Identificazione di sistemi non lineari tramite regressione Gaussiana e applicazione al Wiener-Hammerstein benchmark

Sommario

Si espone un approccio per l'identificazione di sistemi non lineari e la sua applicazione a un sistema reale di tipo Wiener-Hammerstein. L'algoritmo sfrutta la regressione Gaussiana per modellare il sistema come una realizzazione di un campo aleatorio Gaussiano. Esso riduce il problema alla sola determinazione di alcuni iperparametri che forniscono la descrizione completa dell'autocovarianza ad esso associata. Gli iperparametri vengono stimati ottimizzando la loro densità marginale basandosi esclusivamente sui dati di identificazione forniti dal benchmark Wiener-Hammerstein. I risultati sono infine confrontati coi dati appartenenti all'insieme di test dello stesso benchmark per illustrare l'efficacia di questo approccio.

Indice

Capitolo 1

Regressione

1.1 Introduzione

Il problema di apprendere modelli continui ingresso-uscita partendo dai dati empirici osservati, noto col nome di regressione è stato ampiamente studiato. Esistono due approcci classici: il primo è quello di restringere la classe di funzioni cercate, ad esempio prendendo in considerazione solo funzioni lineari degli ingressi; il secondo è quello di fissare a priori una probabilità ad ogni funzione possibile assegnando le probabilità maggiori alle funzioni ritenute più verosimili. Il primo approccio risente di una scarsa capacità predittiva se la funzione obiettivo non può essere modellata sufficientemente bene dalla classe fissata, mentre il secondo presenta il problema che l'insieme di funzioni possibili è infinito e non numerabile. Quest'ultima difficoltà, però, può essere superata nell'ambito dei processi Gaussiani, che forniscono un approccio sofisticato permettendo una facile trattabilità computazionale.

In questo capitolo verrà esposta una procedura per identificare un sistema attraverso un modello lineare, successivamente sarà analizzato un adattamento al caso non lineare.

1.2 Modello lineare Bayesiano

Si definisce *osservazione* la coppia formata da un vettore $\mathbf{x}$ (di dimensione D) di dati di ingresso e dalla corrispondente uscita scalare y. L'insieme di tutte le osservazioni (siano esse n) forma l'insieme: $D = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$, noto come *training set*. L'insieme degli n vettori $\mathbf{x}$ costituisce la matrice X di dimensione $D \times n$, mentre il vettore $\mathbf{y}$ raccoglie tutte le uscite. Si può riscrivere l'insieme delle osservazione come: $D = (X, \mathbf{y})$.

L'analisi Bayesiana introduce un modello di regressione lineare con rumore Gaussiano:

$$y = f(\mathbf{x}) + \varepsilon, \quad f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}, \tag{1.1}$$

dove $\mathbf{x}$ è il vettore d'ingresso, $\mathbf{w}$ è un vettore di parametri (pesi) del modello lineare, f è il valore della funzione, mentre y è il valore osservato. Quest'ultimo differisce dal precedente per via del rumore additivo ε che è notoriamente presente in ogni osservazione empirica. Si assume che il rumore sia caratterizzato da una distribuzione Gaussiana iid[1] a media nulla e varianza σ_n^2, ovvero: $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$.

1.2.1 Inferenza Bayesiana

Per esprimere la conoscenza iniziale posseduta sui parmetri del modello, in ambito Bayesiano, si pone sugli stessi una distribuzione detta "a priori": $p(\mathbf{w})$, nel caso in esame si sceglie una distribuzione a priori Gaussiana a media nulla con matrice di covarianza Σ_p: $w \sim \mathcal{N}(0, \Sigma_p)$.

Sucessivamente, dall'osservazione delle uscite y, si ricava nuova informazione sui parametri. Essa è espressa dalla *verosimiglianza*, la densità di probabilità delle osservazioni dati i parametri: $p(\mathbf{y}|X, \mathbf{w})$

Combinando questa informazione con quella a priori si genera una nuova distribuzione di probabilità denominata "a posteriori" che costituisce il fulcro dell'inferenza Bayesiana. La regola di aggiornamento è fornita dal Teorema di Bayes:

$$posteriori = \frac{verosimiglianza \times priori}{verosimiglianza\ marginale}, \qquad p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)}, \tag{1.2}$$

dove la *verosimiglianza marginale* è una costante di normalizzazione indipendente dai parametri:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w}. \tag{1.3}$$

Grazie all'assunzione di indipendenza, è possibile fattorizzare la distribuzione di probabilità della verosimiglianza:

$$\begin{aligned} p(\mathbf{y}|X, \mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(y_i - \mathbf{x}^\top{}_i \mathbf{w})^2}{2\sigma_n^2}} \\ &= \frac{1}{(2\pi\sigma_n^2)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^\top \mathbf{w}|^2} = \mathcal{N}(X^\top \mathbf{w}, \sigma_n^2 I), \end{aligned} \tag{1.4}$$

Quindi dalla regola di Bayes, tralasciando i termini che non dipendono dai parametri, si ottiene:

[1] con variabili aleatorie indipendenti ed identicamente distribuite

$$
\begin{aligned}
p(\mathbf{w}|\mathbf{y}, X) &\propto e^{-\frac{1}{2\sigma_n^2}(\mathbf{y}-X^\top\mathbf{w})^\top(\mathbf{y}-X^\top\mathbf{w})} e^{-\frac{1}{2}\mathbf{w}^\top\Sigma_p^{-1}\mathbf{w}} \\
&\propto e^{-\frac{1}{2}(\mathbf{w}-\bar{\mathbf{w}})^\top(\frac{1}{\sigma_n^2}XX^\top+\Sigma_p^{-1})(\mathbf{w}-\bar{\mathbf{w}})},
\end{aligned} \tag{1.5}
$$

dove $\bar{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}XX^\top + \Sigma_p^{-1})^{-1}X\mathbf{y}$. Ponendo $A = \sigma_n^{-2}XX^\top + \Sigma_p^{-1}$, si nota che la distribuzione a posteriori è Gaussiana con media $\bar{\mathbf{w}}$ e con matrice di covarianza A^{-1}:

$$
p(\mathbf{w}|\mathbf{y}, X) \sim \mathcal{N}(\bar{\mathbf{w}}, A^{-1}), \tag{1.6}
$$

In fine, per fare una predizione si calcola la media di tutti i valori che i parametri possono assumere, pesandoli in base alla rispettiva probabilità che vengano assunti. La media delle risposte di tutti i possibili modelli lineari, pesata rispetto alla distribuzione a posteriori, fornisce la distribuzione predittiva:

$$
\begin{aligned}
p(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int p(f_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w}|X, \mathbf{y})d\mathbf{w} \\
&= \mathcal{N}(\frac{1}{\sigma_n^2}\mathbf{x}_*^\top A^{-1}X\mathbf{y}, \mathbf{x}_*^\top A^{-1}\mathbf{x}_*).
\end{aligned} \tag{1.7}
$$

dove $f_* = f(\mathbf{x}_*)$, con $\mathbf{x}_*$ punto di cui si vuole stimare il valore. Si noti che la distribuzione predittiva risulta essere anch'essa Gaussiana.

1.2.2 Proiezione nello spazio a maggiori dimensioni

Il modello Bayesiano fin qui presentato assolve bene il compito di identificare funzioni lineari degli ingressi o loro accettabili approssimazioni, ma soffre di un'applicabilità limitata. Una soluzione efficace a questo problema è quella di non applicare il modello direttamente sugli ingressi, bensì in un particolare spazio S^N, di dimensione maggiore ($N > D$), detto spazio delle *feature*. Esso è ottenuto mappando gli ingressi grazie ad una base di apposite funzioni. Il vantaggio è che se la mappatura avviene in modo indipendente dai parametri, il nuovo modello continua ad essere lineare. Il costo di questo procedimento consiste nella scelta delle basi di funzioni.

Sia data la funzione $\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{S}^N$ che mappa un vettore di ingressi $\mathbf{x}$ nella feature $\boldsymbol{\phi}(\mathbf{x})$. L'aggregazione dei vettori $\boldsymbol{\phi}(\mathbf{x})$, generati da tutti gli elementi del training set, costituisce la matrice $\boldsymbol{\Phi}(X)$. Il modello che si ottiene in questo modo risulta:

$$
f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top\mathbf{w}, \tag{1.8}
$$

si noti che ora la dimensione di $\mathbf{w}$ è N. La distribuzione predittiva mantiene la stessa forma, una volta fatte le opportune sostituzioni:

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1}\boldsymbol{\Phi}(X)\mathbf{y}, \boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1}\boldsymbol{\phi}(\mathbf{x}_*)). \tag{1.9}$$

dove $A = \sigma_n^{-2}\boldsymbol{\Phi}(X)\boldsymbol{\Phi}(X)^\top + \Sigma_p^{-1}$. Definendo $K = \boldsymbol{\Phi}(X)^\top\Sigma_p\boldsymbol{\Phi}(X)$ e applicando le sostituzioni: $\boldsymbol{\Phi} = \boldsymbol{\Phi}(X)$, $\boldsymbol{\phi}(\mathbf{x}_*) = \boldsymbol{\phi}_*$ è possibile riscrivere l'equazione precedente in una forma più comoda:

$$\begin{aligned} f_*|\mathbf{x}_*, X, \mathbf{y} \quad &\sim \quad \mathcal{N}(\boldsymbol{\phi}_*^\top\Sigma_p\boldsymbol{\Phi}(K + \sigma_n^2 I)^{-1}\mathbf{y}, \\ &\qquad \boldsymbol{\phi}_*^\top\Sigma_p\boldsymbol{\phi}_* - \boldsymbol{\phi}_*^\top\Sigma_p\boldsymbol{\Phi}(K + \sigma_n^2 I)^{-1}\boldsymbol{\Phi}^\top\Sigma_p\boldsymbol{\phi}_*), \end{aligned} \tag{1.10}$$

Si noti che in questa equazione è richiesto di invertire una matrice di dimensione $n \times n$, mentre nell'equazione (1.9) si deve invertire la matrice A che ha dimensione $N \times N$. Ciò risulta sconveniente dato che, in genere, la dimensione dello spazio delle feature è grande. Si ottiene, quindi, una vantaggiosa riduzione del carico computazionale quando $n < N$.

1.3 Processi Gaussiani

Un processo aleatorio è definito *Gaussiano* se tutti i vettori aleatori, che possono essere costruiti prendendo un numero arbitrario di variabili aleatorie dal processo, sono vettori Gaussiani, cioè possiedono la segunete PDF[2] congiunta:

$$p(\mathbf{a}) = \frac{1}{\sqrt{(2\pi)^N|K|}}e^{-\frac{1}{2}(\mathbf{a}-\mathbf{m})^\top K^{-1}(\mathbf{a}-\mathbf{m})}. \tag{1.11}$$

Questo strumento permette di applicare l'inferenza direttamente nello spazio delle funzioni. Descrivendo una distribuzione di probabilità Gaussiana sulle funzioni stesse e rappresentando con variabili aleatorie il valore della funzione $f(\mathbf{x})$ nel punto $\mathbf{x}$:

$$\begin{aligned} f(\mathbf{x}) \quad &\sim \quad GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \\ m(\mathbf{x}) \quad &= \quad \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') \quad &= \quad \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{1.12}$$

Dove $m(\mathbf{x})$ e $k(\mathbf{x}, \mathbf{x}')$ sono, rispettivamente, la funzione media e coviarianza del processo aleatorio a valori reali $f(\mathbf{x})$; esse, da sole, forniscono una descrizione statistica completa del processo aleatorio.

[2] Funzione Densità di Probabilità

1.3.1 Il modello lineare Bayesiano come processo Gaussiano

Senza difficoltà è possibile ottenere un esempio di processo Gaussiano dal modello Bayesiano esposto precedentemente. In particolare dalle equazioni (1.8), (1.12) e ricordando che per la distribuzione a priori si ha: $\mathbf{w} \sim \mathcal{N}(0, \Sigma_p)$ è facile ricavare quanto segue:

$$\begin{aligned} m(\mathbf{x}) &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}[f(\mathbf{w})] = 0, \\ k(\mathbf{x}, \mathbf{x}') &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}[f(\mathbf{w}\mathbf{w}^\top)]\boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \Sigma_p \boldsymbol{\phi}(\mathbf{x}'). \end{aligned} \tag{1.13}$$

Si consideri ora la funzione covarianza $k(\mathbf{x}, \mathbf{x}')$ nota anche come *kernel*. Dato che Σ_p è definita positiva, è possibile scomporla come: $(\Sigma_p^{\frac{1}{2}})^2$. Esiste, quindi, una funzione $\boldsymbol{\psi}(\mathbf{x}) = \Sigma_p^{\frac{1}{2}} \boldsymbol{\phi}(\mathbf{x})$ grazie alla quale è possibile esprimere il kernel come un prodotto interno: $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x}) \cdot \boldsymbol{\psi}(\mathbf{x}')$.

Se un algoritmo è definito solamente tramite prodotti interni nello spazio degli ingressi, allora può essere elevato nello spazio delle feature semplicemente sostituendo ogni prodotto interno con la funzione covarianza ad esso associata. Grazie a questa procedura, nota come *kernel trick*, non è nemmeno necessario calcolare esplicitamente le funzioni $\boldsymbol{\psi}(\mathbf{x})$ che nell'ambito dei processi Gaussiani possono appartenere ad uno spazio a infinite dimensioni.

1.3.2 Predizione

Lo scopo della regressione è quello di ricavare il vettore $\mathbf{f}_*$ dei valori di uscita generati dal sistema in risposta all'insieme X_* degli ingressi di test. Partendo dall'insieme $D = (X, \mathbf{y})$ delle osservazioni di training e tenedo conto che sono inevitabilmente affette da rumore. Di seguito verrà mostrato come si giunge gradualmente all'espressione della distribuzione a posteriori per il vettore delle uscite $\mathbf{y}_*$.

Predizione senza rumore

Si consideri il caso in cui le osservazioni non siano affette da rumore, ovvero $\mathbf{y}_* = \mathbf{f}_*$ e $D = \{(\mathbf{x}_i, f_i) | i = 1, ..., n\}$. Di conseguenza la distribuzione congiunta a priori delle uscite di training e di quelle di test è:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \tag{1.14}$$

A questo punto, restringendo la distribuzione congiunta alle sole funzioni che soddisfano i dati di training, si ottiene la distribuzione a posteriori. Per fare ciò è sufficiente condizionare la distribuzione a priori rispetto alle osservazioni:

$$\begin{aligned} \mathbf{f}_*|X_*, X, \mathbf{f} &\sim \mathcal{N}(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ &\quad K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)). \end{aligned} \tag{1.15}$$

Nonostante non si presenti mai nella realtà, l'assunzione dell'assenza di rumore è ragionevole in alcuni casi, per esempio nelle simulazioni al computer.

Predizione con rumore

Il passo successivo è quello di ricavare la distribuzione a posteriori dei valori $\mathbf{f}_*$ delle funzioni considerando l'effetto introdotto dal rumore Gaussiano addititvo iid $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. In questo caso la distribuzione a priori sulle osservazioni diventa: $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K(X, X) + \sigma_n^2 I)$. Grazie all'indipendenza assunta, l'unica differenza che compare rispetto all'equazione (1.14) è l'aggiunta di una matrice diagonale nella porzione della matrice covarianza relativa alle osservazioni di training:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \tag{1.16}$$

Di conseguenza l'equazione (1.15) così si aggiorna:

$$\begin{aligned} \mathbf{f}_*|X_*, X, \mathbf{y} &\sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)), \quad \text{dove} \\ \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\mathbf{f}_*|X_*, X, \mathbf{y}] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}, \\ cov(\mathbf{f}_*) &= K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*). \end{aligned} \tag{1.17}$$

Quest'ultima equazione evidenzia una proprietà della distribuzione Gaussiana: la covarianza non dipende dalle uscite osservate, ma esclusivamente dagli ingressi. Senza difficoltà si può riscrivere l'equazione predittiva per un unico punto di test con n osservazioni:

$$\bar{f}_* = K(\mathbf{x}_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}, \tag{1.18}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - K(\mathbf{x}_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, \mathbf{x}_*). \tag{1.19}$$

Il predittore lineare $\bar{f}_*$ dell'equazione (1.18) può essere visto come una combinazione lineare di n funzioni covarianza ognuna centrata su un punto di training:

$$\bar{f}_* = \Sigma_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}_*), \quad \text{con } \boldsymbol{\alpha} = [K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}. \tag{1.20}$$

Il processo Gaussiano definisce una distribuzione Gaussiana congiunta su tutte le varibili aleatorie associate all'insieme X e la base di funzioni che rappresenta

può possedere una cardinalità infinita, ma per fare predizioni sul punto $\mathbf{x}_*$ è sufficiente concentrarsi esclusivamente sulla distribuzione a $(n+1)$ dimensioni definita dal punto di test e dalle n osservazioni.

In fine possiamo calcolare anche la distribuzione a posteriori sui valori di uscita osservati $\mathbf{y}_*$, aggiungendo l'apporto del rumore: $\mathbf{y}_*|X_*, X, \mathbf{y} \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*) + \sigma_n^2 I)$.

Capitolo 2

Selezione del modello

2.1 Funzione covarianza

La funzione covarianza rappresenta sicuramente l'elemento più importante della regressione Gaussiana, in quanto specifica alcune delle proprietà fondamentali della funzione che si vuole identificare. Essa racchiude nella sua espressione i concetti di *vicinanza* e *somiglianza* che sono aspetti essneziali dell'apprendimento supervisionato: è naturale aspettarsi che punti di ingresso vicini conducano a valori simili delle risposte. Risulta necessario, dunque, valutare con attenzione la scelta della funzione covarianza da adottare.

In generale la funzione covarianza può essere una qualsiasi funzione capace di generare una matrice di covarianza definita non negativa per ogni insieme di vettori d'ingresso X. Non è sempre banale trovare funzioni che soddisfino questa richiesta, in compenso esistono alcune proprietà comuni a diversi tipi di funzioni covarianza.

Una funzione covarianza è detta *stazionaria* se è funzione di $\mathbf{x} - \mathbf{x}'$, ovvero è invariante rispetto alle traslazioni nello spazio degli ingressi. Inoltre, se essa è esclusivamente funzione di $|\mathbf{x} - \mathbf{x}'|$, allora è detta *isotropica*, nel senso che è invariante rispetto ai movimenti rigidi. Queste funzioni sono anche note come *funzioni radiali* (RBF[1]) in quanto dipendono esclusivamente da $r = |\mathbf{x} - \mathbf{x}'|$.

Se, invece, la funzione covarianza dipende unicamente dal prodotto $\mathbf{x} \cdot \mathbf{x}'$, è nota come funzione "*dot product*". Essa è caratterizzata dalla proprietà di essere invariante rispetto alla rotazione delle coordinate nell'origine, ma non alle traslazioni.

Continuità in media quadratica

Data la sequenza di punti $\mathbf{x}_1, \mathbf{x}_2, ...$ sia $\mathbf{x}_* \in \mathbb{R}^D$ punto di accumulazione per la sequenza precedente. Allora un processo $f(\mathbf{x})$ è *continuo in media quadratica*

[1] Radial Basis Function

in $\mathbf{x}_*$ se:

$$\lim_{k\to\infty} \mathbb{E}[|f(\mathbf{x}_k) - f(\mathbf{x}_*)|^2] = 0. \tag{2.1}$$

Dato il sottoinsieme $A \subseteq \mathbb{R}^D$, il processo $f(\mathbf{x})$ è detto continuo in media quadratica in A se il limite (2.1) vale $\forall \mathbf{x}_* \in A$.

Un campo aleatorio è continuo in media quadratica in $\mathbf{x}_*$ se e solo se la sua funzione covarianza $k(\mathbf{x}, \mathbf{x}')$ è continua nel punto $\mathbf{x} = \mathbf{x}' = \mathbf{x}_*$. Per le funzioni covarianza stazionarie questo requisito si riduce alla continuità in $k(\mathbf{0})$.

Differenziabilità in media quadratica

La *derivata in media quadratica* del processo $f(\mathbf{x})$ lungo l' i-esima direzione è definita dal seguente limite, se esiste:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \lim_{h\to\infty} \frac{f(\mathbf{x}_* + h\mathbf{e}_i) - f(\mathbf{x})}{h}, \tag{2.2}$$

dove il limite è inteso come limite in media quadratica e $\mathbf{e}_i$ è il versore dell'i-esima direzione.

La funzione covarianza della derivata $\frac{\partial f(\mathbf{x})}{\partial x_i}$ è data da: $\frac{\partial^2 k(\mathbf{x},\mathbf{x}')}{\partial x_i \partial x_i'}$.

La funzione covarianza Gaussiana

La funzione covarianza più largamente adottata in letteratura è senza dubbio quella Gaussiana (assieme alle sue varianti):

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2l^2}}. \tag{2.3}$$

Questa funzione covarianza è infinitamente differenziabile, ovvero il processo Gaussiano ad essa associato possiede le derivate in media quadratica di ogni ordine e questo corrisponde ad una generale assunzione di liscezza.

2.2 Gestione degli iperparametri

La scelta della funzione covarianza fissa quali proprietà caratterizzano la funzione f (per esempio il fatto di essere una funzione liscia), ma generalmente non vincolano alcuni aspetti quantitativi (quanto è liscia la funzione dell'esempio precedente) la cui impostazione viene, invece, affidata a specifici parametri liberi (di solito sono in numero ridotto) detti *iperparametri*. Risulta quindi essenziale introdurre un metodo[2] in grado di scegliere, oltre alla funzione covarianza, anche i valori dei suoi iperparametri a partire dai dati.

[2]Talvolta noto come "*training*" del processo Gaussiano

Per fare un esempio, la funzione covarianza Gaussiana (2.3) può essere riscritta mettendo in evidenza alcuni iperparametri:

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\top M(\mathbf{x}-\mathbf{x}')}, \tag{2.4}$$

dove $\boldsymbol{\theta} = (\{M\}, \lambda^2)^\top$ è un vettore che contiene tutti gli iperparametri con $\{M\}$ che indica i parametri nella matrice simmetrica M.

Per indirizzare la scelta del modello tra le infinite possibilità offerte dalle diverse combinazioni degli iperparametri, vengono seguiti tre principi di base:

1. Calcolare la probabilità del modello sulla base dei dati osservati
2. Stimare l'*errore di generalizzazione*
3. Imporre un limite all'*errore di generalizzazione*

dove per errore di generalizzazione[3] si intende l'errore medio su valori di uscita non osservati. Esso esprime la capacità di generalizzare la funzione basandosi solo su alcuni esempi.

2.2.1 Criterio Bayesiano di selezione del modello

Il procedimento utilizza l'inferenza Bayesiana su tre livelli sovrapposti che, partendo dal livello più basso, prendono in esame rispettivamente prametri $\mathbf{w}$, gli iperparametri $\boldsymbol{\theta}$ (che a loro volta controllano la distribuzione dei parametri del livello inferiore) e infine viene considerato un insieme discreto di diverse strutture $\mathcal{H}_i$ da adottare come modello.

Al livello inferiore, la distribuzione a posteriori sui parametri è data dalla ragola di Bayes (1.2):

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)}, \tag{2.5}$$

dove $p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)$ è la *verosimiglianza* e $p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)$ è la distribuzione di probabilità a priori sui parametri che racchiude le informazioni sui parametri possedute prima di osservare i dati. Questa conoscenza viene combinata con quella fornita dai dati attraverso la verosimiglianza per ottenere la distribuzione a posteriori. Al denominatore c'è la *verosimiglianza marginale* che è una costante di normalizzazione indipendente dai parametri:

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w}. \tag{2.6}$$

Passando al livello intermedio si procede in modo analogo per esprimere la distribuzione a posteriori sugli iperparametri. In questo caso la verosimiglianza è costituita dalla verosimiglianza marginale del livello precedente:

[3] Da non confondere con l'errore di training

$$p(\boldsymbol{\theta}|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)}, \tag{2.7}$$

dove $p(\boldsymbol{\theta}|\mathcal{H}_i)$ è la distribuzione a priori sugli iperparametri, mentre la costante di normalizzazione è data da:

$$p(\mathbf{y}|X, \mathcal{H}_i) = \int p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)d\boldsymbol{\theta}. \tag{2.8}$$

Per finire, al livello superiore, si calcola la distribuzione a posteriori sui modelli:

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)}, \tag{2.9}$$

dove la verosimiglianza marginale in questo caso è data da:

$$p(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)$$

sommata su tutti i modelli presi in considerazione. Solitamente la distribuzione a priori sui modelli $p(\mathcal{H}_i)$ adottata è "piatta" ovvero tale da non favorire un modello rispetto ad un altro prima di osservare i dati.

Sfortunatamente capita che il calcolo degli integrali sopra riportati (in particolar modo quello dell'equazione (2.8)) non sia sempre agevole o che non sia nemmeno trattabile analiticamente. In questi casi si ricorre a tecniche di approssimazione o metodi Monte Carlo.

Una proprietà caratteristica della verosimiglianza marginale è quella di includere un bilanciamento automatico tra adeguatezza e complessita del modello. In questo modo tende a favorire maggiormente il più semplice modello coerente con i dati. Tutto ciò la rende uno strumento prezioso per la selezione del modello e il fatto che tale bilanciamento sia automatico non richiede di introdurre un ulteriore parametro che lo imposti.

2.2.2 Cross-validation

I criteri di selezione del modello basti sull'errore di generalizzazione, si servono spesso di metodi noti col nome di *cross-validation*. Il principio è quello di dividere l'insieme di training in due parti di cui solo una (la più corposa) viene adoperata per l'apprendimento, mentre l'altra è dedicata ad una successiva fase di convalida (*validation*). Durante questa fase viene costantemente controllata la qualità dei risultati ottenuti e la misura delle performance serve come strumento per ricavare l'errore di generalizzazione e, di conseguenza, per selezionare il modello più adeguato. Per fare in modo di non ridurre troppo i dati a disposizione per il training, solitamente, si adotta la strategia del *k-fold cross-validation*. La procedura consiste nel fare k copie dell'intero insieme di training, ciascuna divisa in k parti. Per ogni copia si sceglie una parte (sempre

diversa) come validation set e si usano le altre per la fase di apprendimento. In questo modo si riduce la parte di dati sacrificata per la fase di validation e si evita l'*overfitting*, al prezzo di ripetere l'operazione k volte.

2.2.3 Selezione di modelli a regressione Gaussiana

In precedenza si è accennato al fatto che il calcolo degli integrali sullo spazio dei parametri è problematico per diversi modelli. Fortunatamente quelli che si basano sulla regressione a processi Gaussiani e con rumore Gaussiano sono trattabili analiticamente e garantiscono una buona flessibilità di applicazione.

In questo ambito, l'applicazione dell'equazione (2.5) per la distribuzione predittiva su parametri si traduce nelle equazioni (1.10) e (1.17) già ricavate in precedenza rispetto allo spazio dei pesi e rispetto a quello delle funzioni. Mentre la verosimiglianza marginale è data da:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}, \tag{2.10}$$

e il termine "marginale" è riferito alla marginalizzazione rispetto alle funzioni incognite f. Assumendo la distribuzione a priori e la verosimiglianza come Gaussiane, rispettivamente: $\mathbf{f}|X \sim \mathcal{N}(\mathbf{0}, K)$ e $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$ si ottiene $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$ ed esprimendola in forma logaritmica:

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^\top (K+\sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log(2\pi) = \log p(\mathbf{y}|X, \boldsymbol{\theta}), \tag{2.11}$$

intendendo con K la matrice covarianza per i valori delle funzioni f non affetti da rumore. Questa equazione può essere interpretata nel modo seguente: il primo termine dipende sia dalla funzione covarianza che dalle uscite osservate, esso rappresenta la capacità del modello di spiegare i dati, un'indicazione di quanto sia in grado di adattarsi a questi. Il secondo termine, dipendente esclusivamente dal modello, costituisce una penalizzazione attribuita alla sua complessità, mentre l'ultimo termine è una costante di normalizzazione.

Infine la procedura per impostare i valori ottimi degli iperparametri prevede di calcolare il massimo della verosimiglianza marginale sfruttando i metodi di ottimizzazione basati sul gradiente. Essi richiedono il calcolo delle derivate parziali:

$$\frac{\partial}{\partial \theta_j}\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top K^{-1}\frac{\partial K}{\partial \theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{\partial K}{\partial \theta_j}), \tag{2.12}$$

il tempo richiesto per eseguire questo calcolo è dominato dall'inversione della matice K che ha dimensione $n \times n$ e porta ad una complessità pari a $\mathcal{O}(n^3)$. Fortunatamente è necessario invertire una sola volta la matrice, mentre le derivate interne devono essere calcolate per ogni iperparametro, ma richiedono un tempo inferiore di un ordine: $\mathcal{O}(n^2)$.

Capitolo 3

Identificazione di sistemi non lineari

3.1 Presentazione dell'algoritmo

In questo capitolo viene presentato l'algoritmo per l'identificazione dei sistemi non lineari su cui si basa l'intero lavoro che è oggetto di questo studio. Esso consiste in un particolare approccio per l'individuazione della matrice di covarianza più adatta per generare un modello del sistema in esame [2].

L'intento è quello di convessificare il problema di identificazione sfruttando la regressione Gaussiana e ponendo una opportuna distribuzione Gaussiana a priori su uno spazio di funzioni infinito-dimensionale. Per fare ciò si definisce una classe di specifici kernel studiati appositamente per l'indentificazione di sistemi non lineari. In particolare ogni kernel è costituito da una mistura di kernel Gaussiani che è nota a meno di alcuni iperparametri che tengono conto delle interazioni degli ingessi e delle uscite pregressi. Per individuare il kernel più adatto si valuta la sua probabilità a posteriori secondo il criterio Bayesiano di selezione del modello. Infine il modello non lineare è ottenuto risolvendo un problema variazionale di Tikhonov.

3.2 Formulazione del problema

Si consideri un sistema dinamico non lineare a tempo discreto. Sia $\mathbf{u}^t = [u_{t-1}\ u_{t-2}\ u_{t-3}...]$ un vettore noto che costituisce l'ingresso del sistema e sia $\mathbf{y}^t = [y_{t-1}\ y_{t-2}\ y_{t-3}...]$ un vettore di dati di uscita affetto da rumore. Il modello non lineare del sistema in esame è:

$$y_t = f(\mathbf{y}^t, \mathbf{u}^t) + \varepsilon \tag{3.1}$$

dove f è la funzione non lineare incognita e ε è il rumore Gaussiano bianco con varianza σ_n^2. Il problema consiste nello stimare f dai dati di ingresso e di uscita osservati.

Si noti come il vettore $\mathbf{u}^t$ sia costituito dalla sequenza cronologica dei valori di un unico ingresso e come il valore assunto dalla funzione f dipenda sia dalla sequenza temporale degli ingressi passati che da quella delle uscite. Quindi le coppie (y^t, u^t) rappresentamo i punti di ingresso, mentre l'insieme di training è formato da $\{(y^t, u^t), y_t\}$.

3.3 Struttura del kernel

Lo scopo è quello di ricostruire l'ipersuperficie $f : \mathbb{R}^{\infty\times\infty} \mapsto \mathbb{R}$ considerandola come una realizzazione di un campo aleatorio Gaussiano con kernel (autocovarianza) $k(\cdot,\cdot)$ i cui argomenti sono costituiti dalla sequenza (y^t, u^t) e dalla sequenza diversa (y'^t, u'^t).

Si consideri il kernel Gaussiano:

$$k((y^t, u^t), (y'^t, u'^t)) = \lambda^2 e^{\sum_{i=1}^{\infty}\left[-\frac{(y_{t-i}-y'_{t-i})^2+(u_{t-i}-u'_{t-i})^2}{l^2}\right]}, \tag{3.2}$$

con λ^2 e l^2 scalari da determinare. Sebbene il kernel Gaussiano sia praticamente il più usato per gli approcci non parametrici di questo tipo, non è particolarmente appropriato per l'identificazione di sistemi non lineari. Si può notare, infatti, che esso tende a zero per la maggior parte dei valori d'ingresso a causa della natura infinito-dimensionale dello spazio degli ingressi. Inoltre non assume che l'uscita venga influenza in modo diverso da input con un differente grado di vicinanza (in questo caso temporale). Com'è noto, nel mondo reale, gli effetti sui sistemi fisici di cause pregresse diventano sempre più trascurabili col passare del tempo t.

Per tenere conto di questi aspetti, viene proposto una nuova struttura che si basa su una composizione di kernel Gaussiani. Per comodità è preferibile accorpare le coppie di input in un unico vettore:

$$\begin{aligned} x_1 &= [y_{t-1}\ u_{t-1}] & x'_1 &= [y'_{t-1}\ u'_{t-1}] \\ x_2 &= [y_{t-2}\ u_{t-2}] & x'_2 &= [y'_{t-2}\ u'_{t-2}] \\ &\vdots & &\vdots \end{aligned} \tag{3.3}$$

con $x_i, x'_i \in \mathbb{R}^2$, $i \in \mathbb{N}$. In questo modo il modello diventa: $y = f(\mathbf{x}) + \varepsilon$, dove $\mathbf{x} = [x_1\ x_2\ x_3 ...]$.

Il nuovo kernel, dunque, è definito come segue:

$$k(\mathbf{x}, \mathbf{x}'; p) = \sum_{h=1}^{\infty} \beta_h e^{\left[-\frac{\sum_{i=1}^{p} ||x_{i+h-1}-x'_{i+h-1}||^2}{l^2}\right]}, \tag{3.4}$$

dove $\beta_h = \lambda_1^2 e^{-j\lambda_2^2}$, con $\lambda_1, \lambda_2 > 0$. In questo modo si permettono interazioni tra i dati pregressi in istanti di tempo contigui fino a un massimo di p ordini e allo stesso tempo si privilegiano quelli più recenti. Dato p che ne fissa la struttura, il kernel è individuato a meno di tre iperparametri: l^2, λ_1^2 e λ_2^2 che regola lo smorzamento degli effetti degli input più lontani nel tempo.

3.4 Stima degli iperparametri

Sia $\boldsymbol{\theta} = [l^2, \lambda_1^2, \lambda_2^2, \sigma_n^2]$ il vettore che raccoglie gli iperparametri indicati in precedenza oltre alla varianza del rumore. I loro valori, assieme a quello di p, sono generalmente ignoti per le applicazioni reali e quindi devono essere stimati secondo l'approccio Bayesiano già trattato.

L'idea è quella di individuare il valore di p che massimizza la probabilità a posteriori del modello ad esso associato. Per far questo è necessario trovare prima i valori ottimi cioè tali da massimizzare il seguente obiettivo:

$$p(\mathbf{y}|p) = \int p(\mathbf{y}, f|\boldsymbol{\theta}, p)p(\boldsymbol{\theta})df\,d\boldsymbol{\theta}, \tag{3.5}$$

dove $p(\boldsymbol{\theta})$ è una distribuzione a priori indipendente da p e contenente scarse informazioni che rappresentano esclusivamente la non negatività dei suoi componenti. Successivamente, marginalizzando rispetto a f si ottiene:

$$p(\mathbf{y}|\boldsymbol{\theta}, p) = \frac{e^{-\frac{1}{2}\mathbf{y}^\top \Sigma_y^{-1}\mathbf{y}}}{\sqrt{2\pi|\Sigma_y|}}, \tag{3.6}$$

dove la matrice Σ_y ha dimensione $n \times n$ e l'elemento corrispondente alla posizione (i, j) è così definito:

$$\Sigma_y(i, j) = k((y^i, u^i), (y'^j, u'^j); p) + \sigma_n^2 \delta_{ij}, \tag{3.7}$$

dove δ è la delta di Kroenecker pari a 1 se e solo se $i = j$.

Nonostante l'integrale (3.5) sia affrontabile al calcolatore, essa risulta eccessivamente dispendiosa in termini di risorse ed è preferibile ricorrere ad una soluzione approssimata[1]. Si procede, dunque, a massimizzare la verosimiglianza marginale dell'equazione (3.6) rispetto agli iperparamteri di $\boldsymbol{\theta}$. In particolare si cerca il vettore $\boldsymbol{\theta}$ ottimo che ne minimizzi il logaritmo negativo:

$$\hat{\boldsymbol{\theta}}_p = \arg\min_{\boldsymbol{\theta}} J_p(\boldsymbol{\theta}), \qquad J_p(\boldsymbol{\theta}) := -\log p(\mathbf{y}|\boldsymbol{\theta}, p). \tag{3.8}$$

Infine, la struttura del kernel migliore per modellare il sistema è individuata dal valore di p tale che:

$$\hat{p} = \arg\min_{p} J_p(\hat{\boldsymbol{\theta}}_p). \tag{3.9}$$

[1] Questa approssimazione è nota come Type II maximum likelihood (ML-II)

3.5 Determinazione del modello non lineare

Nei paragrafi precedenti sono stati presentati i metodi utilizzati per scegliere il modello più adatto a rappresentare il sistema che deve essere identificato. Una volta analizzati i dati di training formati dalle coppie di input e output disponibili, dopo aver stimato gli iperparametri e ricavato la struttura del kernel, non resta altro che determinare il modello non lineare $\hat{f}$ [3].

Siano $\hat{p}$ e $\hat{\boldsymbol{\theta}}_p$ i valori ottimi secondo le equazioni (3.9) e (3.8), allora sfruttiamo il risultato (1.20) per ricavare:

$$\hat{f}(\cdot) = \sum_{i=1}^{n} \alpha_i k((y^i, u^i), \cdot; \hat{p}), \qquad \text{con } \boldsymbol{\alpha} = \Sigma_y^{-1}\mathbf{y}. \tag{3.10}$$

Capitolo 4

Wiener-Hammerstein benchmark

4.1 Introduzione

In questo capitolo viene presentato un benchmark studiato appositamente per testare i metodi di modellazione di sistemi non lineari e misurarne le prestazioni. Per la precisione si tratta del Wiener-Hammerstein benchmark [4].

L'intento del benchmark è quello di fornire un utile strumento per approfondire la comprensione delle diverse strategie di identificazione. Proprio per questo motivo, il sistema che propone è affetto da un rumore molto leggero, in questo modo l'attenzione è focalizzata sulla capacità di modellare adeguatamente il comportamento non lineare del sistema, piuttosto che sull'abilità nella rimozione del rumore. Infine gli autori sottolineano la natura non competitiva del test proposto che deve essere considerato un mezzo per confrontare e comparare le varie tecniche possibili anche nell'eventualità che falliscano nel tentativo di identificazione.

4.2 Il sistema non lineare

Il sistema che deve essere identificato è un sistema non lineare con una struttura di tipo Wiener-Hammerstein creato da Gerd Vandersteen.

Un sistema dinamico non lineare a blocchi è detto di tipo Wiener-Hammerstein quando è costituito da un blocco non lineare statico ($f[\cdot]$) preceduto e seguito da due blocchi lineari dinamici (rispettivamente $G_1(s)$ e $G_2(s)$). Solo i segnali di ingresso ($u(t)$) e di uscita ($y(t)$) del sistema sono misurabili, mentre il segnale entrante e quello uscente dal blocco centrale non sono accessibili.

Il primo blocco è costituito da un filtro di Chebyshev del terzo ordine, mentre

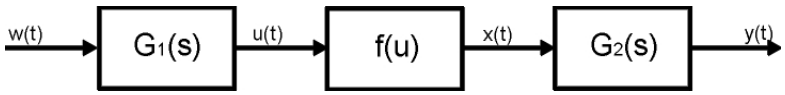

Figura 4.1: Schema di sistema dinamico non lineare a blocchi di tipo Wiener-Hammerstein

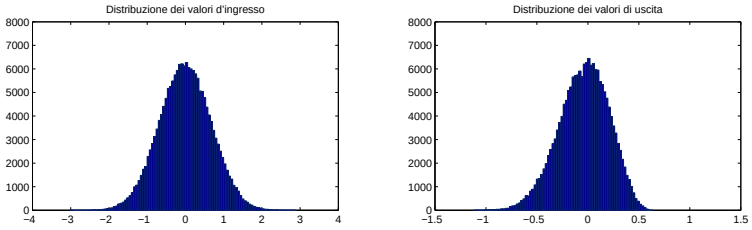

Figura 4.2: Distribuzioni delle ampiezze dei segnali di imput (a sinistra) e di output (a destra)

l'ultimo è un particolare filtro di Chebyshev inverso, sempre del terzo ordine, ma in grado di complicare sensibilmente l'identificazione.

Il blocco centrale, invece, realizza la non linearità grazie ad un circuito dotato di un diodo.

4.3 Dati disponibili

Un generatore di forme d'onda fornisce il segnale d'ingresso, che una volta filtrato, presenta una banda limitata. Sia il segnale d'ingresso che quello di uscita vengono campionati alla frequenza di 51200 Hz. Mentre la distribuzione di ampiezza del segnale di input è di tipo Gaussiano, quella di output è visibilmente asimmetrica.

Come già accennato, il rapporto segnale-rumore è elevato con un livello di disturbo circa 70 dB inferiore al segnale utile.

I dati messi a disposizione sono costituiti da una sequenza di ingresso e una di uscita di uguale lunghezza pari a 188000 campioni. Di questi, i primi 100000 sono assegnati all'insieme di training, mentre i restanti 88000 costituiscono l'insieme di test. Com'è facile immaginare, nella fase di stima dei parametri e di selezione del modello è possibile servirsi esclusivamente del primo insieme di dati.

L'obbiettivo è quello di sfruttare i dati messi a disposizione per identificare un modello lineare da utilizzare successivamente per simulare l'uscita del sistema e

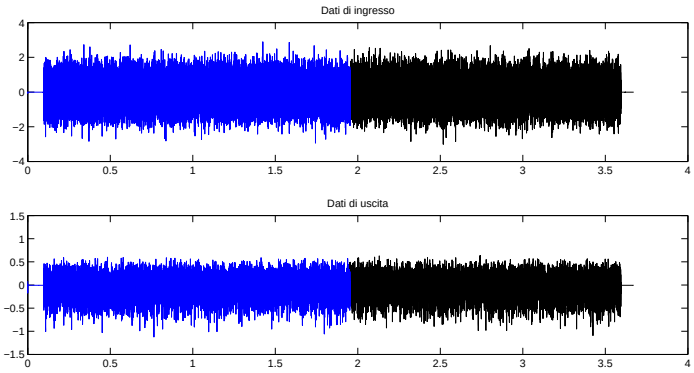

Figura 4.3: Sequenze dei segnali di input (sopra) e di output (sotto). La parte colorata di blu indica il traning set, mentra quella di colore nero indica il test set.

confrontarla con i dati dell'insieme di test. Viene, inoltre, specificato che l'uscita corrente del sistema non dipende dalle uscite pregresse.

4.4 Indici di errore

Per finire, è richiesto di presentare alcuni dati sulle prestazioni in modo da favorire il confronto con risultati ottenuti da altri. E precisamente sono:

- il valore medio dell'errore di simulazione$e_{sim}(t)$: $\mu_t = \frac{1}{87000} \sum_{t=101001}^{188000} e_{sim}(t)$,
- la deviazione standard dell'errore: $s_t = \sqrt{\frac{1}{87000} \sum_{t=101001}^{188000} (e_{sim}(t) - \mu_v)^2}$,
- il valore quadratico medio dell'errore: $e_{RMSt} = \sqrt{\frac{1}{87000} \sum_{t=101001}^{188000} e_{sim}^2(t)}$,

oltre agli stessi indici riferiti ai dati di training calcolati sommando per $t \in [1001, 100000]$.

Capitolo 5

Applicazione del benchmark all'algoritmo

In questo capitolo sarà presentata l'applicazione del Wiener-Hammerstein benchmark all'algoritmo esposto nel capitolo 3. In paricolare si tratta dell'adattamento di un'implementazione realizzata in MATLAB.

5.1 Descrizione dell'implementazione

Innanzitutto il benchmark specifica che l'uscita del sistema da identificare è funzione esclusivamente degli ingressi passati. Quindi il codice è stato semplificato per trattare esclusivamente questo caso, eliminando la dipendenza dalle uscite pregresse.

L'algoritmo carica in due vettori separati le sequenze di dati (di ingesso e di uscita) fornite dal benchmark e poi imposta i valori di una serie di parametri:

- **n** e **nt**: dimensioni del training set X e del test set X_*. Non risulta possibile utilizzare come training set l'intero insieme di dati messo a disposizione. Come si vedrà più avanti, infatti, l'algoritmo utilizza matrici che hanno una dimensione proporzionale al quadrato di quella dell'insieme di training e che occupano, quindi, uno spazio troppo grande per la memoria di un normale computer. Di conseguenza, per il vettore di campioni del segnale di ingresso e per quello delle osservazioni di uscita, si adottano due sequenze formate rispettivamente da n e nt dati. In particolare, nelle prove che sono state eseguite, si sono spesso adottati i valori: $n = 1000$ e $nt = 800$. Questa scelta non si è rivelata particolarmente restrittiva, in quanto sono stati ottenuti comunque buoni risultati. In aggiunta si è preferito trascurare i primi 5000 campioni di training in quanto falsati dal transitorio.

- **np**: lunghezza del predittore. Nell'equazione (3.4) il kernel era costituito dalla somma di infiniti termini esponenziali; per ovvi motivi non è computazionalmente possibile realizzare un algoritmo che tenga conto di infiniti contributi di questo tipo. Considerando, però, che ogni addendo della sommatoria (3.4) viene smorzato dal fattore β_h in maniera sempre più marcata al crescere dell'indice h, ci si accorge che da un certo $h = \bar{h}$ in poi gli apporti aggiuntivi sono trascurabili e quindi si può fare a meno di calcolarli. Arrestando la sommatoria al termine $\bar{h} = np - p + 1$ si ottiene la seguente espressione:

$$k(\mathbf{x}, \mathbf{x}'; p) = \sum_{h=1}^{np-p+1} \beta_h e^{\left[-\frac{\sum_{i=1}^{p} ||x_{i+h-1} - x'_{i+h-1}||^2}{l^2}\right]}. \tag{5.1}$$

  In questo modo np, rappresenta quanto lontanto si va nel passato per misurare gli effetti degli ingressi.

- **lp**: numero di strutture diverse. Questo parametro stabilisce il numero di modelli che vengono costruiti e che successivamente dovranno essere valutati scegliendo quello ottimo in base all'equazione (3.9). Ogni modello è caratterizzato da una struttura del kernel diversa, nel senso che per ogni versione cresce il valore del parametro p (in un intervallo da 1 a lp) della formula (5.1).

- **red**: dimensione del sottoinsieme di dati utilizzato per stimare i parametri. Già nel primo punto di questo elenco è stato sottolineato che le dimensioni delle matrici trattate dall'algoritmo creano problemi per via dell'occupazione memoria. Anche il tempo di calcolo, però, è un fattore importante e strattamente legato alla dimensione della matrice Σ_y nell'equazione (3.6) che deve essere invertita un numero elevato di volte. Perciò, per la stima degli iperparametri, si è deciso di utilizzare un sottoinsieme del dataset ancora più ridotto (tipicamente $red = 100$). Questa scelta è suggerita dal fatto che la precisione necessaria per trovare i valori ottimi per gli iperparametri può essere inferiore a quella richiesta per determinare i valori della funzione f una volta stabilito il kernel da utilizzare.

5.1.1 Costruzione del kernel

Con queste impostazioni il programma prodece con la stima degli lp kernel diversi. Per ogni processo di stima costruisce una matrice U di dimensioni $np \times n$ dove ogni colonna rappresenta una sequenza di np ingressi formata da quello al tempo $t + np - 1$ e da quelli che lo precedono (dove t è l'indice colonna della matrice U).

Dopodiché si genera una nova matrice, chiamata K, di dimensioni $n \times n \times np$. Di seguito si espone il procedimento per calcolare il (i,j,h)-esimo elemento di K.

A partire dalla matrice U vengono estratte due colonne (i e j) e per entrambe si selezionano p elementi contigui che formano le coordinate dei punti $\mathbf{x}_h =$

$(x_h, ...x_{h+p})$ e $\mathbf{x}'_h = (x'_h, ...x'_{h+p})$ al numeratore dell'esponente dell'equazione (5.1). Da queste coordinate si calcola il fattore $e^{-||(\mathbf{x}_h - \mathbf{x}'_h)^2||}$ che viene inserito nella h-esima posizione del vettore $\mathbf{k}_{ij} = (k_{i,j,*})$. Questa operazione viene ripetuta per ogni valore dell'indice h da 1 a $np - p + 1$ e per ogni combinazione delle colonne i e j fino a riempire completamente la matrice K.

A questo punto K contiene i fattori esponenziali (a parte $e^{l^{-2}}$) di tutti gli addendi dell'equzione (5.1) . Resta solo da stimare gli iperparametri prima di poter procedere alla sommatoria.

5.1.2 Stima degli iperparametri

Per la stima degli iperparamtri è necessario calcolare il minimo della funzione $J_p(\boldsymbol{\theta})$. Nel programma essa viene implementata nel modo seguente. A partire dal vettore $\boldsymbol{\theta} = [l^2, \lambda_1^2, \lambda_2^2, \sigma_n^2]$ si completa la costruzione del kernel. Come anticipato in precedenza, si utilizza solamente una porzione della matrice K, ovvero $K_{red} = [k_{i,j,h}]_{i=1,...,red;j=1,...,red}$. In ogni vettore $\mathbf{k}_{ij}$ di questa matrice si sommano tutti gli h elementi pesandoli per il fattore di smorzamento $\beta_h = \lambda_1^2 e^{-j\lambda_2^2}$ e moltiplicandoli per $e^{l^{-2}}$ secondo la sommatoria (5.1). Sia KK la matrice che raccoglie i risultati di questa operazione, essa ha dimensioni $red \times red$. Ad essa si devono sommare gli effetti del rumore Gaussiano per ottenere la matrice $\Sigma_y = KK + \sigma_n^2 I$.

Anzichè procedere direttamente all'inversione della matrice Σ_y, si utilizza un procedimento più rapido per ottenere il risultato dell'equazione (3.6). La matrice Σ_y è simmetrica e definita positiva per come è stata definita l'autocovarianza dell'equazione (3.4), quindi si può applicare la *decomposizione di Cholesky* che stabilisce: $\Sigma_y = LL^\top$. Se $\mathbf{a}$ è il vettore che risolve il sistema $L\mathbf{a} = \mathbf{y}$, si ha che $\mathbf{a} = L^{-1}\mathbf{y}$. Quindi è possibie riscrivere il seguente prodotto:

$$\begin{aligned} \mathbf{y}^\top \Sigma_y^{-1} \mathbf{y} &= \mathbf{y}^\top [LL^\top]^{-1} \mathbf{y} = \mathbf{y}^\top [L^\top]^{-1} [L]^{-1} \mathbf{y} \\ &= \mathbf{y}^\top [L^{-1}]^\top [L]^{-1} \mathbf{y} = \mathbf{a}^\top \mathbf{a}. \end{aligned} \tag{5.2}$$

L'espressione del logaritmo negativo della verosimiglianza marginale dell'equazione (3.6) si può ricavare facilmente da:

$$J_p(\boldsymbol{\theta}) = \frac{1}{2}\mathbf{a}^\top \mathbf{a} + \sum_{i=1}^{red} (\log L_{i,i} + \frac{1}{2} \log 2\pi), \tag{5.3}$$

che costituisce il risultato restituito dalla funzione.

Attraverso il comando MATLAB "*fminsearch*()" il programma calcola per quali valori del vettore $\boldsymbol{\theta}$ la funzione precedente restituisce il valore minimo (a partire da un vettore iniziale scelto arbitrariamente). Il calcolo necessita di effettuare numerose valutazioni della funzione stessa e quindi richiede di compiere molte volte l'inversione di Σ_y, l'utilizzo di un valore basso per il parametro red, unito alla tecnica di Cholesky, permette di diminuire il tempo di elaborazione.

5.1.3 Determinazione delle soluzioni $\hat{\mathrm{f}}$ e $\hat{\mathrm{f}}_*$

Una volta ottenuto il vettore $\hat{\boldsymbol{\theta}}_p$ con le stime degli iperparametri ottimi, è sufficiente ricalcolare la matrice autocovarianza KK (sta volta utilizzando l'intero vettore degli ingressi) e la matrice Σ_y che necessita di essere invertita una volta sola per restituire il vettore con i valori di uscita $\hat{\mathbf{f}}(X)$ secondo l'equazione (3.10).

Dopo aver usato i dati del training set per stimare gli iperparametri, costruire il kernel e ricavare l'insieme delle uscite stimate, si passa alla predizione usando gli ingressi del test set e il vettore $\hat{\boldsymbol{\theta}}_p$ ricavato nella fase precedente.

Il programma calcola una matrice U_t con lo stesso metodo utilizzato per U, partendo, però, dagli ingressi del test set X_*. Allo stesso modo di prima vengono ricostruite le matrici K_t e KK_t analoghe a K, KK, partendo da U_t e utilizzando tutti gli nt ingressi di test. Infine si ricava il vettore $\hat{\mathbf{f}}_*(X_*)$ con i valori delle uscite di test attraverso l'equazione (3.10) e la matrice Σ_y^{-1} ricavata precedentemente.

Ora si è giunti alla conclusione e si hanno a disposizione i vettori $\hat{\mathbf{f}}$ e $\hat{\mathbf{f}}_*$ contenenti, rispettivamente, le risposte del sistema simulato per i dati di trainig e per quelli di test.

5.1.4 Conclusione del programma

Il programma costruisce una serie di grafici che mettono a confronto i dati del benchmark con quelli ricavati dalla simulazione e calcola anche i valori degli indici di errore presentati alla fine del capitolo 4.

L'intero processo esposto sopra viene ripetuto per ogni diversa struttura del kernel (per p che va da 1 a lp) in modo da dare la possibilità di confrontare anche visivamente i risultati ottenuti al variare di p. E come risultati finali vengono scelti i vettori con le soluzioni $\hat{\mathbf{f}}$ e $\hat{\mathbf{f}}_*$ associati al kernel che restituisce il minimo valore $J_p(\hat{\boldsymbol{\theta}}_p)$ per la funzione (5.3) secondo l'equazione (3.9)[1].

5.2 Grafici e indici di errore

Di seguito vengono presentati alcuni risultati ottenuti impostando la lunghezza del predittore a $np = 50$ e $lp = 3$. Di conseguenza si ottengono tre strutture diverse per il kernel. Dato che le dimensioni delle matrici K e K_t (che sono le matrici più grandi) sono proporzionali al parametro np, la scelta del suo valore è fortemente limitata dalla capacità di memoria disponibile. In compenso ripetute prove hanno confermato che per tale valore si ottengono risultati soddisfacenti. Il parametro red (fissato pari a 100), invece, influenza molto i tempi di elaborazione, dato che la dimensione della matrice Σ_y è proporzionale al quadrato del

[1] Si faccia attenzione che la struttura ottima del kernel viene scelta unicamente in base ai dati di training. Essa viene compiuta dopo aver calcolato la soluzione per i dati di test $\hat{\mathbf{f}}_*$ esclusivamente per permettere una più comoda comparazione grafica dei diversi risultati.

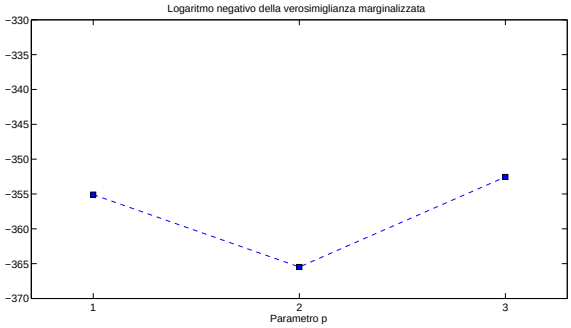

Figura 5.1: Grafico del logaritmo negativo della verosimiglianza marginalizzata in funzione del parametro p.

suo valore ed è richiesta molte volte la sua inversione. L'impostazione di questo parametro, dunque, costituisce un *trad-off* tra velocità di esecuzione e bontà degli iperparametri stimati.

Dopo queste premesse, riportiamo i grafici dei valori $\hat{\mathbf{f}}$ e $\hat{\mathbf{f}}_*$ ottenuti con le tre differenti strutture del kernel.

Si può notare che in tutti e tre i casi la simulazione è buona, ma il valore di $J_p(\hat{\boldsymbol{\theta}}_p)$ nel caso $p = 2$ (Figura 5.1) è minore degli altri, per tale valore, dunque, si ottiene il kernel ottimo che massimizza la verosimiglianza marginale (3.6). Dalle tabelle che presentano gli indici di errore si evince che la soluzione proposta non è quella che minimizza l'errore sui dati di training, contrariamente alla soluzione con $p = 3$ (Figura 5.4). Bisogna, però, ricordare che la verosimiglianza marginale non favorisce semplicemente il modello che si adatta meglio ai dati di training, bensì bilancia questo obiettivo con quello di ottenere una struttura meno complessa. Di conseguenza risulta che il modello ottimo è quello associato a $p = 2$ (Figura 5.3).

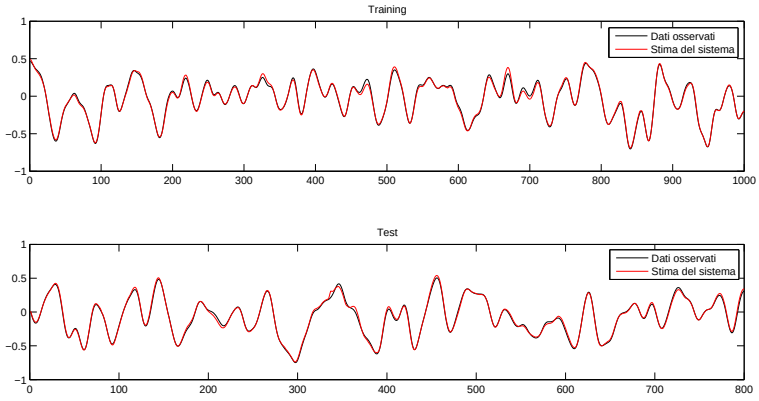

Figura 5.2: Grafico del segnale di uscita per la sequenza di training (sopra) e di test (sotto). In questo caso $p = 1$.

Tabella 5.1: Indici di errore nel caso $p = 1$.

	Training	Test
Valore medio dell'errore	$\mu_e = 1.3985 \cdot 10^{-7}$	$\mu_t = 0.0045$
Deviazione standard dell'errore	$s_e = 0.0203$	$s_t = 0.0200$
Valore quadratico medio dell'errore	$e_{RMSe} = 0.0203$	$e_{RMSt} = 0.0205$

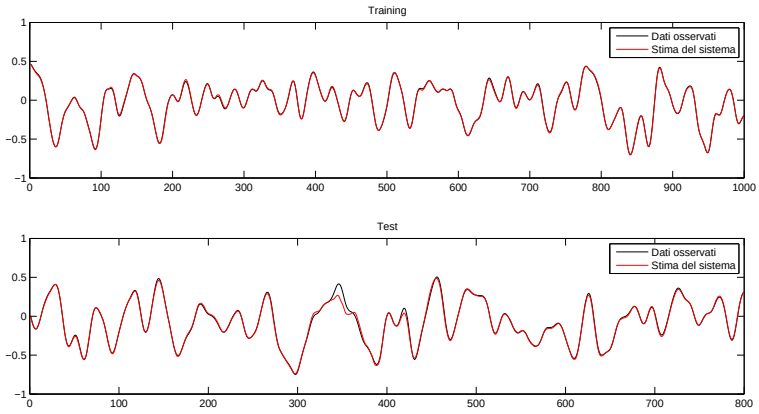

Figura 5.3: Grafico del segnale di uscita per la sequenza di training (sopra) e di test (sotto). In questo caso $p = 2$.

Tabella 5.2: Indici di errore nel caso $p = 2$.

	Training	Test
Valore medio dell'errore	$\mu_e = 1.8879 \cdot 10^{-6}$	$\mu_t = -0.0050$
Deviazione standard dell'error	$s_e = 0.0100$	$s_t = 0.0255$
Valore quadratico medio dell'errore	$e_{RMSe} = 0.0100$	$e_{RMSt} = 0.0260$

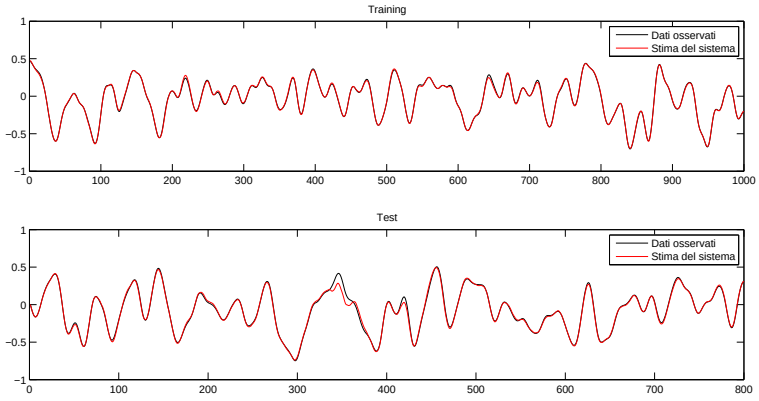

Figura 5.4: Grafico del segnale di uscita per la sequenza di training (sopra) e di test (sotto). In questo caso $p = 3$.

Tabella 5.3: Indici di errore nel caso $p = 3$.

	Training	Test
Valore medio dell'errore	$\mu_e = -1.7128 \cdot 10^{-6}$	$\mu_t = -0.0053$
Deviazione standard dell'errore	$s_e = 0.0085$	$s_t = 0.0245$
Valore quadratico medio dell'errore	$e_{RMSe} = 0.0085$	$e_{RMSt} = 0.0250$

Capitolo 6

Conclusioni

L'algoritmo proposto è caratterizzato da una struttura del kernel che permette di approssimare fedelmente le proprietà dei sitemi fisici non lineari. L'approccio non parametrico consente di ignorare completamente la struttura del sistema da modellare e di basarsi esclusivamente sui dati osservati. La base della teoria Bayesiana su cui poggia permette una implementazione semplice e computazionalmente affrontabile e alcuni accorgimenti, come quello di utilizzare un sottoinsieme ridotto del dataset per la stima degli iperparametri, ne incrementano le prestazioni senza pregiudicare i risultati. L'applicazione del benchmark Wiener-Hammerstein, ha confermato che il comportamento dell'algoritmo risponde con successo a test basati su dati presi dal mondo reale.

Bibliografia

[1] C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

[2] H. Quang, G. Pillonetto, A. Chiuso. Nonlinear System Identification Via Gaussian Regression and Mixtures of Kernels. Proceedings of the 15th IFAC Symposium on System Identification SYSID 2009 Saint-Malo France July 6-8 2009

[3] Poggio, T., and Girosi, F., Networks for Approximation and Learning, PIEEE(78), No. 9, September 1990, pp. 1481-1497.

[4] Schoukens Johan, Suykens Johan, Ljung Lennart, Wiener-Hammerstein Benchmark, 15th IFAC Symposium on System Identification (SYSID 2009), July 6-8, 2009, St. Malo, France.

Ringraziamenti

Ringrazio i miei genitori per la pazienza che ho messo spesso a dura prova durante questi anni all'università e per il sostegno che non mi hanno mai fatto mancare. Ringrazio Monica per la fiducia che ha sempre avuto in me e mi ha saputo trasmettere, specialmente nei momenti più difficili. Ringrazio l'Appartamentissimo per l'affetto dei miei coinquilini. Ringrazio, inoltre, il professor Pillonetto per la sua infinita disponibilità.