposizione $m = N = \dim Z_e$.

Tuttavia, seppure l'errore sui dati di allenamento sia prossimo a zero, V_{valid} è solitamente molto grande. Questo si verifica perché il modello è diventato troppo complesso, iniziando a modellare anche il rumore nei dati. Questa condizione è nota come *overfitting*: il modello si adatta perfettamente ai dati di allenamento, ma generalizza male sui dati nuovi. In questa situazione, il modello acquisisce una varianza elevata, che indica la sua sensibilità alle piccole variazioni nei dati di allenamento causate dal rumore.

La scelta ottimale sull'ordine del modello è ottenuta attraverso un compromesso tra *bias* e varianza, che è differente per ogni problema. Nella teoria classica, questa scelta è guidata dai seguenti tre principi fondamentali.

1. **Rasoio di Occam:** questa regola, attribuita a Guglielmo da Occam, afferma *"Entia non sunt multiplicanda praeter necessitatem"* ovvero "Non bisogna moltiplicare gli elementi più del necessario". In altre parole, non c'è motivo alcuno per complicare ciò che è semplice, perciò è meglio eliminare parti del modello superflue.

2. **Principio di parsimonia:** afferma che, se si dispone di due spiegazioni concorrenti per un fenomeno, quella che fa meno assunzioni è generalmente preferibile.

3. **Tesi di Popper:** nota anche come principio di falsificabilità, afferma che un modello scientifico deve essere formulato in modo tale che possa essere sottoposto a test empirici che possano dimostrarne la falsità. Inoltre, il modello migliore è il più semplice tra tutti quelli non falsificati.

3.1.1 Studio asintotico dell'errore di generalizzazione

Le considerazioni finora discusse forniscono una guida generale per la selezione dell'ordine ottimale di un modello. Tuttavia, per rafforzare queste intuizioni, è utile approfondire ulteriormente l'analisi attraverso strumenti matematici.

In particolare, lo studio dell'errore di generalizzazione fornisce una visione quantitativa di come il numero di parametri m influisce sulla capacità del modello di adattarsi correttamente ai dati. Lo studio matematico riportato nella sezione 16.4 di [6] offre un quadro asintotico che consente di esaminare l'effetto del numero di parametri sulla generalizzazione del modello.

Si consideri il caso in cui il numero di dati a disposizione sia molto più grande rispetto al numero di parametri ($m \ll N$), tendente all'infinito per alcune semplificazioni $N \to \infty$. Supponendo inoltre che i dati riguardanti il fenomeno siano disturbati da un rumore a media nulla e varianza σ^2, allora la stima $\hat{\theta}_N$ è una variabile aleatoria con le seguenti media e varianza:

$$\mathbb{E}\Big[\hat{\theta}_N\Big] \approx \theta^* \qquad\qquad \mathrm{Var}\Big[\hat{\theta}_N\Big] = P_\theta = \mathbb{E}\Big[(\hat{\theta}_N - \theta^*)(\hat{\theta}_N - \theta^*)^T\Big] \tag{3.1}$$

dove si assume che il modello contenga la vera struttura rappresentata da θ^*.

Sia l'errore atteso rispetto ai dati sul modello definito in questo modo:

$$\bar{V}(\theta) = \lim_{N\to\infty} \mathbb{E}[V_N(\theta, Z_e)] \tag{3.2}$$

il cui valore minimo è $\bar{V}(\theta^*)$ ed è uguale alla varianza del rumore σ^2.

Inoltre, la matrice di covarianza della stima $\hat{\theta}_N$ è tipicamente approssimata da:

$$P_\theta \sim 2\sigma^2 \frac{\big[\nabla^2_\theta \bar{V}(\theta^*)\big]^{-1}}{N} \tag{3.3}$$

A questo punto si vuole valutare la capacità del modello di generalizzare correttamente. Per questo scopo si considera la media di (3.2) rispetto ai parametri stimati, che è equivalente al valore atteso dell'errore di validazione $\mathbb{E}\Big[\bar{V}(\hat{\theta}_N)\Big] = \mathbb{E}[V_{\text{valid}}]$.

Per calcolare questo valore, innanzitutto si valuta l'espansione di Taylor al secondo ordine di $\bar{V}(\hat{\theta}_N)$.

$$\bar{V}(\hat{\theta}_N) \approx \bar{V}(\theta^*) + \nabla_\theta \bar{V}(\theta^*)^T\,(\hat{\theta}_N - \theta^*) + \frac{1}{2}(\hat{\theta}_N - \theta^*)^T\,\nabla^2_\theta \bar{V}(\theta^*)\,(\hat{\theta}_N - \theta^*) \tag{3.4}$$

Siccome θ^* è punto di minimo di $\bar{V}(\theta)$, allora il suo gradiente in quel punto è nullo, quindi $\nabla_\theta \bar{V}(\theta^*) = 0$. Applicando il valore atteso al membro di secondo ordine, sfruttando le equazioni (3.1, 3.3), si ottiene:

$$\begin{aligned}
\mathbb{E}\Big[(\hat{\theta}_N - \theta^*)^T\,\nabla^2_\theta \bar{V}(\theta^*)\,(\hat{\theta}_N - \theta^*)\Big] &= \mathrm{Tr}\Big[\nabla^2_\theta \bar{V}(\theta^*)\,\mathbb{E}\Big[(\hat{\theta}_N - \theta^*)(\hat{\theta}_N - \theta^*)^T\Big]\Big] \\
&= \mathrm{Tr}\big[\nabla^2_\theta \bar{V}(\theta^*)\,P_\theta\big] = \mathrm{Tr}\left[\frac{2\sigma^2}{N} I\right] \\
&= \frac{2\sigma^2}{N} m
\end{aligned} \tag{3.5}$$

Usando questo sviluppo si ha che:

$$\mathbb{E}[V_{\text{valid}}] \approx \mathbb{E}\big[\bar{V}(\theta^*)\big] + \frac{1}{2}\mathbb{E}\Big[(\hat{\theta}_N - \theta^*)^T\,\nabla^2_\theta \bar{V}(\theta^*)\,(\hat{\theta}_N - \theta^*)\Big] = \sigma^2\left(1 + \frac{m}{N}\right) \tag{3.6}$$

In questo modo, si è legato l'errore di validazione al rapporto m/N, mostrando come un numero elevato di parametri peggiori in media la capacità di generalizzazione del modello. Questo studio asintotico fornisce quindi una giustificazione matematica ai principi cardine tradizionalmente seguiti nella selezione dell'ordine di un modello.

Tuttavia, è importante notare che questa analisi si basa su assunzioni restrittive sul numero di parametri e di dati, come $N \to \infty$ e $m \ll N$. Per grandi parametrizzazioni, dove $m > N$, questo studio perde di validità. Nella sezione successiva, verrà affrontato proprio il comportamento del modello in questi casi particolari, che sono stati a lungo trascurati nella teoria classica e che porteranno allo studio del fenomeno della doppia discesa.

3.2 Fenomeno della doppia discesa

In tempi più recenti, l'uso delle reti neurali profonde, caratterizzate da numerosi strati di neuroni, e di altri modelli non lineari complessi è diventato sempre più diffuso. Queste strutture sono spesso caratterizzate da un numero di parametri talmente elevato da essere comparabile con la quantità di dati utilizzata per l'allenamento.

Nonostante la loro elevata complessità e la capacità di interpolare quasi perfettamente i dati di addestramento, questi predittori riescono comunque a generalizzare in modo efficace su nuovi dati. Tale risultato è in contrasto con le regole tradizionali, secondo cui una capacità così elevata di adattamento ai dati di addestramento avrebbe dovuto portare a problemi di *overfitting*.

Tuttavia, nell'applicazione di questi modelli si è osservato che le prestazioni continuano a migliorare anche oltre la soglia di interpolazione. Questo fenomeno ha portato gli autori di [1] a descrivere l'andamento dell'errore di generalizzazione di questi modelli con il termine "curve a doppia discesa".

3.2.1 Seconda discesa nella curva di errore

La rinnovata consapevolezza riguardante la relazione tra complessità e capacità di generalizzazione non rimpiazza, ma integra le conoscenze classiche. Ciò che si osserva, infatti, nella regione in cui $m < N$, è il comportamento classicamente assunto che vede l'errore di generalizzazione seguire un andamento ad "U", come mostrato in Figura 3.2.

Quando si giunge alla soglia di interpolazione ($m = N$), il modello solitamente ha un picco nell'errore di generalizzazione. Tuttavia, aumentando l'ordine del modello, l'errore comincia a scendere di nuovo, migliorando le prestazioni. In molti casi, quando $m \gg N$, la qualità del modello risulta superiore rispetto alla condizione tradizionale $m = m^*$.

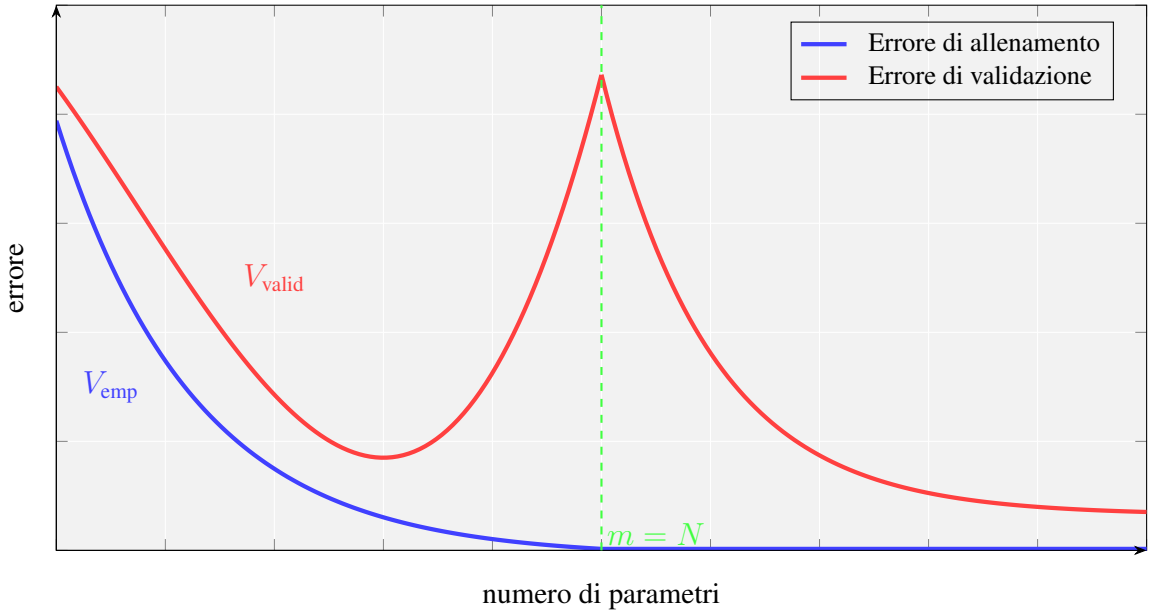


Figura 3.2: Andamento degli errori di allenamento e validazione al variare del numero di parametri, che illustra il fenomeno della doppia discesa. La linea verticale verde rappresenta la soglia di interpolazione, che separa la regione sottoparametrizzata del regime classico da quella sovraparametrizzata del regime di interpolazione.

Questo fenomeno, siccome mostra un errore inferiore dove il modello è più complesso, sembra contraddire i principi di parsimonia e semplicità. In realtà, questa teoria innovativa non è totalmente in contrasto con quella classica, che predilige un modello semplice a uno troppo complicato. Infatti, la definizione di complessità di un modello può variare. In questo capitolo, si è sempre assunto il numero di parametri m come indicatore di complessità, ma ciò potrebbe non valere per modelli con un numero molto elevato di parametri.

Per questo scopo, è importante notare che, in regime di interpolazione, esistono molteplici soluzioni per i parametri $\hat{\theta}_N$ che minimizzano l'errore (1.1). Un metodo comune è quello di selezionare la soluzione a norma minima tra tutte quelle possibili.

$$\hat{\theta} = \arg\min_{\theta} \|\theta\|_2 \tag{3.7}$$

Alla soglia di interpolazione, si ottiene una soluzione unica con una norma molto grande, tuttavia, aumentando m, si amplia lo spazio delle soluzioni disponibili, rendendo possibile individuarne una con norma più piccola.

La norma dei parametri è strettamente legata alla *smoothness* della curva di predizione del modello. Scegliere una soluzione con norma bassa porta a modelli dotati di curve meno ondulate e più regolari, che possono essere considerate più semplici. Pertanto, questo approccio può essere visto come una forma di applicazione del rasoio di Occam, dove, tra un insieme di

possibili parametrizzazioni, si preferisce una funzione più regolare e quindi più semplice.

In conclusione, aumentando il numero di parametri del modello oltre la soglia $m = N$, si rende in realtà il modello più semplice. Perciò, queste scoperte innovative riguardanti la curva di errore non si scontrano con i principi di parsimonia classicamente adottati, ma piuttosto fanno luce sull'arbitrarietà della definizione di complessità di un modello.

3.2.2 Modello asintotico della doppia discesa

Come per la teoria classica, anche queste nuove scoperte relative alla "doppia discesa" sono supportate da studi matematici asintotici. Precedentemente, il numero di dati tendeva all'infinito mentre il numero di parametri variava all'interno di un intervallo finito. Tuttavia, per analizzare il fenomeno della doppia discesa, è necessario considerare scenari in cui sia il numero di parametri che il numero di dati tendono simultaneamente all'infinito, mantenendo costante il loro rapporto $\frac{m}{N}$ che converge a un valore finito γ.

Si consideri il semplice problema di regressione lineare, in cui i dati di allenamento e di validazione vengono generati linearmente con l'aggiunta di rumore:

$$y_t = x_t^T \theta^* + \varepsilon_t \qquad\qquad (x_t, \varepsilon_t) \sim P_x \times P_\varepsilon \tag{3.8}$$

dove gli ingressi x_t e il rumore ε_t sono indipendenti e identicamente distribuiti al variare di $t = 1, 2, \ldots, N$. Si suppone che P_x sia una distribuzione su $\mathbb{R}^m$ tale che $\mathbb{E}[x_t] = 0$, $\mathrm{Var}[x_t] = \Sigma$, e che P_ε sia una distribuzione su $\mathbb{R}$ con $\mathbb{E}[\varepsilon_t] = 0$, $\mathrm{Var}[\varepsilon_t] = \sigma^2$. La norma del vettore di parametri di generazione è indicata con $r^2 = \|\theta^*\|^2$.

Inoltre, sia X la matrice di dimensioni $N \times m$, in cui l'i-esima riga è data dal vettore x_i^T. Si assume che i parametri vengano stimati attraverso lo stimatore ai minimi quadrati $\hat{\theta} = (X^T X)^+ X^T y$, dove $(X^T X)^+$ è la matrice pseudo inversa di Moore-Penrose di $X^T X$. Questo stimatore ha la proprietà di restituire la soluzione a norma minima in regime di interpolazione.

Dato un punto dell'insieme di test $x_0 \sim P_x$, il rischio di previsione esterno all'apprendimento si definisce come:

$$R_X(\hat{\theta}, \theta^*) = \mathbb{E}\Big[(x_0^T \hat{\theta} - x_0^T \theta^*)^2 \mid X\Big] = \mathbb{E}\left[\|\hat{\theta} - \theta^*\|_\Sigma^2 \mid X\right] \tag{3.9}$$

dove $\|v\|_\Sigma^2 = v^T \Sigma v$.

È importante notare che il rischio così definito è condizionato dai dati di allenamento X e

può essere scomposto in due componenti: una dovuta al *bias* e una alla varianza.

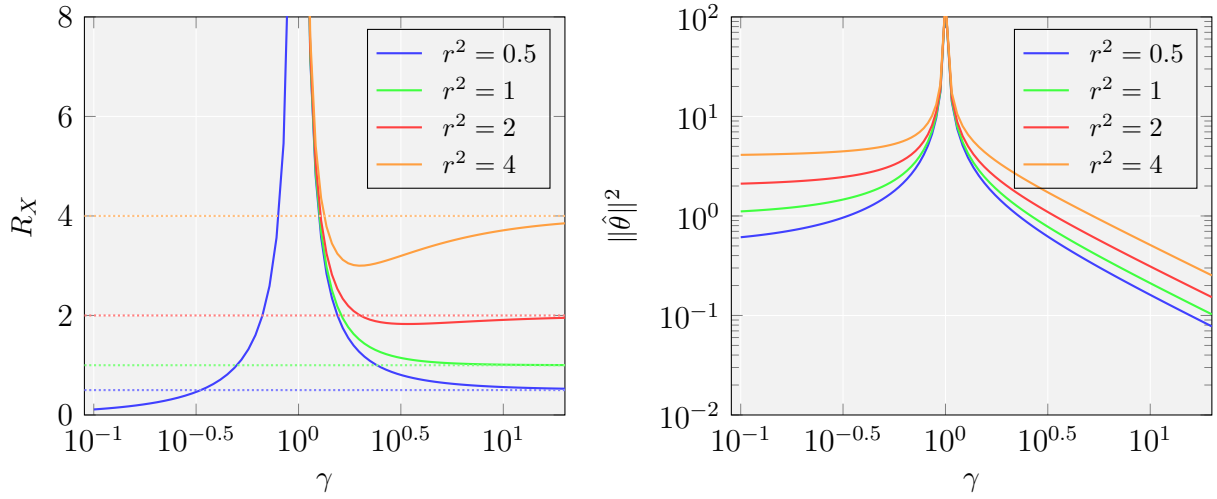
$$R_X(\hat{\theta}, \theta^*) = \underbrace{\left\| \mathbb{E}\left[\hat{\theta}|X\right] - \theta^* \right\|_\Sigma^2}_{B_X(\hat{\theta},\theta^*)} + \underbrace{\mathrm{Tr}\left[\mathrm{Var}\left[\hat{\theta}|X\right]\Sigma\right]}_{V_X(\hat{\theta},\theta^*)} \tag{3.10}$$

Dall'analisi asintotica sviluppata in [4], si conclude che, per $m, N \to \infty$ ed $\frac{N}{m} \to \gamma$, il rischio può essere espresso come segue:

$$R_X(\gamma) = \begin{cases} \sigma^2 \frac{\gamma}{1-\gamma} & \text{per } \gamma < 1 \\ r^2 \left(\frac{\gamma-1}{\gamma}\right) + \sigma^2 \frac{1}{\gamma-1} & \text{per } \gamma > 1 \end{cases} \tag{3.11}$$

Allo stesso modo, si ottiene asintoticamente la seguente espressione per il valore medio della norma al quadrato di $\hat{\theta}$.

$$\mathbb{E}\left[\|\hat{\theta}\|^2|X\right] = \begin{cases} r^2 + \sigma^2 \frac{\gamma}{1-\gamma} & \text{per } \gamma < 1 \\ r^2 \frac{1}{\gamma} + \sigma^2 \frac{1}{\gamma-1} & \text{per } \gamma > 1 \end{cases} \tag{3.12}$$



(a) Rischio di previsione

(b) Norma media

Figura 3.3: Andamento asintotico del rischio di previsione e della norma dei parametri stimati. Per il tracciamento delle curve si è assunto $\sigma^2 = 1$ e si è fatto variare la norma dei parametri "veri" $\|\hat{\theta}\|^2 = r^2$.

Quando $\gamma < 1$, il problema si trova nella regione sottoparametrizzata, mentre per $\gamma > 1$, si entra nella regione sovraparametrizzata. L'andamento del rischio, rappresentato nel grafico a sinistra di Figura 3.3, mostra chiaramente una seconda discesa nella regione di sovraparametrizzazione. Tuttavia, in questo modello teorico, nonostante si osservi il fenomeno della dop-

pia discesa, le prestazioni del modello nella regione sottoparametrizzata risultano comunque superiori rispetto a quelle nella regione sovraparametrizzata.

Inoltre, attraverso questo studio, è possibile analizzare anche l'andamento della norma media, a destra di Figura 3.3. Superata la soglia di $\gamma = 1$, nella regione sovraparametrizzata la norma dei parametri stimati diminuisce perché si sta utilizzando uno stimatore a norma minima. Quindi il modello, seppure con sempre più parametri, diventa più semplice e questo porta ad una discesa nel rischio di predizione.

3.3 Esperimento con modello NARX

In questa sezione verrà presentato un esempio pratico di problema in cui si manifesta il fenomeno della doppia discesa nella curva dell'errore. L'obiettivo è verificare sperimentalmente i risultati teorici ottenuti riguardo l'errore di predizione e la norma dei parametri, fornendo un riscontro concreto. L'esperimento è stato condotto utilizzando l'ambiente di sviluppo Matlab.

Il sistema dinamico non lineare da apprendere è descritto dalla seguente relazione ingresso-uscita:

$$\begin{aligned} y_t = &\left(0.8 - 0.5\mathrm{e}^{-y_{t-1}^2}\right) y_{t-1} - \left(0.3 + 0.9\mathrm{e}^{-y_{t-1}^2}\right) y_{t-2} \\ &+ u_{t-1} + 0.2u_{t-2} + 0.1u_{t-1}u_{t-2} + v_t \end{aligned} \tag{3.13}$$

dove $u_t \in \mathbb{R}$ è il segnale di ingresso, $y_t \in \mathbb{R}$ è il segnale di uscita e $v_t \sim \mathcal{N}(0, \sigma_v^2)$ rappresenta un rumore gaussiano con $\sigma_v = 0.1$ che perturba i dati.

3.3.1 Generazione dei dati

Per acquisire i dati su cui basare il modello, sono stati generati $N = 200$ campioni ingresso-uscita per l'allenamento e $N' = 100$ campioni per il test. Il segnale u_t è stato ottenuto applicando un filtro passa-basso con frequenza di taglio $\omega_c = 0.7$ a un segnale gaussiano bianco con varianza unitaria. Dopo aver generato anche il rumore v_t, l'uscita del sistema si ricava dall'espressione (3.13).

Il codice Matlab utilizzato per la generazione dei dati dell'esperimento è il seguente:

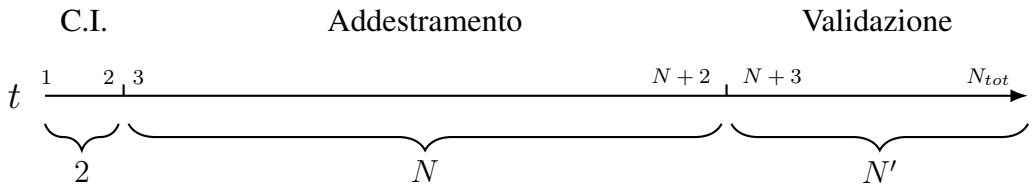
```
% generazione segnale di ingresso
u = randn(Ntot, 1);
[z, p, k] = ellip(8, 0.1, 60, cutoff_freq);     % filtro passa basso
sos = zp2sos(z, p, k);
u = filtfilt(sos, 1, u);

```

```
% generazione del rumore
v = randn(Ntot, 1) * sigma;

% generazione del segnale di uscita
y = zeros(Ntot, 1);     % y(1) e y(2) sono posti a 0
for t = 3:Ntot
    % Calcola y_t usando la formula data
    y(t) = (0.8 - 0.5*exp(-y(t-1)^2)) * y(t-1) ...
           - (0.3 + 0.9*exp(-y(t-1)^2)) * y(t-2) ...
           + u(t-1) + 0.2*u(t-2) ...
           + 0.1*u(t-1)*u(t-2) ...
           + v(t);
end
```

Per calcolare y_t sono necessari i valori precedenti y_{t-1}, y_{t-2}, quindi sono state poste le condizioni iniziali $y_1 = y_2 = 0$. Questi due campioni non saranno usati per l'allenamento, di conseguenza il numero totale di campioni da generare è $N_{TOT} = 2 + N + N'$. I dati per l'addestramento comprenderanno le coppie ingresso-uscita per $t \in [3, N + 2]$, mentre i dati per la validazione coprono l'intervallo di tempi $t \in [N + 3, N_{tot}]$.



3.3.2 Realizzazione e apprendimento del modello

Per identificare il sistema è stato utilizzato un modello NARX, caratterizzato dal seguente vettore di regressione:

$$x_t = [y_{t-1}, y_{t-2}, u_{t-1}, u_{t-2}] \tag{3.14}$$

In questo modo, il modello $f(x_t)$ dipenderà dagli ingressi e uscite passate, proprio come il sistema dinamico originale (3.13).

La predizione dell'uscita del sistema è stata effettuata utilizzando un modello lineare nei parametri, descritto dalla seguente equazione:

$$\hat{y}_t = f(x_t) = \sum_{i=1}^{m} \phi_i(x_t)\theta_i \tag{3.15}$$

dove le *features* $\phi_i(x_t)$ sono state selezionate in modo da approssimare il *kernel Hilbert space* e sono definite come segue:

$$\phi_i(x_t) = \sqrt{\frac{2}{m}} \cos(x_t^T w_i + b_i) \tag{3.16}$$

dove $w_i \in \mathbb{R}^4$ è un vettore di lunghezza pari a quella del vettore di regressione, in cui ogni elemento è indipendentemente generato secondo la distribuzione $\mathcal{N}(0, 2\gamma)$. Per questo esperimento, si considera $\gamma = 0.6$. Inoltre, il termine $b_i \in \mathbb{R}$ è campionato dalla distribuzione uniforme $\mathcal{U}[0, 2\pi)$.

La stima dei parametri $\hat{\theta}$ si ottiene minimizzando la funzione di errore 1.1, che può essere riscritta in forma matriciale in questo modo:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N} \|y - \Phi\theta\|^2 \tag{3.17}$$

dove $\Phi \in \mathbb{R}^{N \times m}$ è la matrice contenente sulla riga t e colonna i la *feature* $\phi_i(x_t)$, mentre $y \in \mathbb{R}^N$ è il vettore delle uscite. La soluzione analitica per θ è data da:

$$\hat{\theta} = \Phi^+ y \tag{3.18}$$

dove Φ^+ denota la matrice pseudo inversa di Moore-Penrose di Φ.

Il codice Matlab per eseguire la stima dei parametri è il seguente:

```
% intervallo di tempi per i dati dell'allenamento
intervallo_allenamento = 3:N+2;

% matrice di regressione
X = zeros(N, 4);
X(:, 1) = y(intervallo_allenamento-1);        % y(t-1)
X(:, 2) = y(intervallo_allenamento-2);        % y(t-2)
X(:, 3) = u(intervallo_allenamento-1);        % u(t-1)
X(:, 4) = u(intervallo_allenamento-2);        % u(t-2)

% generazione dei pesi
w = randn(4, m) * sqrt(2*gamma);
% generazione dei bias
b = ones(N, 1) * rand(1, m)*2*pi;
% matrice delle features
phi = sqrt(2/m) * cos(X*w + b);

% stima dei parametri
theta = pinv(phi)*y(intervallo_allenamento);
```

3.3.3 Interpretazione dei risultati

Al fine di visualizzare la curva a doppia discesa nella capacità di generalizzazione del modello, sono stati eseguiti diversi esperimenti variando il numero di parametri nell'intervallo $[N \cdot 10^{-1}, N \cdot 10^{2}]$, ripetendo ogni configurazione 20 volte.

Per ogni sperimentazione è stato valutato il *Mean Squared Error* come indicatore delle prestazioni. Ad esempio, per i dati di validazione esso si calcola in questo modo:

$$\text{MSE} = \frac{1}{N'} \sum_{i=1}^{N'} \|\hat{z}_i - z_i\|^2 \tag{3.19}$$

dove $z, \hat{z}$ sono rispettivamente le uscite del sistema e quelle stimate dal modello, relative all'intervallo temporale $[N+3, N_{tot}]$. La Figura 3.4 mette a confronto le due differenti uscite nella configurazione con più parametri, mostrando che l'MSE tra le due è alquanto piccolo.

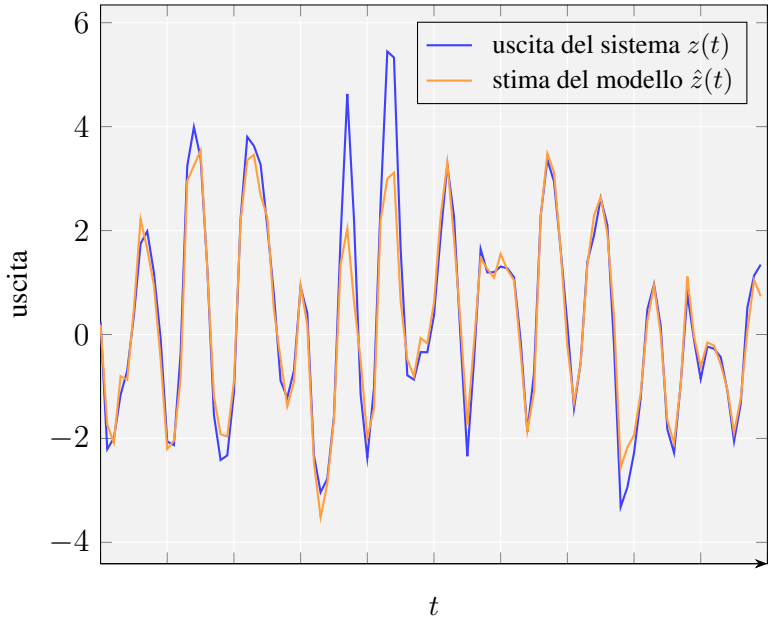
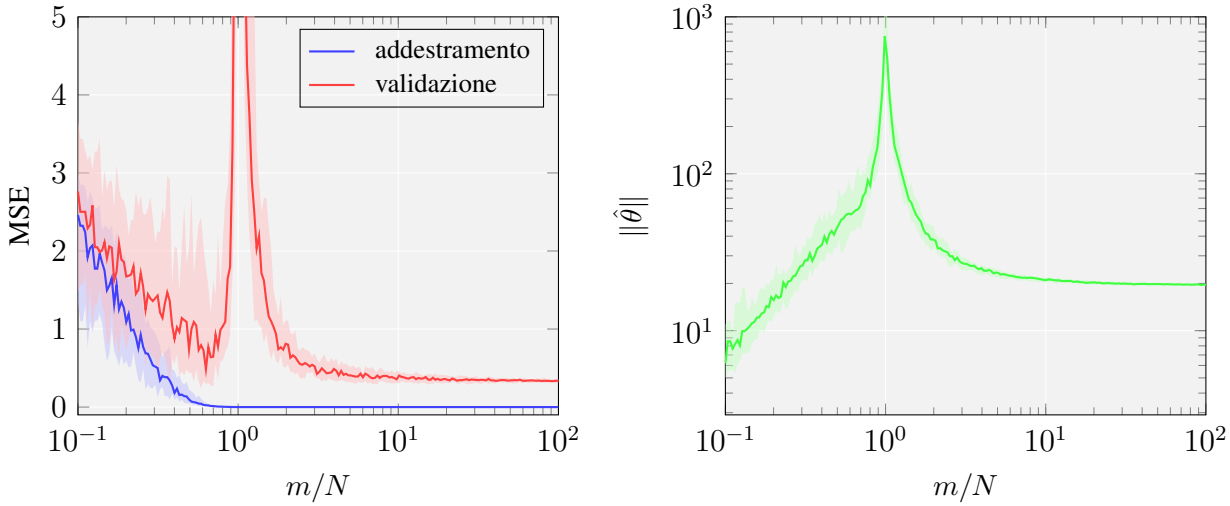


Figura 3.4: Uscita del sistema e predizione del modello a confronto. Le due uscite a confronto appartengono all'intervallo di tempi dedicati alla validazione. La predizione del modello disegnata in figura è ottenuta attraverso la parametrizzazione più vasta ($m/N = 10^2$). Poiché $\hat{z}$ è la stima su dati di ingresso nuovi rispetto a quelli dell'allenamento, è presente un errore tra le due curve di circa 0.3041.

In Figura 3.5 sono presentati i risultati degli esperimenti in funzione del rapporto m/N, ovvero il numero di parametri diviso per il numero di dati a disposizione per l'addestramento.

In Figura 3.5a è possibile notare come l'errore sui dati di allenamento è monotonamente decrescente, interpolando i dati in prossimità della soglia $m/N = 1$. Diverso è il comportamento dell'errore di validazione, che rappresenta la capacità del modello di generalizzare su dati non visti prima. Inizialmente, l'errore di validazione è caratterizzato da un andamento a "U" prima



(a) Errore sui dati di allenamento e testing

(b) Norma dei parametri stimati

Figura 3.5: Curva a doppia discesa nelle prestazioni del modello NARX. Per ogni differente proporzione m/N sono stati ripetuti 20 esperimenti generando diversamente la matrice Φ. In entrambi i grafici, la linea solida centrale rappresenta la mediana dei risultati, mentre la zona colorata più in chiaro delimita l'intervallo tra 10% percentile e 90%. Questa zona rappresenta dove risiedono l'80% dei risultati, quindi mostra loro variabilità rispetto alla mediana.

della soglia di interpolazione. Tuttavia, superata questa soglia, l'errore di validazione inizia a diminuire di nuovo. Questa seconda discesa nella curva dell'errore indica che, per $m/N > 1$, il modello riesce a migliorare la sua capacità di generalizzazione. Sorprendentemente, in questo caso le prestazioni del modello in regime di alta parametrizzazione risultano migliori rispetto a quelle nella regione classicamente considerata ottimale.

Un aspetto interessante emerso dai risultati riguarda la variabilità dell'errore di validazione, che si riduce significativamente all'aumentare del numero di parametri. Questo suggerisce che il modello tende a essere meno sensibile al rumore, offrendo prestazioni più stabili e affidabili, con una minore dispersione dei risultati attorno alla mediana.

In Figura 3.5b è rappresentato l'andamento della norma dei parametri stimati. All'aumentare del numero di parametri del modello, inizialmente la norma di $\hat{\theta}$ cresce rapidamente, per poi decrescere superata la soglia $m/N = 1$. Questo fenomeno può essere dovuto al metodo di risoluzione del problema ai minimi quadrati tramite la matrice pseudo inversa, che restituisce la stima $\hat{\theta}$ a norma minima. Perciò, potendo scegliere tra sempre più soluzioni per il problema, è possibile trovare una soluzione con norma più piccola.

In conclusione, i risultati dell'esperimento riproducono il fenomeno della doppia discesa, confermando che in un regime di sovra-parametrizzazione i modelli possono mostrare migliori capacità di generalizzazione rispetto al regime tradizionalmente considerato ottimale. Ciò combacia con quanto emerso dallo studio asintotico, i cui grafici (Figura 3.3) presentano un

comportamento molto simile a quello osservato nei risultati sperimentali. Questo suggerisce che, contrariamente a quanto si potrebbe pensare, dotare il modello di un numero di parametri molto maggiore rispetto ai dati utilizzati per stimarli può, in alcuni casi, portare a risultati migliori.

Capitolo 4

Conclusioni

Questa tesi ha approfondito alcuni degli sviluppi più innovativi nell'ambito dell'identificazione di sistemi, concentrandosi sull'applicazione delle reti neurali per l'apprendimento di sistemi dinamici complessi. Le reti neurali, grazie alla loro straordinaria flessibilità, si sono rivelate efficaci nel risolvere un'ampia varietà di problemi. Attraverso i modelli basati sull'energia, esse si sono rivelate utili anche nel problema di regressione, tipicamente affrontato quando si vuole apprendere un sistema dinamico dai dati.

Comunemente le reti neurali presentano un numero molto elevato di neuroni e quindi di parametri, tuttavia esse sono in grado di generalizzare efficacemente nonostante la loro complessità. Questo risultato ha portato allo studio del fenomeno delle curve a doppia discesa, che descrive come l'errore di generalizzazione può ridursi ulteriormente quando il numero di parametri supera quello dei dati di allenamento. È stato argomentato intuitivamente che tale comportamento in realtà non contraddice i principi classici di semplicità e parsimonia. Infatti, in regime di interpolazione, la soluzione per la stima dei parametri che minimizza l'errore di predizione non è unica, perciò tra le varie soluzioni è possibile sceglierne una a norma minore. Questo implica che la funzione di predizione del modello abbia un andamento più regolare e con meno oscillazioni, e dunque più semplice.

In conclusione, il fenomeno della doppia discesa offre nuove linee guida per l'ottimizzazione dei modelli di apprendimento automatico, suggerendo che, in determinate condizioni, un maggiore numero di parametri può effettivamente migliorare le performance predittive.

Bibliografia

[1] M. Belkin, D. Hsu, S. Ma e S. Mandal. ≪Reconciling modern machine-learning practice and the classical bias–variance trade-off≫. In: *Proceedings of the National Academy of Sciences* 116.32 (ago. 2019), pp. 15849–15854.

[2] M. Bisiacco, G. Pillonetto e S.E. Esculapio. *Sistemi e Modelli*. III. Esculapio, 2024. ISBN: 9788893854344.

[3] K. Gustafsson, M. Danelljan, G. Bhat e T.B. Schön. ≪Energy-based models for deep probabilistic regression≫. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[4] T. Hastie, A. Montanari, S. Rosset e R.J. Tibshirani. ≪Surprises in high-dimensional ridgeless least squares interpolation≫. In: *The Annals of Statistics* 50.2 (2022), pp. 949–986.

[5] S. Haykin. *Neural Networks*. 2nd. Upper Saddle River, NJ: Prentice Hall, 1999.

[6] L. Ljung. *System Identification - Theory for the User*. 2nd. Upper Saddle River, N.J.: Prentice-Hall, 1999.

[7] L. Ljung, C. Andersson, K. Tiels e T.B. Schön. ≪Deep learning and system identification≫. In: *Proceedings of the 21st IFAC World Congress*. Vol. 53. 2020, pp. 1175–1181.

[8] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A.H. Ribeiro e T.B. Schön. ≪Deep networks for system identification: a Survey≫. In: *Automatica* (2024).

[9] A.H. Ribeiro, J. Hendriks, A. Wills e T.B. Schön. ≪Beyond Occam's razor in system identification: double-descent when modeling dynamics≫. In: *Proceedings of the 19th IFAC Symposium on System Identification (SYSID)*. 2021.