POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

Simulazioni RANS e impiego di Machine Learning per il miglioramento di modelli di turbolenza

Relatori

Prof. F. Larocca
Prof. A. Ferrero

Candidato

Marianna Loffredo

Dicembre 2020

Sommario

L'intelligenza artificiale è attualmente oggetto di grande attenzione da parte della comunità scientifica e recentemente numerose ricerche sono state effettuate nell'ambito delle tecniche di apprendimento artificiale, meglio note con il nome di Machine Learning (ML). Tra le diverse applicazioni del ML figura l'analisi dei dati ed in particolare la modellizzazione di relazioni complesse nascoste all'interno dei dati stessi.
In fluidodinamica la modellizzazione e l'analisi di fenomeni turbolenti in presenza di separazioni e transizioni rappresentano ancora una grande sfida. Da una parte, le simulazioni ad alta fedeltà (DNS, LES) hanno un costo computazionale troppo elevato quando applicate a problemi con geometrie complesse o numeri di Reynolds grandi. D'altra parte, l'utilizzo di equazioni più semplici compromette l'affidabilità del risultato. Una possibile alternativa è considerare le equazioni di Navier-Stokes mediate alla Reynolds (RANS), che hanno un basso costo computazionale, e implementarle con modelli di chiusura per ridurre l'approssimazione nel risultato. Tutti questi aspetti sono descritti in modo piu approfondito nell'introduzione.
Lo scopo del presente lavoro di tesi è quello di riuscire a trovare una relazione nascosta all'interno di dati ricavati da simulazioni ad alta fedeltà o sperimentalmente, attraverso il paradigma FIML (Field Inversion and Machine Learning), che applica inizialmente l'approccio di inversione del campo e successivamente tecniche di ML. Nel capitolo 2 si introduce l'approccio di inversione del campo implementato con il metodo dell'aggiunto, utile per migliorare il modello di turbolenza Spalart-Allmaras utilizzato per la chiusura delle equazioni RANS. Il caso in esame è una schiera di pale di turbina a bassa pressione T106c, analizzata nel capitolo 3. Una volta ottenuto il database, vengono applicate tre diverse tecniche di machine learning: programmazione genetica, reti neurali artificiali e random forest, rispettivamente nei capitoli 4, 5 e 6. Infine, negli ultimi due capitoli verranno presentati i risultati ottenuti da un'analisi di fluidodinamica computazionale (CFD) delle tecniche di ML testate (capitolo 7), un confronto tra queste e le conclusioni (capitolo 8).

Ringraziamenti

Risorse di calcolo fornite HPC@POLITO, progetto di Academic Computing del Dipartimento di Automatica e Informatica presso il Politecnico di Torino (http://www.hpc.polito.it)

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Introduzione al Machine Learning

Il Machine Learning è una branca dell'Intelligenza Artificiale (AI). Quest'ultima può essere definita come un insieme di teorie e algoritmi che permettono ai computer di "mimare" l'intelligenza umana.
L'invenzione del termine Intelligenza Artificiale risale al 1995, ad opera del matematico statunitense John McCarthy anche se l'interesse verso questa materia si era manifestato a partire dal 1943 quando i due ricercatori Warren McCulloch e Walter Pitt proposero al mondo scientifico il primo neurone artificiale. Il primo modello di rete neurale risale al 1958: una rete con uno strato di ingresso ed uno di uscita con una regola di apprendimento intermedio basata sull'algoritmo 'error back-propagation', proposto da Frank Rosenblatt (noto psicologo e computer scientist americano) ([7]).
È possibile classificare l'AI in due grandi filoni, alla base poi della distinzione tra Machine Learning e Deep Learning:

- **Intelligenza Artificiale debole (weak AI)**: non raggiunge capacità intellettive tipiche dell'essere umano ma tende a simularne alcune capacità cognitive. Tipiche di questo ambito sono i programmi di problem-solving volti ad allenare la macchina a poter prendere decisioni.
- **Intelligenza Artificiale forte (strong AI)**: si tratta di sistemi capaci di sviluppare in modo autonomo dei processi di pensiero molto simili a quelli umani.

Come già anticipato questa suddivisione è alla base della differenza tra Machine Learning e Deep Learning. Essi si distinguono sulla base dei modelli di apprendimento che utilizzano. In particolare, quando si parla di Machine Learning (ML) si intende un insieme di metodi capaci di apprendere attraverso un sistema di correzione errori e allenamenti per poter svolgere un compito in modo autonomo, senza la necessità di avere un sistema preprogrammato che stabilisca come deve comportarsi e reagire. Differentemente, il Deep Learning (DL) si ispira al funzionamento della mente umana, non è corretto definire dei modelli di apprendimento ma piuttosto un modello capace di emulare il cervello biologico. A tal proposito il DL necessita di reti neurali progettate ad hoc ([7]).
Ci concentriamo ora sulla parte di Machine Learning. La definizione attualmente più accreditata è quella di Tom Michael Mitchell:
"Si dice che un programma apprende dall'esperienza E con riferimento a alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T,

come misurato da P, migliorano con l'esperienza E."
Per meglio comprendere la definizione si introduce una breve descrizione di compiti (T), preformance (P) ed esperienza (E).

Compiti T: i compiti (o meglio Tasks) sono di solito descritti in termini di *come* il sistema di machine learning dovrebbe processare un esempio. Ci sono molteplici tasks risolvibili dal ML, due tra i più importanti:

- classificazione: specificare a quale delle k categorie gli input appartengono;
- regressione: predire un valore numerico dato un certo numero di input.

Performance P: parametro importante per valutare le capacità dell'algoritmo di ML utilizzato. La scelta più comune è quella di utilizzare due gruppi di dati differenti. Il primo gruppo denominato *training dataset* è quello che serve alla macchina per allenarsi e apprendere. Il secondo gruppo, *validation dataset*, viene utilizzato per valutare le performance dell'algoritmo su un insieme di dati che non ha mai visto. In molti casi non è immediato decidere cosa deve essere misurato, molto dipende anche dal task prefissato.

Esperienza E: in base all'esperienza che devono possedere, gli algoritmi di ML sono suddivisi in tre categorie:

- Apprendimento supervisionato: al sistema vengono forniti una serie di coppie input/output e il sistema deve riuscire a trovarne una correlazione in modo tale da essere capace di prevedere un output noti i soli input.
- Apprendimento non supervisionato: il sistema deve essere in grado di gestire e individuare schemi o strutture che leghino una serie di input in ingresso.
- Apprendimento per rinforzo: in questo caso è presente un feedback loop tra il sistema e le sue esperienze attraverso un sistema di ricompense e punizioni. Quando la macchina esegue una scelta corretta viene ricompensata, altrimenti viene corretta fino ad ottenere un'ottimizzazione nella scelta.

Uno degli aspetti cruciali nel Machine Learning è che l'algoritmo sia "generale", cioè che performi bene su problemi non visti, non solo su quelli di training, per i quali è stato allenato. Per tal motivo si definiscono due errori: l'errore di training e quello di test. Il primo misura l'errore dell'algoritmo utilizzando il training dataset (quello di apprendimento), mentre il secondo misura l'errore dell'applicazione del modello a un dataset di test. L'obiettivo è quello di minimizzare l'errore di training e rendere quanto più piccola possibile la differenza tra i due errori. Questa considerazione è alla base dei cosiddetti problemi di *overfitting* e *underfitting*. In riferimento alla figura 1.1, la funzione lineare (sinistra) non riesce a catturare la curvatura presente nei dati e il suo utilizzo risulta inappropriato, si tratta di un classico esempio di underfitting: sebbene sia stata trovata una relazione, essa non riesce a catturare le caratteristiche principali dei dati al suo interno. Il polinomio di nono grado (destra), sebbene passi per tutti i punti ha molti più parametri che punti e quindi è in grado di rappresentare altre infinite funzioni. Risulta poi una funzione poco generalizzabile ad altri problemi. È possibile che sia specifica per il problema esaminato, per il quale la macchina è stata allenata; cambiando caso di studio la funzione potrebbe non rappresentare bene un nuovo set di dati. La soluzione migliore risulta essere la funzione quadratica (centro) che riesce a rappresentare in modo univoco il modello.

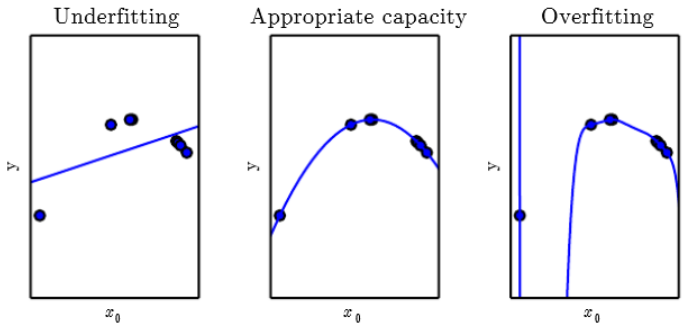


Figura 1.1: funzione lineare che soffre del fenomeno di underfitting, non cattura la curvatura presente nei dati *(Sinistra)*. funzione quadratica che fitta bene i dati *(Centro)*. Polinomio che soffre del fenomeno di overfitting *(Destra)*. ([20])

1.2 Modellazione e simulazione della turbolenza

Nel presente lavoro di tesi verranno analizzati fenomeni turbolenti e di transizione nel campo di moto attorno ad una paletta di turbina a bassa pressione. Nello studio di flussi turbolenti l'obiettivo è quello di ottenere un modello o una teoria utilizzabile per calcolare le quantità di interesse. Tuttavia ci sono molte difficoltà che impediscono la creazione di un modello analitico adeguato: campo di velocità tridimensionale e fortemente dipendente dal tempo, dipendenza del modello dalla geometria del problema, costo computazionale legato ai metodi più affidabili, ecc.
Ad esempio, nel campo delle turbomacchine, l'obiettivo principale è quello di riuscire a migliorare numerosi aspetti: dai materiali, alla struttura fino alla aerotermodinamica, per citarne alcuni. Tra questi, l'aerodinamica è di fondamentale importanza per quanto riguarda l'efficienza, che influirà a sua volta sulla durata, affidabilità e costo dei componenti. Il progetto di questi ultimi è svolto attraverso tecniche di fluidodinamica computazionale (CFD). Sfortunatamente, nel campo CFD ci sono ancora numerose incertezze legate soprattutto all'accuratezza nel catturare le proprietà fisiche del flusso, a maggior ragione quando questo si manifesta in modo turbolento. Infatti, una delle problematiche riscontrate in questo approccio è la natura instazionaria dei flussi nel loro contributo stocastico (dovuto alla presenza di turbolenza) e deterministico (tipico dell'interazione tra componenti), caratterizzati da frequenze diverse. L'interazione di questi contributi è una delle principali cause di irreversibilità e quindi perdite presenti nelle turbomacchine. Dal punto di vista della turbolenza questa interazione può essere trattata attraverso l'esistenza del gap spettrale. Ci sono tre approcci principali per lo studio della turbolenza che illustreremo brevemente: Direct Numerical Simulation (DNS), Large Eddy Simulation (LES) e Raynolds-Average Navier-Stokes (RANS). In figura 1.2 è possibile notare una relazione tra i modelli di turbolenza e la frequenza. Nelle **Direct Numerical Simulation** le equazioni di Navier-Stokes sono usate per determinare il campo di velocità istantanea dalla quale si ricavano tutte le altre informazioni del flusso. Tuttavia, le DNS prevedono una discretizzazione sia in spazio che in tempo ed hanno un costo computazionale elevato. Si stima che la complessità computazionale cresca fortemente con il numero di Reynolds, Re^3, per cui l'applicabilità dell'approccio è limitata a flussi con numeri di Reynolds moderati, impedendo così una generalizzazione del metodo a più flussi e geometrie. Risulta necessario ricorrere a metodi approssimati per valutare gli effetti della turbolenza. Numerosi modelli sono stati sviluppati a tal proposito, classificabili principalmente in cinque categorie:

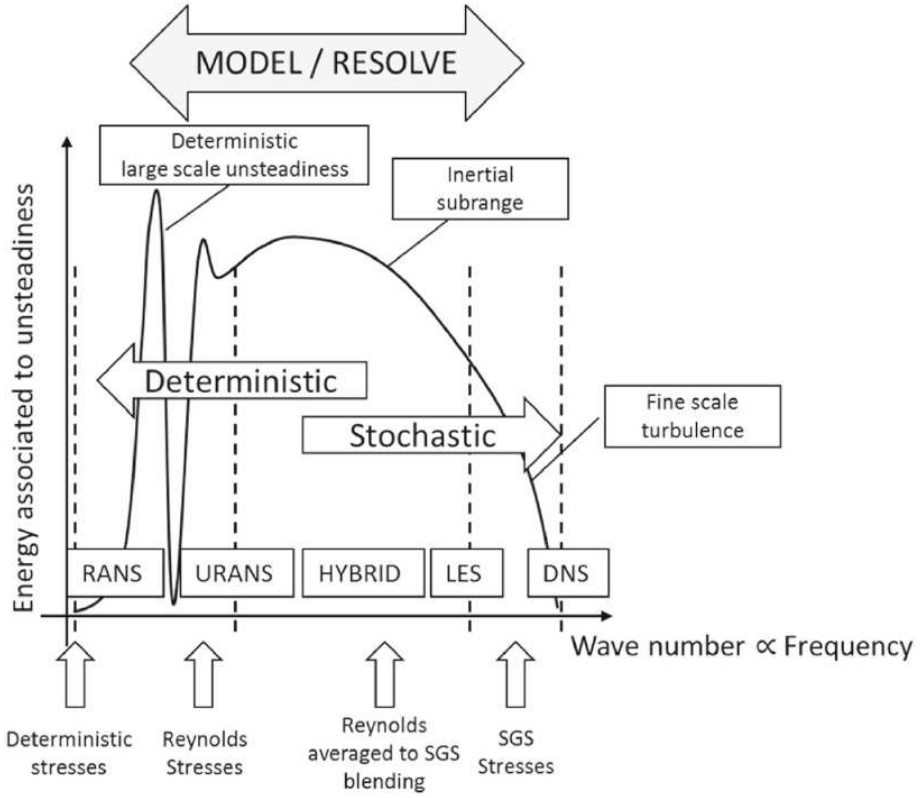


Figura 1.2: Spettro turbolento e modelli di turbolenza in funzione della frequenza ([34]).

- modelli algebrici;
- modelli ad una equazione;
- modelli ad equazioni multiple;
- modelli di chiusura del secondo ordine;
- Large Eddy Simulation (LES).

I primi tre appartengono ai sistemi di chiusura di primo ordine.
Per meglio capirne la suddivisione, si osservi la figura 1.3.

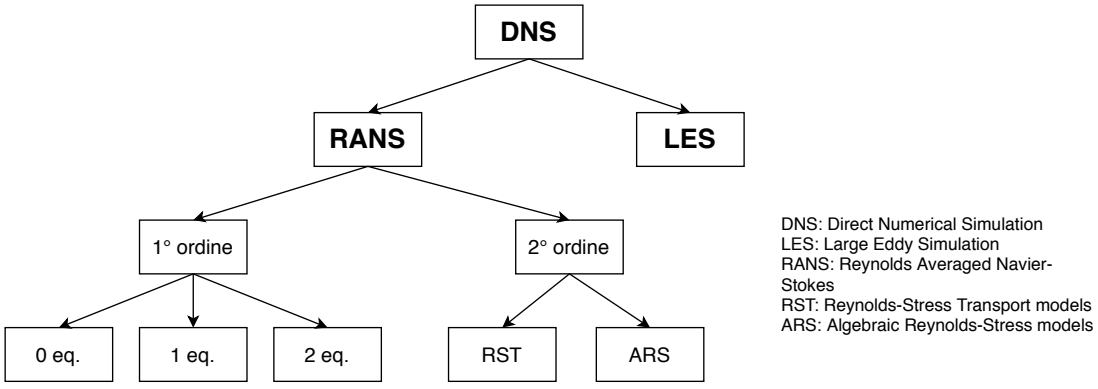


Figura 1.3: Gerarchia modelli turbolenti ([6])

Per ridurre il costo computazione è possibile utilizzare l'approccio della **Large Eddy Simulation**: gli effetti di larga scala del moto turbolento tridimensionale sono considerati preponderanti e rappresentati in maniera diretta, mentre quelli di piccola scala sono modellati. Con questa suddivisione il costo computazione si riduce rispetto alle DNS ma rimane comunque tendenzialmente alto. Il terzo, e più usato, approccio è quello basato sulle equazioni di Navier-Stokes mediate alla Reynolds (**RANS**). Come suggerito dal nome queste equazioni sono risolte per un campo di velocità medio. All'interno delle equazioni, però, compaiono due nuovi termini: gli sforzi turbolenti e il flusso di calore turbolento, il sistema quindi ha bisogno di modelli di chiusura e come visto in figura 1.3 esistono modelli a zero-, uno- o due equazioni. Tra i più utilizzati ricordiamo:

- modello $k - \varepsilon$: modello a due equazioni. L'equazione di trasporto è risolta per le due quantità di turbolenza da cui il modello prende il nome: energia cinetica turbolenta k e dissipazione ε. È il modello più utilizzato per la sua semplicità e perché è incorporato in molti codici CFD commerciali.
- modello $k - \omega$: modello a due equazioni. La seconda equazione di trasporto è scritta per la frequenza caratteristica dei vortici ω. Modello sviluppato per evitare il problema della singolarità a parete del modello $k - \varepsilon$.
- modello di Spalart-Allmaras (SA): modello ad una equazione. Equazione di trasporto risolta per la viscosità turbolenta cinematica. Sviluppato inizialmente per applicazioni aerodinamiche, ma molto utilizzato per la sua semplicità.
- Reynolds-Stress model (RSM): le equazioni di trasporto sono risolte per gli sforzi di Reynolds e per la dissipazione. Applicabile a qualunque flusso turbolento.

Con l'utilizzo delle equazioni RANS il costo computazionale si riduce di molto e questo lo rende il modello più utilizzabile ma d'altra parte la soluzione non ha un buon grado di accuratezza e affidabilità, dovuto alle numerose approssimazioni introdotte. L'obiettivo ottimale potrebbe essere quello di utilizzare i modelli RANS per avere un costo computazionale basso ma migliorarne le prestazioni per ottenere risultati più affidabili. Negli anni si è pensato di riuscire a migliorare i modelli di turbolenza mentre ora ci si concentra più su un'altra strada: utilizzare i dati disponibili dalle simulazioni numeriche dirette e attraverso algoritmi di apprendimento automatico ottenere risultati migliori e modelli generalizzabili.
Il concetto fondamentale è che attraverso la tecnica di Machine Learning, piuttosto che concentrarsi sulla chiusura delle RANS si potrebbe trovare una relazione tra le incognite aggiuntive e le altre variabili fisiche note. Un esempio è mostrato in figura 1.4. Per preparare il dataset di training, le caratteristiche del flusso sono calcolate dalle simulazioni RANS mentre gli sforzi di Raynolds sono presi dalle simulazioni ad alta fedeltà DNS/LES. La tecnica di ML è utilizzata per stabilire una relazione funzionale tra le caratteristiche (input) e gli sforzi (output). Gli stress di Reynolds ottenuti vengono poi sostituiti nelle RANS per ottenere la soluzione finale.
Per ulteriori approfondimenti, si consulti il lavoro di Sandberg e Michelassi ([34]) che propone una revisione sullo stato dell'arte di simulazioni ad alta fedeltà. In questa direzione gli studi negli ultimi anni sono stati molteplici. Per citarne alcuni: Wang et al. ([38]) propongono un approccio data-driven utilizzando la tecnica di Random Forest per correggere errori nelle RANS utilizzando un database ottenuto da simulazioni DNS e sulla base di questo cercano di predire errori in altri flussi dove non sono disponibili dati. Mohebujjaman et al. ([29]) utilizzano un approccio simile per migliorare modelli di ordine ridotto. Sulla stessa scia, Yang et al. ([39]) utilizzano due metodi di machine

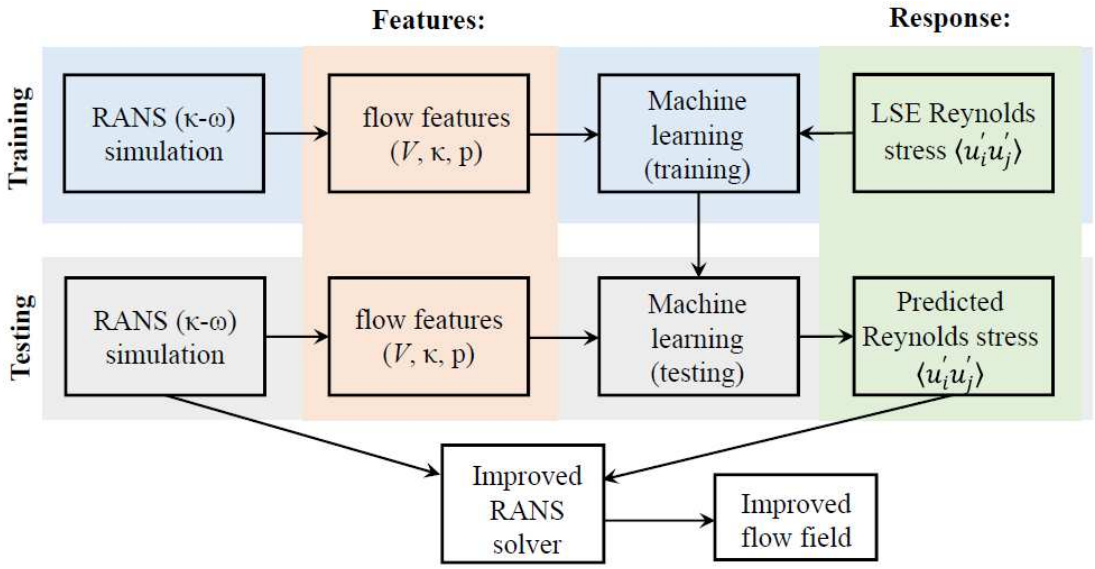


Figura 1.4: Machine Learning e simulazioni RANS [27]

learning, Random Forest e Artificial Neural Network (ANN), per migliorare il modello di transizione a quattro equazioni ($k - \omega - \gamma - A_r$).
Importanti sono i lavori di Singh et al. ([35]) e di Parish, Duraisamy ([32]): in entrambi si utilizza l'approccio di Machine Learning e quello di inversione del campo; nel primo si dimostra l'abilità del metodo di inversione del campo per predire informazioni utilizzando un numero di dati limitati, mentre nel secondo viene proposto un modello chiamato Field Inversion and Machine Learning (FIML), che sfrutta i dati provenienti da simulazioni dirette per agevolare la creazione di modelli di chiusura.

Capitolo 2

Metodo di inversione del campo

Come già anticipato, il problema di trovare un modello di turbolenza scegliendo un compromesso tra facilità dello stesso, costo computazionale e affidabilità dei dati non è banale. Recentemente, molte tecniche di machine learning hanno cercato di migliorare le previsioni del sistema di equazioni di Navier-Stokes mediate alla Reynolds (RANS). In questo capitolo ci occuperemo di una precisa tecnica introdotta per la prima volta da Parish e Duraisamy ([32]): inversione del campo e machine learning (originalmente *Field Inversion and Machine Learning*, FIML). Questo approccio verrà poi implementato con il metodo dell'aggiunto per ridurre il costo computazionale.
Andando per gradi, inizialmente verrà descritto il sistema di equazioni RANS e il modello di turbolenza Spalart-Allmaras (SA) e successivamente si indagherà più approfonditamente sull'approccio di inversione del campo.

2.1 Modello fisico - RANS

Per ricavare il sistema di equazioni RANS conviene partire dal set di equazioni di Navier-Stokes:

- Equazione di conservazione della massa

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v_i)}{\partial x_i} = 0 \tag{2.1}$$

- Equazione di conservazione della quantità di moto

$$\frac{\partial (\rho v_i)}{\partial t} + \frac{\partial (\rho v_i v_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.2}$$

- Equazione di conservazione dell'energia

$$\frac{\partial (\rho E)}{\partial t} + \frac{\partial (\rho v_j H)}{\partial x_j} = \frac{\partial (v_i \tau_{ij})}{\partial x_j} + \frac{\partial}{\partial x_j}\left(k \frac{\partial T}{\partial x_j}\right) \tag{2.3}$$

Dove $\rho, p, E, H, T, k, t, v_{i,j}, x_{i,j}, \tau_{ij}$ sono densità, pressione, energia totale, entalpia totale, temperatura, conducibilità termica, tempo, generica componente di velocità e coordinate, tensore degli sforzi viscosi, rispettivamente.
Assumendo l'ipotesi di Stokes, si può scrivere il tensore degli sforzi viscosi in forma estesa:

$$\tau_{ij} = 2\mu \left(S_{ij} - \frac{1}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right)$$

con S_{ij} tensore della velocità di deformazione così definito:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

Per ottenere le equazioni mediate alla Reynolds bisogna scomporre ogni variabile di flusso in due componenti: un termine medio e uno fluttuante, per poi risolvere le equazioni per i valori medi. Il termine medio può essere calcolato come media temporale (in questo caso al termine medio verrà sovrapposta una barretta ' $\bar{}$ ' e al termine fluttuante un singolo apice) oppure con il metodo di Favre considerando densità del flusso comprimibile variabile (termine medio con tilde '~' e termine fluttuante con doppio apice). Ad esempio:

$$\rho = \bar{\rho} + \rho' \text{: Reynolds} \qquad\qquad v = \tilde{v} + v'' \text{: Favre}$$

Con queste considerazioni si ottengono le equazioni di Navier-Stokes mediate alla Reynolds (RANS):

$$\begin{cases} \dfrac{\partial \bar{\rho}}{\partial t} + \dfrac{\partial(\bar{\rho}\tilde{v}_i)}{\partial x_i} = 0 \\ \dfrac{\partial(\bar{\rho}\tilde{v}_i)}{\partial t} + \dfrac{\partial(\bar{\rho}\tilde{v}_i\tilde{v}_j)}{\partial x_j} = -\dfrac{\partial \bar{p}}{\partial x_i} + \dfrac{\partial \tilde{\tau}_{ij}}{\partial x_j} \\ \dfrac{\partial \tilde{E}}{\partial t} + \dfrac{\partial(\tilde{v}_j\tilde{H})}{\partial x_j} = \dfrac{\partial}{\partial x_j}(\tilde{v}_j\tilde{\tau}_{ij} - q_i) \end{cases} \tag{2.4}$$

Dove il flusso di calore è espresso in termini di viscosità dinamica e numero di Prandtl classici e turbolenti (pedice t per indicare le grandezze turbolente):

$$q_i = -\frac{\gamma}{\gamma - 1}\left(\frac{\mu}{\mathrm{Pr}} + \frac{\mu_t}{\mathrm{Pr}_t}\right)\frac{\partial T}{\partial x_i}$$

e il tensore degli sforzi laminari e turbolenti:

$$\tilde{\tau}_{ij} = (2\mu + 2\mu_t)\left(S_{ij} - \frac{1}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right)$$

Per approfondimenti e una più rigorosa derivazione delle equazioni si può consultare il libro *Computational fluid dynamics: principles and applications*, Blazek ([6]).

2.2 Modello di turbolenza

I modelli di chiusura del primo ordine rappresentano il modo più semplice per approssimare gli sforzi di Reynolds. Il modello di Spalart-Allmaras (SA) descrive un modello a singola equazione di trasporto nell'incognita $\tilde{\nu}$ (viscosità cinematica turbolenta modificata) inizialmente sviluppato per applicazioni aerodinamiche ([36]). Il modello SA è uno dei più utilizzato per la chiusura delle equazioni RANS poiché semplice e accurato per flussi completamente turbolenti ad alti numeri di Reynolds. Il modello originale presenta 4 versioni: dalla più semplice per il flusso di taglio libero, alla forma più complessa applicabile a flussi viscosi attorno a un corpo tridimensionale. In ogni versione vengono aggiunti termini o fattori per considerare gli effetti fisici in questione. Per una descrizione più dettagliata si rimanda al lavoro originale proposto da Spalart-Allmaras *"A one-equation turbulence model for aerodynamic flows"* ([36]).

Nel modello originale e nella forma più generale, si valutano gli sforzi di Reynolds con l'ipotesi di Boussinesque che prevedere un legame tra la viscosità cinematica turbolenta (ν_t) e la stessa quantità modificata:

$$\nu_t = \tilde{\nu} f_{v_1}$$

dove:

$$f_{v_1} = \frac{\chi^3}{\chi^3 + c_{v_1}^3} \qquad \chi = \frac{\tilde{\nu}}{\nu}$$

La variabile χ è definita come rapporto tra la viscosità turbolenta modificata e la classica viscosità cinematica e prende il nome di *intensità della turbolenza*. L'equazione di trasporto in $\tilde{\nu}$ valida sia per flussi incompressibili che compressibili è:

$$\frac{D\tilde{\nu}}{Dt} = P - D + T + \frac{1}{\sigma}\left[\nabla \cdot \left(\left(\nu + \tilde{\nu}\right)\nabla\tilde{\nu}\right) + c_{b_2}\left(\nabla\tilde{\nu}\right)^2\right] \tag{2.5}$$

I termini P, D, T sono rispettivamente i termini di Produzione, Distruzione e Trasporto, così definiti:

$$P = c_{b_1}\left(1 - f_{t_2}\right)\tilde{S}\tilde{\nu} \qquad D = \left(c_{w_1} f_w - \frac{c_{b_1}}{k^2} f_{t_2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 \qquad T = f_{t_1}\left(\Delta\mathbf{u}\right)^2$$

dove

$$\tilde{S} = S + \underbrace{\frac{\tilde{\nu}}{k^2 d^2}}_{\bar{S}} f_{v_2} \qquad f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}}$$

$$f_w = g\left[\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right]^{1/6} \qquad g = r + c_{w_2}\left(r^6 - r\right) \qquad r = \min\left(\frac{\bar{S}}{\tilde{S}}, r_{lim}\right)$$

con S intensità di vorticità e d distanza dalla parete più vicina.
Il termine f_{t_1} è introdotto nel modello quando si inserisce un punto di transizione da laminare a turbolento e compare infatti nel termine di trasporto T. Inserito il punto di transizione un altro termine si attiva: f_{t_2}, per ritardare la transizione. Essendo il punto di transizione non noto a priori, spesso si preferisce non utilizzarlo, per cui il termine f_{t_1} risulta nullo e f_{t_2} trascurabile. Maggiori informazioni sulla loro definizione e utilizzo possono essere trovate in [36], [16].
Si introduce una breve descrizione dei principali termini che compaiono. Nella versione più semplice del modello (flusso di taglio libero) è presente solo il termine di produzione, privato del contributo di trasporto f_{t_2}. Per tal motivo il pedice "b" presente sia nelle funzioni adimensionali f_i che nelle costanti c_i sta per *"basic"*. Considerando un flusso vicino a parete si introduce il termine distruttivo costituito da funzione adimensionale e costante con il pedice "w", con riferimento alla parete *"wall"*. Per alti numeri di Reynolds anche il termine viscoso viene introdotto: f_{v_i}, c_{v_i} dove il pedice "v" sta appunto per *"viscoso"*. Infine, nel termine di trasporto il pedice dominante è "t" che però non verrà utilizzato nel presente lavoro per motivi già spiegati.
Tuttavia, uno dei problemi di questo modello è la possibilità che la vorticità modificata $\tilde{S}$ diventi negativa. Generalmente si ha un limite inferiore a tale valore $\bar{S} < 0.3S$, ma esso può annullarsi o diventare negativo a causa del termine f_{v_2} che può essere negativo in alcuni range di valori di χ. Per tal motivo è stata proposta da Allmaras et al. ([2])

una modifica al modello originale nella definizione della vorticità modificata: essa rimane invariata nel caso in cui $\tilde{S} > 0.3S$ e cambia altrove, in questo modo:

$$\tilde{S} = \begin{cases} S + \bar{S}, & \text{se } \bar{S} \geq -c_{v_2} S \\ S + \dfrac{s\left(c_{v_2}^2 S + c_{v_3} \bar{S}\right)}{\left(c_{v_3} - 2c_{v_2}\right) S - \bar{S}}, & \text{se } \bar{S} < -c_{v_2} S \end{cases}$$

È possibile scrivere il modello in forma differenziale conservativa per un flusso compressibile, combinando il modello di turbolenza SA con l'equazione di conservazione della massa. In generale vale:

$$\rho \cdot \{S - A\} + \tilde{\nu} \cdot \{mass\} = 0 \tag{2.6}$$

Sviluppando l'equazione 2.6, eliminando i termini di trasporto f_{t_1}, f_{t_2}, si ottiene in definitiva:

$$\frac{\partial(\rho\tilde{\nu})}{\partial t} + \nabla\cdot(\rho\mathbf{u}\tilde{\nu}) = \rho(P-D) + \frac{1}{\sigma}\nabla\cdot\left(\rho\left(\nu+\tilde{\nu}\right)\nabla\tilde{\nu}\right) + \frac{c_{b_2}}{\sigma}\rho\left(\nabla\tilde{\nu}\right)^2 - \frac{1}{\sigma}\left(\nu+\tilde{\nu}\right)\nabla\rho\cdot\nabla\tilde{\nu} \tag{2.7}$$

con

$$P = c_{b_1}\tilde{S}\tilde{\nu} \qquad\qquad D = c_{w_1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2$$

e gli altri termini già definiti. L'equazione 2.7 è la versione definita del modello di turbolenza che verrà utilizzata nell'approccio di inversione del campo.
Le costanti definite hanno valore:
$c_{b_1} = 0.1355$, $c_{b_2} = 0.622$, $\sigma = 2/3$, $k = 0.41$, $c_{w_1} = 0.806$, $c_{w_2} = 0.3$, $c_{w_3} = 2$, $c_{v_1} = 7.1$, $r_{lim} = 10$.

2.3 Inversione del campo e metodo dell'aggiunto

Il paradigma dell'inversione del campo è stato proposto per la prima volta da Duraisamy et al. ([32]), dal titolo originale *Field Inversion and Machine Learning*, che permette di migliorare le abilità di predizione dei modelli RANS in due step: inizialmente si applica l'approccio di inversione del campo che permette di ricavare un fattore correttivo del modello di turbolenza e successivamente tale fattore viene analizzato con varie tecniche di machine learning al fine di generalizzare i risultati e poter fare delle previsioni che siano in accordo con i dati sperimentali disponibili. L'inversione del campo, nel nostro caso, verrà applicato al modello di turbolenza prescelto: quello di Spalart-Allmaras. Nell'equazione di trasposto del modello viene inserito un campo correttivo $\beta(x)$ in un punto significativo. Questo campo viene calcolato attraverso un problema di ottimizzazione multidimensionale: si definisce una funzione goal (o obiettivo) allo scopo di misurare le discrepanze tra il modello originale e quello riprodotto. La minimizzazione di questa funzione è eseguita attraverso il metodo del gradiente discendente, e successivamente si vedrà anche l'introduzione del metodo dell'aggiunto per ridurre il costo computazionale. Una volta trovato il campo correttivo, si passa alla parte di machine learning per cercare di applicare quanto trovato a flussi differenti. Un approccio generico è visualizzabile in figura 2.1.

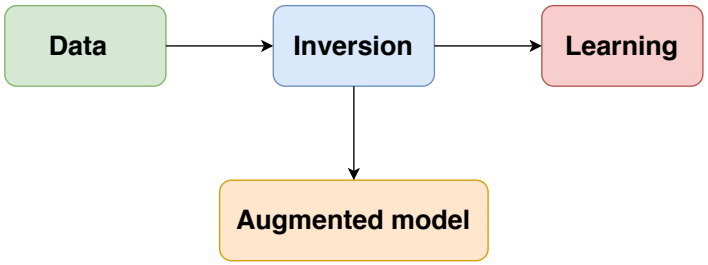


Figura 2.1: Diagramma di flusso FIML [22]

Il riquadro "data" rappresenta la selezione di dati ad alta fedeltà o provenienti da risultati sperimentali scelti come base per migliorare il modello. Il processo FIML utilizza questi dati per produrre un nuovo modello che riesce a fittarli bene. Nel riquadro "inversion" si determina un campo di correzione ottimale al fine di minimizzare la funzione goal (o obiettivo). La scelta della funzione obiettivo è di grande importanza perché deve misurare la discrepanza tra i due modelli tenendo in considerazione che la perfetta uguaglianza è irraggiungibile, ma l'errore deve essere contenuto. Infine, il riquadro "learning" rappresenta il processo di estrazione di variabili dalla soluzione del processo inverso e si testa un algoritmo di machine learning che produce una relazione che lega il campo di correzione a delle variabili fisiche note.
Il campo $\beta(x)$ è inserito in un termine correttivo $h(\beta(x))$ che è applicato all'equazione di trasporto definita dal modello di turbolenza SA, e in particolare moltiplica il termine di produzione:

$$\frac{\partial(\rho\tilde{\nu})}{\partial t}+\nabla\cdot(\rho\mathbf{u}\tilde{\nu}) = \rho(h(\beta(x))P-D)+\frac{1}{\sigma}\nabla\cdot(\rho\left(\nu+\tilde{\nu}\right)\nabla\tilde{\nu})+\frac{c_{b_2}}{\sigma}\rho\left(\nabla\tilde{\nu}\right)^2-\frac{1}{\sigma}\left(\nu+\tilde{\nu}\right)\nabla\rho\cdot\nabla\tilde{\nu} \tag{2.8}$$

Il campo di correzione $h(\beta(x))$ è stato assunto variabile nell'intervallo [0,1], in modo tale da poter agire come funzione di intermittenza.
Come già anticipato il problema di ottimizzazione è risolto attraverso il metodo del gradiente discendente. L'obiettivo dello step di ottimizzazione è quello di minimizzare la differenza tra i valori di riferimento e quelli ottenuti dal modello. In questo caso, come già accennato, si utilizza il metodo del gradiente discendente che consiste nel calcolare la direzione di massima discesa in un punto corrispondente all'opposto del gradiente della funzione in quel punto. In questo modo si riesce a ottenere una direzione che prosegue nella ricerca di un minimo con più processi iterativi. Partendo una soluzione iniziale β questa viene aggiornata ad ogni iterazione:

$$\beta = \beta - \delta\frac{dG}{d\beta} \tag{2.9}$$

dove δ è il passo di iterazione per ogni step temporale e G è la funzione goal. Tuttavia, la grandezza del problema di ottimizzazione è legata al numero di gradi di libertà per equazione e in questo caso risulta essere molto grande, bisognerebbe risolvere il sistema per ogni cella della griglia computazionale: per evitare questo enorme costo computazionale si ricorre al **metodo dell'aggiunto**. Si introduce un set di equazioni aggiuntive molto più semplici (a livello computazionale) delle precedenti, il gradiente della funzione goal viene calcolato quindi con questo metodo:

$$\frac{\mathrm{d}G}{\mathrm{d}\beta} = \frac{\partial G}{\partial\beta}+\Psi\frac{\partial R}{\partial\beta} \tag{2.10}$$

e il sistema di euqazioni aggiunto risulta essere di conseguenza:

$$\left[\frac{\partial R}{\partial U}\right]^T \Psi = -\left[\frac{\partial G}{\partial U}\right]^T \tag{2.11}$$

Dove R è il residuo dal set di equazioni di governo, Ψ è la variabile aggiunta e U è il set di variabili fluidodinamiche.
Una volta risolto il sistema di equazioni aggiunto e trovato il valore del gradiente si può procedere con l'ottimizzazione. L'obiettivo di quest'ultima è la minimizzazione delle differenze tra i due modelli: quello di turbolenza modificato e quello di riferimento. Il fattore di correzione viene aggiornato ad ogni step temporale fino ad ottenere la configurazione finale, che sarà alla base dell'addestramento degli algoritmi di machine learning che si vedranno nei prossimi capitoli.
La procedura può essere riassunta in alcuni passi:

- si cerca una soluzione stazionaria del modello di turbolenza SA (residui delle equazioni di governo inferiori alla soglia 10^{-6})
- si calcola il gradiente $\frac{\mathrm{d}G}{\mathrm{d}\beta}$ con il metodo dell'aggiunto
- il valore di β viene aggiornato
- si genera un transitorio nel tempo che si conclude una volta trovata una nuova soluzione stazionaria.

Nel capitolo successivo queste nozioni verranno specificate per il caso studio in esame e verrà infine introdotto il database ottenuto con questo approccio.

Capitolo 3

Caso di studio

3.1 Schiera T106c

Nei moderni motori turbofan per aerei una importante analisi è quella della turbina di bassa pressione, utilizzata per produrre potenza e muovere il fan. Quest'ultimo è di grande dimensione, ha un diametro notevole e quindi necessita di una grande potenza senza però compromettere l'efficienza totale. Inoltre, il trend attuale è quello di ridurre il numero di pale nelle turbomacchine per abbassare i costi di manutenzione e ottenere anche delle macchine più leggere. Queste considerazioni portano allo sviluppo di turbomacchine, in tal caso di turbine, caratterizzate da un alto carico aerodinamico che si manifesta in profili molto portanti. Tali profili, però, essendo caratterizzati da un grande campo di velocità sul dorso possono incorrere nel problema di separazione (specialmente a bassi numeri di Reynolds) del flusso nello strato limite a causa del forte gradiente di pressione avverso. Notoriamente fenomeni di separazione e di transizione provocano numerose perdite che si preferisce evitare. Nel lavoro di Curtis et al. ([11]) si evince che la causa primaria di perdite è nello strato limite sul dorso della paletta (suction side), specialmente in presenza di una bolla di separazione, per tal motivo analizzare soprattutto il dorso della paletta è di estrema importanza. Negli attuali motori turbofan, il range di variazione del numero di Reynolds in una turbina a bassa pressione è $10^5 - 4 \cdot 10^5$ dove i valori più bassi si manifestano in crociera e quelli più alti al decollo. A causa del basso numero di Reynolds in condizioni di crociera una parte dello strato limite sul dorso della pala della turbina di bassa pressione può subire una transizione da laminare a turbolento. Inoltre, se la conseguente diffusione è grande può manifestarsi anche un fenomeno di separazione, eventualmente seguito da un flusso riattaccato. I fenomeni di transizione e separazione in tal caso si influenzano a vicenda e si comprende quanto sia importante conoscere a fondo questi fenomeni e riuscire a prevederli al fine di controllarli e ridurre le perdite, per lo sviluppo di motori sempre più efficienti.
In figura 3.1 è rappresentato un inviluppo di volo di un tipico aereo commerciale, con riferimento al numero di Reynolds raggiunto dalla turbina di bassa pressione. Le LPTs (Low Pressure Turbine) generalmente lavorano in un range di numero di Mach in uscita $0.6 < M < 0.8$, considerando una quota di crociera di circa $35000 - 40000$ piedi, dal grafico si nota che si raggiungono valori di numeri di Reynolds inferiori a 200000, negli aerei moderni spesso si arriva anche valori inferiori di 100000. In fase di decollo, invece, si riscontra un valore di numero di Reynolds ben più alto, questo a conferma del fatto che la LPT subisce una grande variazione nel range di Reynolds che può causare non pochi problemi di separazione e transizione. Tali fenomeni influenzano la tipologia di flusso. Il caso in esame, quindi, tratta proprio una schiera con profili ad alta portanza di una

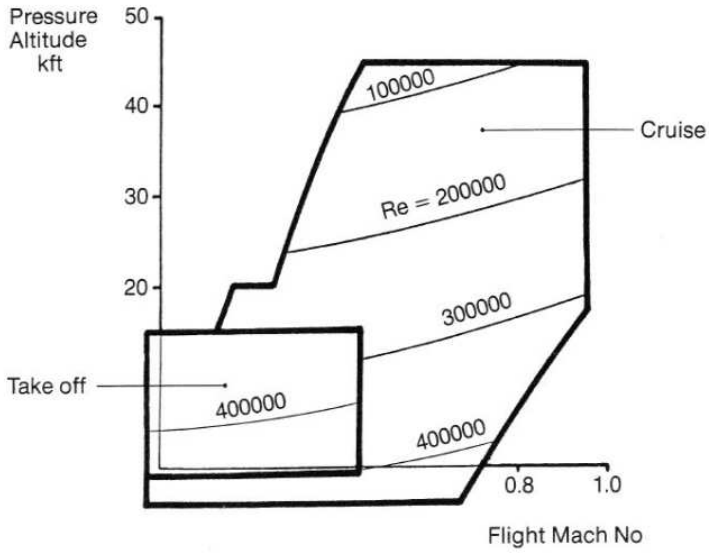


Figura 3.1: Inviluppo di volo di un aereo commerciale con i numeri di Reynolds tipici della turbina di bassa pressione ([23]).

turbina a bassa pressione T106c a due diversi numeri di Reynolds: $8 \cdot 10^4$ e $2.5 \cdot 10^5$. Ulteriori informazioni su performance aerodinamiche e risultati sperimentali su questa schiera si possono trovare nell'articolo di Michálek et al. ([28]).
Riprendendo l'approccio di Ferrero et al. ([16], [17]), le equazioni RANS sono state integrate con il metodo delle linee utilizzando una discretizzazione spaziale di Galerkin del secondo ordine e un'integrazione nel tempo con il metodo implicito di Eulero linearizzato. Il dominio spaziale è stato creato attraverso il tool Gmsh e il dominio computazionale è stato suddiviso con una mesh strutturata nello strato limite e una non strutturata nel resto del dominio, come mostrato in figura 3.2.

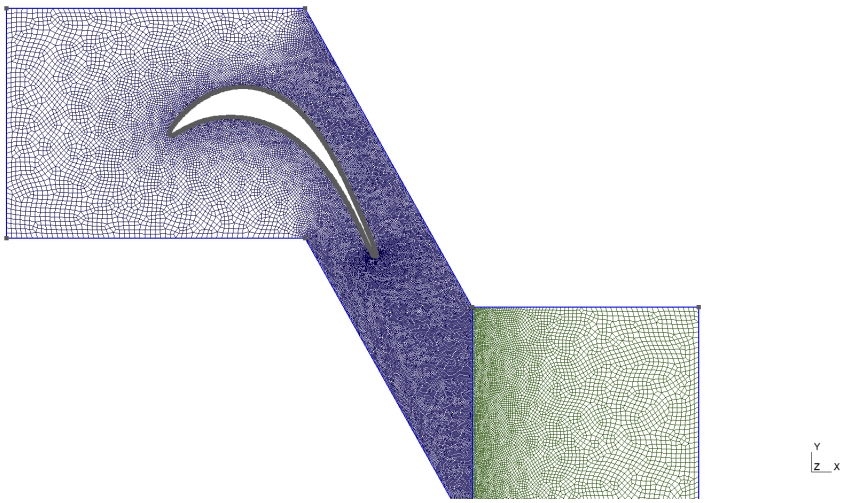


Figura 3.2: Mesh 2D paletta T106c

Questa schiera è stata studiata sperimentalmente al VKI e sono disponibili numerosi risultati per diversi numeri di Reynolds, di nostro interesse distribuzione di mach isoentropico, perdite cinetiche e angolo di uscita nella scia. Analisi sperimentali sono disponibili in ([28], [3]).
Il campo di moto in questione è studiato con i seguenti parametri:

- Mach isoentropico di uscita: $M_{2_s} = 0.65$
- Angolo di incidenza: $\alpha = 32.7°$
- Numero di Reynolds isoentropico di uscita: $8 \cdot 10^4$ e $2.5 \cdot 10^5$.

Il numero di Reynolds Re_{2s} è stato calcolato considerando la corda del profilo, la velocità isoentropica di uscita e la relativa densità.
Inoltre, si considera un'intensità turbolenta costante e pari 0.9%, e di conseguenza il rapporto $\frac{\tilde{\nu}}{\nu} = 0.1$; un passo temporale costante $\delta = 0.1$. Come fluido si considera l'aria, e perciò si assumono $\gamma = 1.4, Pr = 0.72, Pr_t = 0.9$.
Nel prossimo paragrafo si introduce la generazione del database con il metodo di inversione del campo, che sarà alla base delle successive analisi con le tecniche di machine learning.

3.2 Generazione del database

In questo lavoro l'approccio di inversione del campo è applicato alla cascata di turbina a bassa pressione T106c a due diversi numeri di Reynolds (8000, 250000). Come già descritto il metodo del gradiente e quello dell'aggiunto verranno utilizzati per la minimizzazione dell'errore sulla funzione obiettivo. Quest'ultima è stata definita da Ferrero et al ([17]) come:

$$G = \int_w (M_s - M_s^{exp})^2 \, \mathrm{d}l + \lambda \int_\Omega (\beta - 1)^2 \, \mathrm{d}\Omega \tag{3.1}$$

Il primo è un integrale di linea sulla parete della paletta e serve a misurare l'errore in norma 2 della distribuzione del Mach isoentropico. Il secondo, invece, è un integrale di superficie sul dominio computazionale Ω, è utile per evitare delle correzioni non necessarie durante il processo di ottimizzazione. Si è scelto di utilizzare il numero di Mach isoentropico come funzione goal, definito nel seguente modo:

$$M_s = \sqrt{\frac{2}{\gamma - 1}\left[\left(\frac{p_i^\circ}{p_w}\right)^{\frac{\gamma-1}{\gamma}} - 1\right]}$$

dove p_w è la pressione statica a parete e p_i° è la pressione totale di ingresso. Questa scelta è legata al fatto che sono disponibili dei dati sperimentali su questa grandezza, e che graficamente si nota un plateau nella distribuzione quando è presente una separazione sul dorso.
Il database per i due diversi numeri di Reynolds è ottenuto come descritto nel paragrafo 2.3 e il campo di correzione $h(\beta(x))$ è stato costruito. Per la scelta del fattore di correzione si segue la strada di Ferrero et al. ([17]) che hanno dimostrato essere la più robusta tra varie alternative:

$$h(\beta) = \begin{cases} 0 & \beta \leq 0 \\ 3\beta^2 - 2\beta^3 & 0 < \beta < 1 \\ 1 & \beta \geq 1 \end{cases}$$

Esso agisce come fattore di intermittenza, nel range [0,1] al fine di simulare i fenomeni transitori. In forma sintetica si ottiene un database costituito dai seguenti dati di input:

Input	Descrizione
$\bar{S}$	vorticità adimensionata
$\chi = \frac{\tilde{\nu}}{\nu}$	rapporto tra viscosità turbolenta e cinematica
$\frac{S}{\omega + \varepsilon}$	rapporto tra modulo dei tensori S e vorticità ω
f_d	shedding function
$\frac{P}{D + \varepsilon}$	rapporto tra termine di produzione e di distruzione
$\frac{\nabla p \cdot \mathbf{u}}{p\lvert\mathbf{u}\rvert} d$	gradiente di pressione adimensionale
M_c	Numero di Mach convettivo

Tabella 3.1: Input nel database

La funzione f_d è stata usata da Singh et al. ([35]), originariamente proposta da Spalart ([37]) e poi modificata nella versione $f_d^{'}$ da Ferrero et al. ([17]).

$$f_d = 1 - \tanh\left((8r_d)^3\right) \longrightarrow f_d^{'} = 1 - \tanh\left((8r_d)^{0.5}\right)$$

con r_d grandezza adimensionale che combina il gradiente di velocità, la viscosità molecolare e turbolenza e la distanza da parete:

$$r_d = \frac{\nu + \tilde{\nu}}{d^2 k^2 \sqrt{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}}}$$

dove $k = 0.41$ è la costante di von Karman.
Notiamo che in alcuni rapporti è stato inserito il termine $\varepsilon = 10^{-10}$ per evitare di dividere per zero. Per la definizione del Mach convettivo si segue la proposta di Paciorri e Sabetta ([31]) che lo definiscono nel modo seguente:

$$M_c = \frac{|u_1 - u_2|}{a_1 + a_2}$$

Dove u_1, u_2 sono le velocità agli estremi dello strato di taglio e a_1, a_2 sono le relative velocità del suono. Esso riesce ad identificare le zone in cui è presente uno scorrimento, è un sensore che sente la variazione di velocità tangenziale, se presente. Risulta quindi molto adatto nella definizione della zona dello strato limite e della scia, in cui si intende allenare le tecniche di machine learning per prevedere il fattore di correzione.
Negli algoritmi di apprendimento che verranno analizzati, la parte importante è riuscire a prevedere il campo di moto attorno alla paletta e le eventuali separazioni. Tuttavia, nel database originale il 98% dei dati corrisponde a un fattore di correzione $\beta > 0.9$. Utilizzando un errore quadratico medio si rischia di riuscire a prevedere bene tutte le zone in cui β è prossimo all'unità (zone esterne allo strato limite di scarso interesse) ma

avere errori nelle zone in cui il fattore è prossimo a zero. Pertanto risulta utile ridurre il database originale: si potrebbe pensare di utilizzare la distanza normalizzata da parete per capire se si è lontani o vicini al corpo ma si perderebbero informazioni sulla scia. La strada più conveniente sembra quella di utilizzare il Mach convettivo per eseguire questa riduzione e "pulizia". Tuttavia questo presuppone la conoscenza del numero di Mach convettivo, che spesso è ignoto. Per ovviare il problema Paciorri e Sabetta (2003) propongono un'approssimazione del numero di mach convettivo locale $\tilde{M}_c$ espresso in funzione di quantità locali note, definito da un'equazione non lineare

$$\tilde{M}_c^2 f_2(\tilde{M}_c) = \frac{1}{4\tau_i}\frac{\tilde{\nu}|\omega|}{a^2} \tag{3.2}$$

dove la funzione $f_2(\tilde{M}_c)$ è definita come

$$f_2(\tilde{M}_c) = 0.44\left[\frac{1}{1+14\tilde{M}_c^5}\right] + 0.56$$

Nell'espressione 3.2 il termine τ_i è il massimo sforzo di taglio adimensionale e posto pari a 0.01.
Il database è quindi stato ridotto considerando solo i punti in cui $\tilde{M}_c > 10^{-3}$. La scelta è motivata considerando la figura 3.3, in cui si l'isolinea viola corrisponde al valore soglia e si nota chiaramente che tale linea crea una separazione tra il campo di moto inviscido in cui il fattore di correzione vale 1, e la zona dello strato limite e della scia che occorre analizzare. Nonostante questa riduzione, il database risulta ancora pieno di valori per

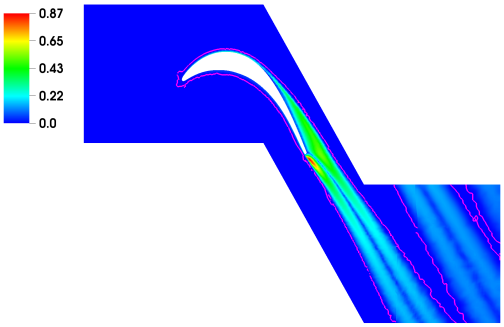


Figura 3.3: Distribuzione del Mach convettivo locale per $Re_{2_s} = 8 \cdot 10^4$

cui $\beta = 1$, per tale motivo si è pensato di adottare un sistema di "pesatura". Tutti i punti per cui $\beta > 0.9$ vengono contati una sola volta, mentre i restanti vengono contati 10 volte. Questo consente di creare un database in cui siano presenti molti più valori di β prossimo a zero, riducendo il problema indotto dalla stima dell'errore quadratico medio. Più nello specifico queste procedure sono descritte nel paper *Field Inversion and Machine Learning for improving RANS modelling in turbomachinery* di Ferrero et al. che verrà presentato nella conferenza ETC14 ad Aprile 2021 ([18]).

3.3 Linee guida alla risoluzione del problema

Lo scopo del lavoro di tesi è quello di riuscire a predire la transizione e separazione sul dorso del paletta di turbina a bassa pressione T106c a diversi numeri di Reynolds. A tal

proposito si considera il set di equazioni RANS con modello di turbolenza ad una equazione Spalart-Allmaras e si applica il paradigma Field Inversion and Machine Learning proposto da Parish e Duraisamy ([32]): inizialmente viene applicato il metodo dell'inversione del campo per ottenere il fattore di correzione $h(\beta(x))$ da applicare al modello di turbolenza. Successivamente, questo campo viene processato da diverse tecniche di machine learning (programmazione genetica, reti neurali e random forest) al fine di trovare una relazione funzionale tra il campo di correzione $\beta(x)$ e delle variabile fisiche note $\beta(\Phi)$. Ottenuta questa relazione, si può eseguire un'analisi CFD sulla paletta allo scopo di visualizzare la separazione sul dorso e confrontare i dati con quelli sperimentali. La procedura è riassunta nello schema in figura 3.4.

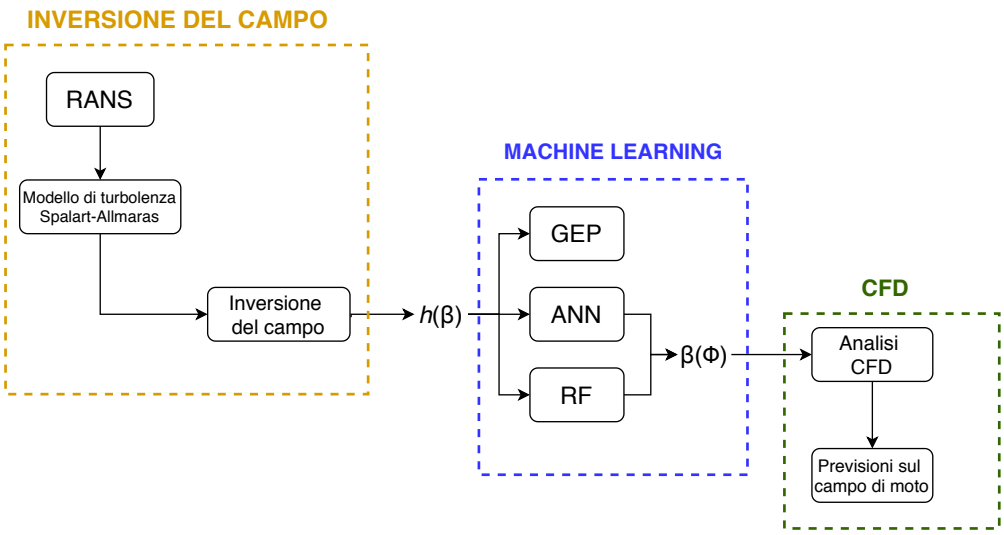


Figura 3.4: Linee guida alla risoluzione del problema

La procedura di inversione del campo e del metodo dell'aggiunto è presentata nella sezione 2.3, il database generato è spiegato invece nel paragrafo 3.2. Successivamente l'analisi delle tecniche di machine learning è effettuata in tre diversi capitoli: programmazione genetica, reti neurali artificiali e random forest, rispettivamente nei capitoli 4, 5, 6. Con i risultati ottenuti si effettua un'analisi CFD nel capitolo 7 nel quale vengono presentate le visualizzazioni del campo e i confronti tra le tecniche di machine learning, i modelli originali e le analisi sperimentali. Infine, i modelli ottenuti dalle tecniche di Machine Learning verranno utilizzati per fare delle previsioni a numeri di Reynolds differenti, non inclusi nell'approccio di inversione del campo e nell'addestramento allo scopo di analizzare quanto generale è il modello risultante.

Capitolo 4

Programmazione genetica

4.1 Generalità

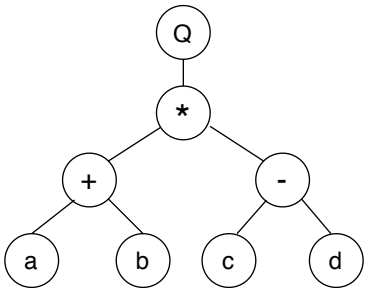
La programmazione ad espressione genetica (Gene Expression Programming, GEP) è stata inventata da Ferreira nel 1999 ed include sia algoritmi genetici che programmazione genetica. Nello specifico, gli algoritmi genetici (Genetic Algorithm, GA) sono stati introdotti per la prima volta da Holland nel 1990 con conseguente applicazione informatica nel 1975 ([21]): questi algoritmi sono una semplificazione dell'evoluzione biologica. L'aspetto negativo di questi algoritmi è che i cromosomi (la parte principale) fungono sia da genotipi che fenotipi, in altre parole sono l'oggetto di selezione ma sono anche i custodi dell'informazione genetica e per tal motivo non sono facilmente manipolabili. Nel 1985 Cramer ([10]) inventa la programmazione genetica (Genetic Programming, GP), successivamente sviluppata da Koza, nel 1992 ([25]), creando un sistema di rappresentazione più versatile rispetto al precedente. Come già anticipato la programmazione ad espressione genetica si basa sia su cromosomi semplici, lineari e lunghezza fissata introdotti negli algoritmi genetici, ma anche su alberi funzionali più complessi e ramificati tipici della programmazione genetica. In questo tipo di programmazione si utilizza un algoritmo in cui è presente una popolazione di individui iniziale. Ad ogni iterazione viene selezionato l'individuo più promettente in accordo con una funzione di fitness e vengono poi apportate delle modifiche genetiche ai cromosomi per creare una nuova generazione di individui. È importante sottolineare che esistono due linguaggi differenti in GEP: quello relativo ai geni e quello relativo agli alberi funzionali. Il gioco reciproco tra cromosomi e alberi funzionali nella programmazione genetica implica una traduzione univoca dei due diversi linguaggi. I vantaggi di questo sistema sono molteplici:

- i cromosomi possono essere entità semplici, lineari, compatti e facili da manipolare.
- gli alberi sono espressione esclusivamente dei rispettivi cromosomi. Quando il cromosoma subisce mutazioni genetiche, necessariamente anche il rispettivo albero si modifica.

L'elemento fondamentale è il cromosoma: è una stringa di lunghezza fissata costituito da uno o più geni. Per comprendere meglio la struttura di un albero funzionale e la sua relazione univoca con il singolo cromosoma è utile considerare il seguente esempio. Si vuole tradurre l'espressione analitica

$$\sqrt{(a+b)\cdot(c-d)}$$

in termini di albero funzionale e stringa costituente del cromosoma. L'albero assume la forma:



La relativa rappresentazione in cromosoma è:

0 1 2 3 4 5 6 7
Q∗ +−a b c d

Da questo esempio possiamo evincere delle considerazioni:

- per tradurre l'albero in cromosoma bisogna leggerlo dall'alto in basso e da sinistra verso destra;
- il linguaggio dei cromosomi è comunemente chiamato *Karva*, e l'espressione che ne deriva è detta espressione-K;
- in questo esempio il cromosoma è costituito da un singolo gene, da posizione 0 a posizione 7;
- l'albero riesce a rappresentare l'intero cromosoma. È bene notare che non sempre questo accade, spesso l'albero si interrompe prima, ma questo aspetto verrà analizzato più nel dettaglio in seguito;
- è possibile eseguire anche il processo inverso: dal cromosoma si costruisce l'albero. La posizione 0 corrisponde alla radice dell'albero, sotto ogni funzione vengono inseriti tanti rami quanti sono gli argomenti della funzione stesso, in questo esempio le tre funzioni analitiche hanno 2 argomenti ciascuna. L'assemblaggio è terminato quando nella parte finale dell'albero sono presenti solo dei terminali, cioè delle costanti non ramificabili.

4.2 Algoritmo di funzionamento

Per spiegare nel dettaglio il funzionamento dell'algoritmo si può far riferimento al diagramma di flusso in figura 4.1 Il processo inizia con la creazione casuale di un certo numero di cromosomi che definiscono la popolazione iniziale. Da questo punto consideriamo come entità indistinguibili il cromosoma e l'individuo, possiamo quindi dire che una popolazione è costituita da un certo numero di cromosomi o di individui. Una volta che la popolazione iniziale è stata creata, si valuta l'idoneità di ogni individuo, attraverso la funzione di fitness già introdotta; in relazione a questa gli individui sono selezionati per riprodursi con delle modifiche, dette mutazioni genetiche, al fine di creare una nuova generazione. Tutto questo si ripete un certo numero di generazioni fino al raggiungimento di una soluzione soddisfacente in accordo con dei requisiti di arresto.

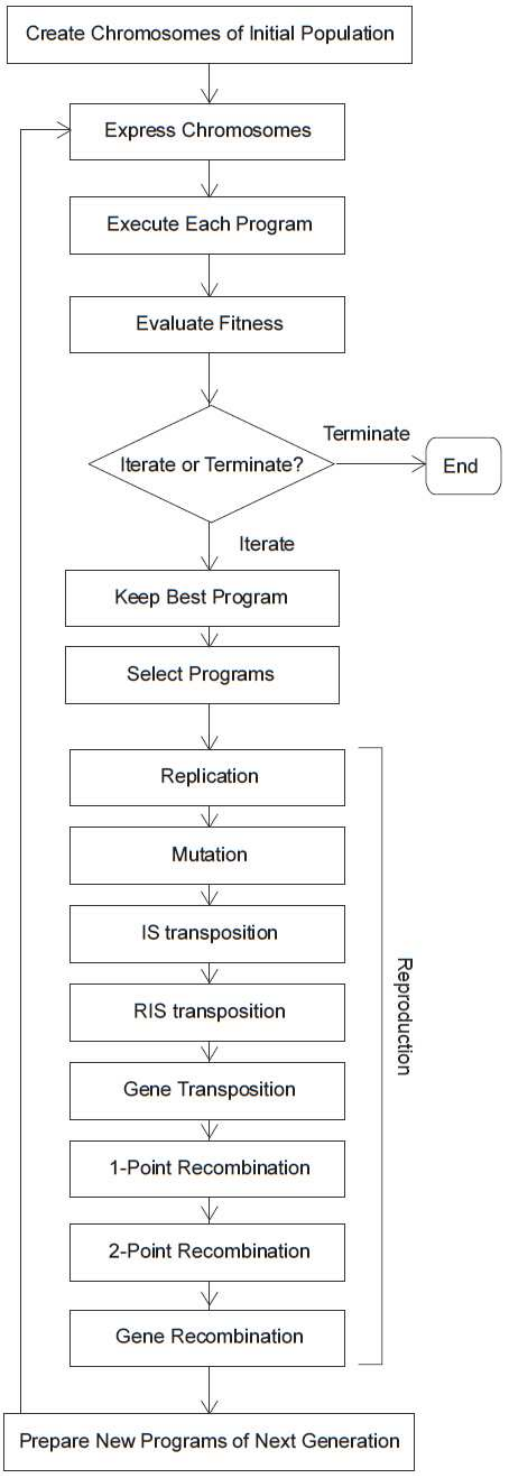


Figura 4.1: Diagramma di flusso programmazione genetica [13].

4.2.1 Popolazione e cromosomi

Come già anticipato ogni cromosoma è una stringa di caratteri di lunghezza fissata, costituito da uno o più geni, anch'essi di lunghezza determinata a priori. Un certo numero di cromosomi rappresenta una popolazione. Ogni gene è caratterizzato da una testa (head, h) e da una coda (tail, t). La testa può contenere sia funzioni che terminali, mentre la coda può contenere solo terminali. Per ogni problema la lunghezza della testa è scelta

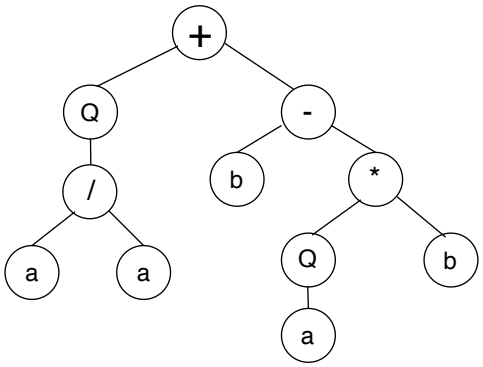
a priori mentre quella della coda dipende da h e dal massimo numero di argomenti per funzione n presente nel set di funzioni, secondo la relazione

$$t = h(n-1) + 1$$

A titolo di esempio consideriamo un gene composto da un set di funzioni $\{F\} = \{Q, *, /, +, -\}$ e un set di terminali $\{T\} = \{a, b\}$. Il massimo numero di argomenti per le funzioni è $n = 2$. Prendiamo il seguente cromosoma:

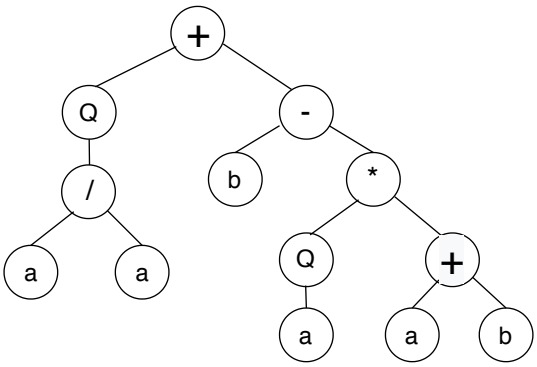
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
+Q−/ b ∗ a a Qb **a a b a a b b a a a b**

La coda del gene è rappresentata in grassetto. Si ha $h = 10$, $n = 2$ e di conseguenza $t = 11$. La lunghezza del gene totale è $l = h + t = 21$. La sua rappresentazione in albero funzionale è



Si nota immediatamente che il gene termina in posizione 20 mente l'albero in posizione 10, perché non può più essere ramificato. È proprio qui che risulta utile il concetto di mutazione genetica, che verrà comunque approfondito in seguito. Supponiamo di sostituire il terminale b in posizione 9 con l'operatore $+$, il cromosoma e il relativo albero diventano:

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
+Q−/ b ∗ a a Q+ **a a b a a b b a a a b**

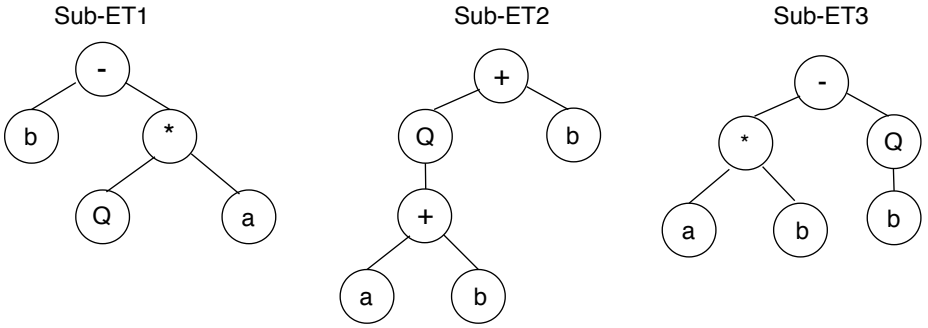


L'albero è diventato più ramificato, ora si stoppa in pozione 12 piuttosto che in 10. Si può concludere che sebbene ogni gene sia di lunghezza fissata ha la potenzialità di formare

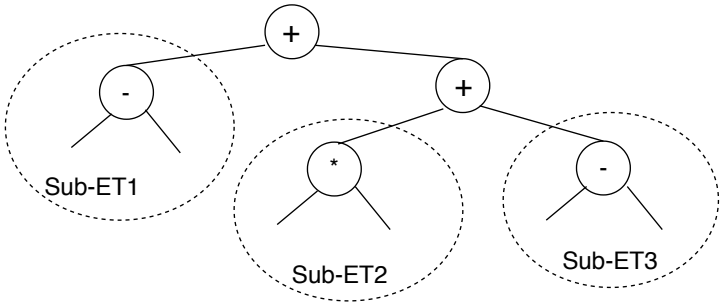
alberi di diverse forme e profondità.
Fino ad ora abbiamo considerato cromosomi costituiti da un singolo gene, ma molto frequentemente si usano i cosiddetti cromosomi multigenetici. Essi hanno un ulteriore grado di libertà, dato dalla scelta del numero di geni che compongono il cromosoma stesso. Ogni gene codifica un sotto albero (sub-expression tree, sub-ET), una volta ottenuti i sotto alberi, questi devono essere in qualche modo collegati per ottenere un'espressione finale univoca. Consideriamo il seguente cromosoma composto da 3 geni, ognuno di lunghezza 9, da posizione 0 a posizione 8:

$$\begin{array}{c} 0\,1\,2\,3\,4\,5\,6\,7\,8\,0\,1\,2\,3\,4\,5\,6\,7\,8\,0\,1\,2\,3\,4\,5\,6\,7\,8 \\ -\mathrm{b}*\mathrm{b}\,\mathbf{abba}\,\mathbf{b}*\mathrm{Qb}+\mathbf{abbba}-*\mathrm{Qa}\,\mathbf{bbaba} \end{array}$$

Si possono costruire i relativi sotto alberi:



Ogni albero è sia un'entità separata che una parte di una struttura molto più complessa ed è bene ricordare che l'insieme è superiore alla somma delle parti. Uno dei modi più semplici per collegare i singoli sotto alberi in un unico grande albero è attraverso una funzione, la cosiddetta *linking function*. Ad esempio, se si considera come linking function l'addizione, si avrebbe:



La funzione di collegamento è scelta a priori per ogni problema.
Alla luce di quanto mostrato possiamo fare delle ultime considerazioni:

- La lunghezza della testa h di ogni gene determina il massimo peso e profondità dei sotto alberi codificati in ogni gene. Il peso w e la profondità d sono espressi da:

  $$w = (n-1)h + 1 \qquad d = \frac{h+1}{m} \cdot \frac{m+1}{2}$$

  dove m è il minimo degli argomenti delle funzioni mentre n è il massimo;

- per ogni problema c'è una lunghezza del cromosoma che permette di ottenere l'evoluzione più efficiente;

- i cromosomi più compatti non è detto che siano i più efficienti, c'è bisogno di una certa ridondanza per permettere una buona evoluzione;
- il numero di geni in un cromosoma è un parametro importante, dividendo i cromosomi in unità più facilmente manipolabili si ottiene un modello più elegante ed efficiente.

4.2.2 Funzione di fitting

Per valutare la bontà dei risultati è necessario possedere un criterio, in questo caso una funzione che misura la distanza tra i dati originali e i dati predetti dal modello evolutivo. Diverse funzioni possono essere adottate e la loro prima classificazione dipende dall'ambiente di lavoro. Generalmente si distingue tra due problemi principali:

- problemi di regressione: l'obiettivo è trovare una correlazione analitica tra i vari dati;
- problemi di classificazione: si cerca di classificare i dati in alcune sezioni.

Nel nostro caso ci occuperemo di un problema di regressione e più nello specifico di *function finding*, cioè la ricerca di una funziona analitica che permetta di ottenere l'output a partire da una serie di input.
Diverse funzioni esistono per tale scopo, alcune basate sull'errore assoluto tra i valori predetti e quelli di target ed altre sull'errore relativo piuttosto che su indici statistici. La scelta di una di queste funzioni non è una questione banale e non può essere scelta dallo studio del problema stesso. Di solito si preferisce utilizzare funzioni con misure standard, come quella basata sull'errore quadratico medio (Mean Square Error, MSE) o sul coefficiente Rsquare, perché sono universali e possono essere usate per ogni tipo di problema.
L'errore quadratico medio E_i di un programma i è espresso da due formulazioni: la prima si riferisce all'errore assoluto, mentre la seconda a quello relativo:

$$E_i = \frac{1}{n} \sum_{j=1}^{n} \left(P_{(ij)} - T_j \right)^2 \tag{4.1}$$

$$E_i = \frac{1}{n} \sum_{j=1}^{n} \left(\frac{P_{(ij)} - T_j}{T_j} \right)^2 \tag{4.2}$$

dove $P_{(ij)}$ è il valore predetto dal programma i per il caso j, T_j è il valore di target per il caso j. Una previsione perfetta si ha quando $P_{(ij)} = T_j$ e di conseguenza $E_i = 0$, per cui si conclude che l'errore varia dal valore 0 a infinito, dove il minimo è il caso ideale. La funzione di fitness f_i dipende dall'errore ed è definita come:

$$f_i = 1000 \cdot \frac{1}{1 + E_i} \tag{4.3}$$

e varia nel range tra 0 e 1000, dove 1000 è l'accoppiamento perfetto tra i valori di previsione e quelli di target.
Il coefficiente menzionato *Rsquare* è la radice del coefficiente di correlazione di Pearson. Questo indice varia nell'internvallo $[-1,1]$ e rappresenta la relazione lineare tra i valori predetti e quelli di target. Quando $R^2 = 1$, c'è una correlazione lineare perfetta, invece quando $R^2 = -1$ c'è una perfetta correlazione anti-lineare. Infine, quando $R^2 = 0$ non c'è alcuna relazione tra i dati.

4.2.3 Set di funzioni e terminali

Un ulteriore grado di libertà è la scelta del set di funzioni utilizzabili negli alberi funzionali e delle costanti. Generalmente è possibile scegliere ogni tipo di funzione e il peso che questa deve avere nel modello. Consideriamo un generico set di funzioni $\{F\} = \{F_1, F_2, \dots F_n\}$, le generiche funzioni F_i possono essere

- funzioni algebriche, i classici operatori principali: $+, -, *, /, \sqrt{\cdot}$ ecc
- funzioni goniometriche, di tipo $\sin, \cos, \tan, \tanh$ ecc
- cicli ricorsivi: *if, do, for, while* ecc
- operatori booleani: *AND, OR, NOT* ecc

la scelta delle funzioni da utilizzare dipende dai dati del problema. Ad esempio si vedrà che nel nostro caso l'output β varia nel range [0,1] ed è opportuno quindi scegliere funzioni che presentano degli asintoti orizzontali, come la tangente iperbolica o l'arcotangente, oltre che agli operatori classici.
Un ulteriore grado di libertà è rappresentato dal set dei terminali $\{T\} = \{a_1, a_2, \dots, a_n\}$, è possibile scegliere sia il numero di costanti utilizzabili per gene che il loro range di variazione numerico.

4.2.4 Operatori genetici

Introdotta la popolazione iniziale, selezionati gli individui, essi sono chiamati a riprodursi con mutazioni creando diversificazioni genetiche che permettono la corretta evoluzione delle successive generazioni. Le principali mutazioni genetiche sono:

- **Replicazione**: i cromosomi sono semplicemente copiati nella generazione successiva.
- **Mutazione**: può avvenire in ogni punto del cromosoma, mantenendo invariata la sua struttura. Nella testa del gene alcuni simboli possono modificarsi in funzioni o terminali, mentre nella coda solo in terminali. Ad esempio

  0 1 2 3 4 5 6 7
  Q∗ +−a b c d

  Modificando la posizione 2 del gene e sostituendo la moltiplicazione $*$ con la divisione $/$, si ottiene il cromosoma mutato:

  0 1 2 3 4 5 6 7
  Q/ +−a b c d

  Si nota come la struttura del cromosoma sia rimasta invariata, in termini di lunghezza, ma l'albero risultate potrebbe variare nel profondo, soprattutto considerando un cromosoma multigenetico più complesso.
- **Trasposizione**: alcuni frammenti di un gene possono essere trasferiti in altre parti del cromosoma. Ci sono tre tipi di elementi trasponibili:

1. **IS** (sequenza di inserzione): piccoli frammenti possono essere spostati nella testa del gene ma non nella radice (prima posizione del primo gene del cromosoma). Ad esempio, consideriamo un cromosoma composto da due geni, come di seguito:

   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
   ∗ −+∗ a −+a ∗ b b a b b a a b a b a b Q∗ ∗ +a b Qb b ∗ a a **bba** a a a b b a

   Supponiamo che la sequenza **bba** nel secondo gene, dalla posizione 12 alla 14, sia scelta per essere un elemento IS e sia trasferita nel primo gene, tra la posizione 5 e 6. In questo caso si ottiene:

   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
   ∗ −+∗ a −**bba**+b a b b a a b a b a b Q∗ ∗ +a b Qb b ∗ a a **bba** a a a b b a

   Durante la trasposizione, la sequenza a monte dell'inserzione rimane invariata mentre quella a valle perde tanti simboli quanta è la lunghezza dell'elemento IS.

2. **RIS** (sequenza di inserzione alla radice): dei piccoli frammenti che cominciano con una funzione possono esse trasferiti anche alla radice del gene selezionato. Considerando per semplicità l'esempio precedente:

   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
   ∗ −+∗ a −+a ∗ b b a b b a a b a b a b Q∗ ∗ +**a b** Qb b ∗ a a b b a a a a b b a

   Supponiamo di prende la sequenza +**ab** del secondo gene, dalla posizione 3 alla posizione 5 e posizionarla alla radice del gene stesso, ottenendo:

   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
   ∗ −+∗ a −+a ∗ b b a b b a a b a b a b +**a b** Q∗ ∗ +**a b** Q a a b b a a a a b b a

   come nel caso procedente, a valle del gene, nella testa, si modifica l'informazione per poi continuare indisturbata.

3. Trasposizione di geni: non solo dei frammenti di gene possono essere spostati, ma anche il gene intero all'inizio del cromosoma. A differenza degli altri di IS e RIS, in questo caso il gene è eliminato nel luogo di origine al fine di mantenere la lunghezza originale del cromosoma. Consideriamo il seguente cromosoma composto da 3 geni:

   0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8
   ∗ a −∗ a b b a b −**QQ/ a a a b b** Q+a b a b a b b

   Supponiamo che il secondo gene subisca la transizione e si sposta quindi all'inzio del cromosoma:

   0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8
   −**QQ/ a a a b b** ∗ a −∗ a b b a b Q+a b a b a b b

   È bene notare che se la linking function scelta per legare i sotto alberi è l'addizione, questo genere di trasposizione non implica nessuna variazione. Risulta molto importante, invece, quando la funzione di link è non commutativa.

- **Ricombinazione**: due cromosomi sono scelti per scambiarsi reciprocamente del materiale genetico. Anche in questo caso è possibile distinguere tre casi:

1. Ricombinazione ad un punto: tra due cromosomi c'è uno scambio da un certo punto in poi. Ad esempio, considerando due cromosomi a due geni:

   0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8
   −b +Qb b a b b / a Qb b b a a b
   / −a / a b a b b −b a −a b a a a

   Ipotizziamo di eseguire una ricombinazione prendendo come punto di riferimento nel primo gene tra la posizione 2 e la posizione 3, ottenendo:

   0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8
   −b +/ a b a b b −b a −a b a a a
   / −a Qb b a b b / a Qb b b a a b

2. Ricombinazione a due punti: avviene la stessa cosa del precedente ma vengono scelti due punti di scambio e non uno solo.
3. Ricombinazione genetica: la differenza dai precedenti casi è che interi geni vengono scambiati fra i due cromosomi.

Sono state elencate solo le principali tra le molteplici mutazioni genetiche, ulteriori approfondimenti possono essere trovati nel libro *Gene Expression Programming*, Ferreira ([14]).

4.3 Risultati

Il software utilizzato per analizzare gli algoritmi di programmazione genetica è *GeneXProTools*. Per il problema di *function finding*, cioè della ricerca di una funzione che colleghi in modo analitico un certo numero di input all'output, è necessario inserire l'intero database. Per questa simulazione è stato considerato il database completo ottenuto per entrambi i numeri di Reynolds, considerando un totale di sei input (descritti in tabella 3.1 ad eccezione del gradiente di pressione) e l'output β. Per la simulazione si è considerato il 75% del database per il training dell'algoritmo e il restante 25% per la validazione. I parametri inseriti nel software sono elencati in tabella 4.1.

Input	Valore
Numero di individui	50
Numero di geni	4
Lunghezza della testa	7
Funzione di linking	+
Funzione di fitness	RMSE
Set di funzioni	$+, -, *, /, \exp, \ln, 1/x, x^2, \sqrt[3]{x}, min2,$ $max2, avg, \arctan, \tanh, (1-x)$
Numero di costanti per gene	70 ∈[-100,100]
Operatori genetici	Optimal evolution

Tabella 4.1: Lista degli input per il software GeneXProTool

Per quanto riguarda il set di funzioni è bene fare delle precisazioni per trovare una giusta corrispondenza negli alberi funzionali:

$1/x$ è la funzione inversa, nell'albero apparirà con il simbolo *inv*, $(1-x)$ è la funzione complementare, rappresentata con NOT; infine, *3Rt* sta per la radice cubica del valore. In questa simulazione si nota che il fitting dei dati non ottimale, il coefficiente Rsquare, già analizzato, è molto basso. Nello specifico si ottiene

$$R^2 = 0.3998 \qquad \text{best}(f_i) = 865.55$$

dove il secondo valore è il miglior fitness ottenuto, corrispondente al valore di R^2. Inoltre specifichiamo che i precedenti risultati sono stati ottenuti per i dati di training, ma sono molto simili anche per quelli di validazione.
In figura 4.2 si possono visualizzare i due valori e notare come la crescita del coefficiente di Pearson sia molto molto lenta.

Figura 4.2: Grafico Fitness e coefficiente di regressione

Avendo scelto 4 geni per cromosoma ci saranno 4 sotto alberi funzionali, da collegare tra di loro con la funzione di link imposta (in questo caso l'addizione). La rappresentazione grafica dei 4 sotto alberi è riportata in figura 4.3. I principali problemi del fitting av-

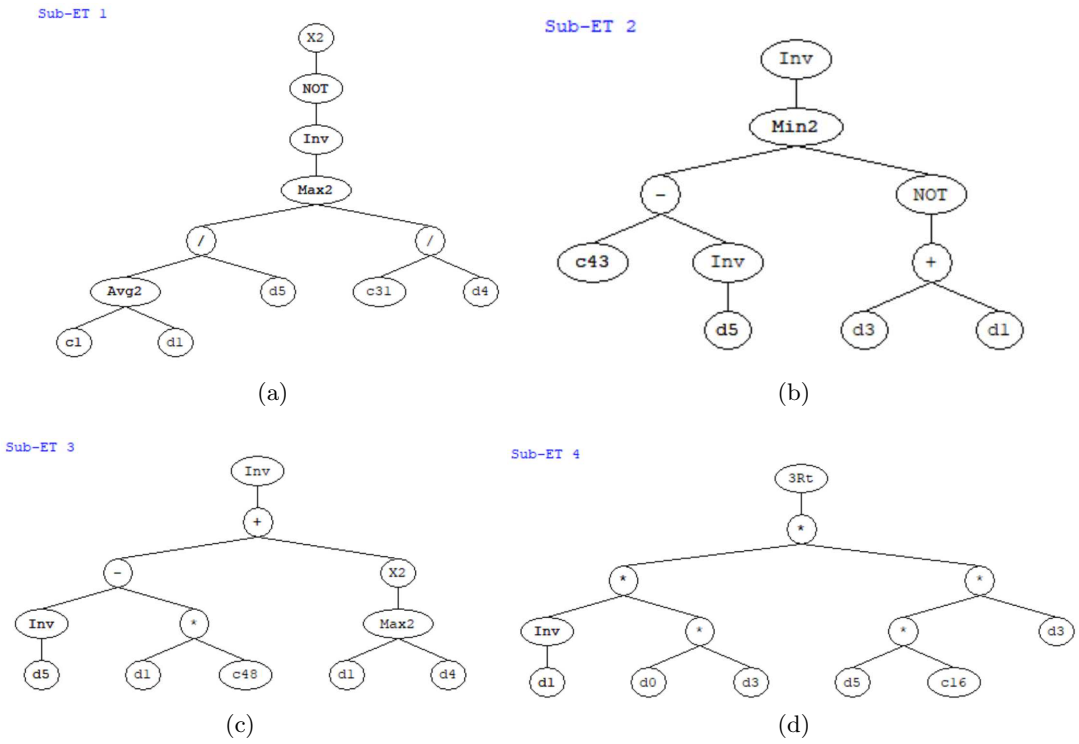


Figura 4.3: Sotto alberi: subET1 (a), subET2 (b), subET3 (c), subET4 (d)

vengono quando l'output assume valori molto bassi, vicini allo zero. Questa situazione è mostrata bene in figura 4.4.

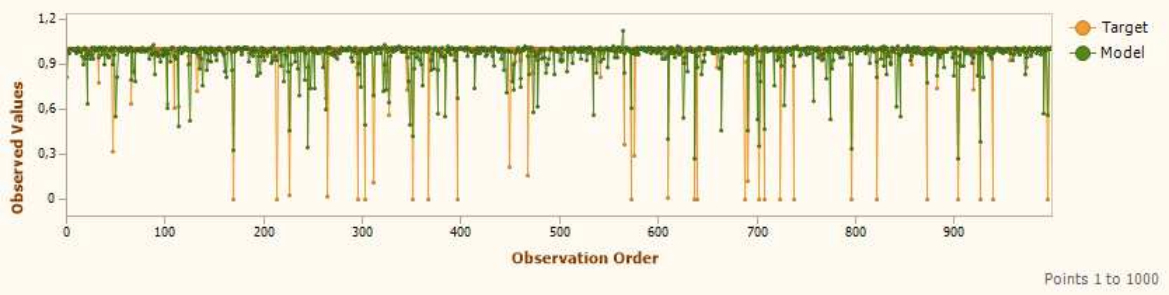


Figura 4.4: Dati di target in confronto con il modello

Questa tecnica non ha portato a risultati ottimali, un'ulteriore conferma si può ottenere visualizzando il diagramma di regressione in figura 4.5. I valori predetti sono molto

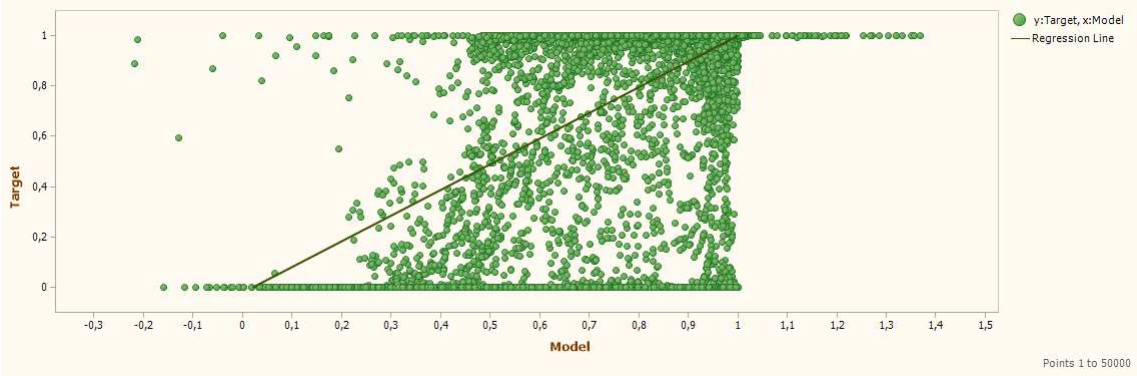


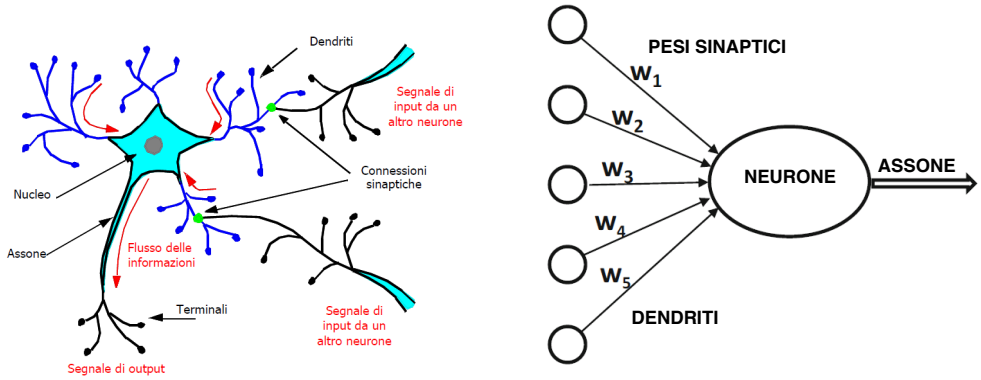
Figura 4.5: Scatter plot programmazione genetica.

distanti da quelli di target, gli errori sono ancora eccessivi e per tal motivo si è scelto di non effettuare un'analisi CFD poiché non potrebbe portare a nessun miglioramento cercato nel modello. Le tecniche di programmazione genetiche sono attualmente in grande sviluppo e dovranno essere analizzate molto più nello specifico.

Capitolo 5

Reti Neurali Artificiali

Le reti neurali artificiali (Artificial Neural Networks, ANN) sono delle tecniche di Machine Learning con lo scopo di simulare il funzionamento del cervello biologico. Per meglio capire la loro struttura descriviamo brevemente il sistema nervoso umano: esso è composto da miliardi di cellule, dette **neuroni**, che scambiano informazioni tra di loro attraverso dei collegamenti, comunemente detti **assoni** e **dendriti**. Le regioni di collegamento tra assoni e dendriti sono dette **sinapsi** e sono molto importanti perché sono in grado di cambiare forma in base all'informazione ricevuta e agli stimoli esterni. Volendo replicare questo comportamento in modo computazionale, si considerano delle unità computazionali, i neuroni, connessi tra di loro con dei **pesi** variabili, che svolgono una funzione simile alla sinapsi. In linea generale una semplice rete neurale è composta da neuroni di input e neuroni di output, questi sono connessi attraverso un sistema di pesi variabili e funzioni di attivazioni; i pesi vengono aggiornati in modo iterativo (e questa è la fase di apprendimento) fino al raggiungimento del risultato desiderato. In figura 5.1 si può notare un confronto tra uno schema semplificato del sistema nervoso biologico e una banale rete neurale. È anche noto che il sistema nervoso riceve degli stimoli dall'esterno, nel caso



(a) (b)

Figura 5.1: Sistema nervoso biologico (a), rete neurale artificiale (b) [1]

della rete neurale questi stimoli sono forniti dai dati in ingresso (che chiameremo target) del problema ai quali il modello finale deve tendere con più accuratezza possibile.
Le caratteristiche principali di una rete neurale sono tre:

- architettura della rete: struttura e tipologia delle connessioni tra i nodi, numero di livelli intermedi;

- dipendenza funzionale tra input e output: scelta della funzione di attivazione;
- modalità di apprendimento: metodo per aggiornamento dei pesi.

Questi tre aspetti verranno spiegati più nel dettaglio di seguito.

5.1 Architettura della rete

L'architettura della rete è un aspetto fondamentale nella risoluzione del problema. I gradi di libertà sono la scelta del numero dei livelli intermedi (strati nascosti) e del numero di neuroni per ogni strato. Si vedrà che è una scelta di compromesso, al fine di ottenere buone previsioni ma evitando il problema dell'overfitting. Generalmente le due distinzioni principali sono tra una rete a singolo strato, il classico percettone, o una rete multistrato.

5.1.1 Rete neurale a singolo strato. Il percettone

Una rete di questo tipo è caratterizzata da un singolo strato di dati di input e un nodo di output. Tra i due sono presenti i pesi, necessari all'apprendimento, e la funzione di attivazione.
Ad esempio, consideriamo un insieme di dati di input (training set), in cui $\bar{X}$ sono le variabili di input e y è la variabile di output, cosi formati:

$$\bar{X} = [x_1, x_2, \ldots, x_d] \qquad y \in \{-1,1\}$$

Lo strato di input contiene i pesi $\bar{W}$:

$$\bar{W} = [w_1, w_2, \ldots, w_d]$$

ovviamente della stessa dimensione di $\bar{X}$.
In figura 5.2 è rappresentata questa semplice rete a singolo strato All'interno del nodo di

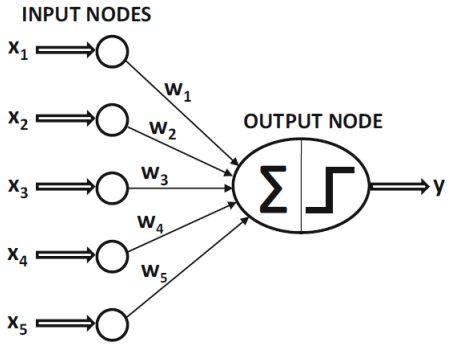


Figura 5.2: Rete neurale a singolo strato ([1])

output sono presenti due simboli, la sommatoria e la funzione segno. Il primo rappresenta la combinazione lineare che avviene tra gli input e i pesi, la seconda è la funzione di attivazione che trasforma il valore della combinazione lineare in un numero compreso tra -1 e 1, coerente con il range di variazione di y.
A livello matematico, la combinazione lineare è

$$\bar{W} \cdot \bar{X} = \sum_{i=1}^{d} w_i x_i$$

e la funzione attivazione agisce in questo modo:

$$\hat{y} = \text{sgn}\left\{\bar{W} \cdot \bar{X}\right\} = \text{sgn}\left\{\sum_{i=1}^{d} w_i x_i\right\}$$

Il valore $\hat{y}$ è il valore predetto dalla rete neurale, che deve essere quanto più simile possibile al valore originale y. Per cui, per ogni iterazione è possibile calcolare l'errore commesso:

$$E(\bar{X}) = y - \hat{y}$$

Nel caso in cui si abbia $E(\bar{X}) \neq 0$, i valori dei pesi devono essere modificati. L'aggiornamento dei pesi avviene nel seguente modo:

$$\bar{W} \Rightarrow \bar{W} + \alpha(y - \hat{y})\bar{X} = \bar{W} + \alpha E(\bar{W})\bar{X}$$

dove α è il rateo di apprendimento della rete neurale. L'algoritmo modifica iterativamente i pesi fino al raggiungimento della convergenza, ogni ciclo iterativo prende il nome di **epoca**.
Uno dei problemi di predizione è che i dati possono risultare sbilanciati, questo può produrre un effetto indesiderato, cioè che la media dei valori di output sia diversa da zero anche a posteriori dell'applicazione della funzione di attivazione di tipo segno. In questo caso è utile poter aggiungere una piccola unità computazionale detta *bias* che assume costantemente valore pari a 1 e ha un suo peso b, che varia come tutti i pesi w_i. In sintesi il bias serve ad evitare l'errore dovuto allo sbilanciamento dei dati, traslando la funzione nel punto giusto. Nel caso lineare in questione, il bias serve a traslare la retta nell'origine degli assi. Un'architettura di rete neurale a singolo strato con bias è mostrata in figura 5.3.

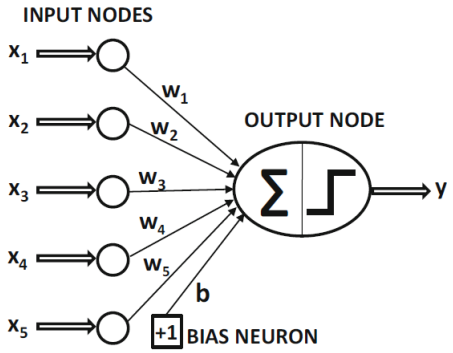


Figura 5.3: Rete neurale a singolo strato con bias ([1])

Il valore di predizione $\hat{y}$ subisce quindi una modifica:

$$\hat{y} = \text{sgn}\left\{\bar{W} \cdot \bar{X} + b\right\} = \text{sgn}\left\{\sum_{i=1}^{d} w_i x_i + b\right\}$$

e di conseguenza cambia il valore dell'errore e dell'aggiornamento dei pesi.
Bisogna notare che questa architettura funziona bene per variabili che sono linearmente separabili, ma ha molte difficoltà per altri tipi di problemi. Per questo motivo è necessario lo sviluppo di reti più complesse e generali.

5.1.2 Rete neurale multistrato

In una rete neurale multistrato, come suggerito dal nome, tra i nodi di input e quello di output sono presenti ulteriori strati, che prendono il nome di *strati nascosti* poiché le operazioni che contengono sono sono direttamente visualizzabili dall'utente. Questi strati sono organizzati sempre allo stesso modo, composti da un certo numero di neuroni collegati attraverso dei pesi agli strati successivi. Un esempio comprendente i bias è mostrato in figura 5.4. Una rete di questo tipo prende il nome di *feed forward* perché scorre sempre

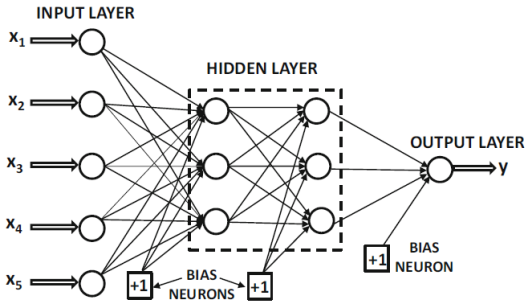


Figura 5.4: Rete neurale multistrato con bias ([1])

in avanti, da uno strato al successivo fino ad arrivare all'output.
La rete rappresentata in figura 5.4 è composta in totale da 3 strati, due nascosti e uno di output, lo strato di input non viene contato. I pesi ora sono contenuti in una matrice e non più in un vettore. Tra lo strato di input e il primo strato nascosto la matrice dei pesi $\bar{W}_1$ ha dimensione $d \times p_1$, dove p_1 è il numero di unità nel primo strato nascosto. In modo analogo tra lo strato nascosto r e il successivo $r+1$, la matrice dei pesi $\bar{W}_r$ ha dimensione $p_r \times p_{r+1}$. Infine, tra l'ultimo strato nascosto e l'output si ha $\bar{W}_{k+1}$ di dimensione $p_k \times s$, dove k è il numero totale di strati nascosti, s è il numero di nodi nell'output.
Il processo ricorsivo tra l'input e l'output è quindi il seguente:

$$\bar{h}_1 = \Phi\left(w_1^T \bar{X}\right) \qquad \text{tra input e primo strato nascosto}$$

$$\bar{h}_{p+1} = \Phi\left(w_{p+1}^T \bar{h}_p\right), \quad \forall p \in 1, \ldots, k-1, \text{ tra due strati nascosti}$$

$$\bar{s} = \Phi\left(w_{k+1}^T \bar{h}_k\right) \qquad \text{tra ultimo strato nascosto e output.}$$

Dove Φ è la generica funzione di attivazione, che verrà approfondita nella prossima sezione. I pesi si aggiornano iterativamente con dei metodi a scelta dall'utente.

5.2 Funzione di attivazione

La scelta della funzione di attivazione è un problema non banale, come già accennato l'esigenza di una funzione di attivazione non lineare si manifesta nel passaggio da una rete a singolo strato a una multistrato. Utilizzando la notazione Φ per questo tipo di funzione, generalmente si può esprimere l'output predetto come:

$$\hat{y} = \Phi(\bar{W}\bar{X} + b)$$

Il valore calcolato prima dell'applicazione di Φ è detto valore di pre-attivazione, in modo analogo il valore successivo prendere il nome di valore di post-attivazione. Un esempio

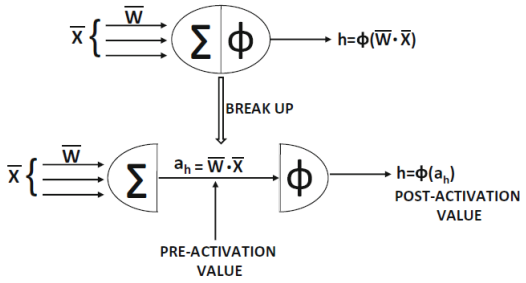


Figura 5.5: Valori di pre e post attivazione. ([1])

schematico è mostrato in figura 5.5. Ci sono molteplici funzioni non lineari utili da poter utilizzare. Sono graficamente rappresentate in figura 5.6 e di seguito espresse in forma matematica

- Funzione segno:

$$\Phi(v) = \text{sgn}(v)$$

  La funzione segno ha immagine nel range [0,1], risulta utile per problemi di stima della probabilità (figura 5.6(a)).

- Funzione sigmoide:

$$\Phi(v) = \frac{1}{1 + \text{e}^{-v}}$$

  Questa funzione ha la stessa variazione della precedente ma è continua, utile per problemi il cui output assume valori compresi tra 0 e 1 con continuità (figura 5.6(b)).

- Funzione tangente iperbolica:

$$\Phi = \frac{\text{e}^{2v} - 1}{\text{e}^{2v} + 1}$$

  Come la funzione sigmoide, la tangente iperbolica è continua e assume valori nel range tra -1 e 1 (figura 5.6(c)).

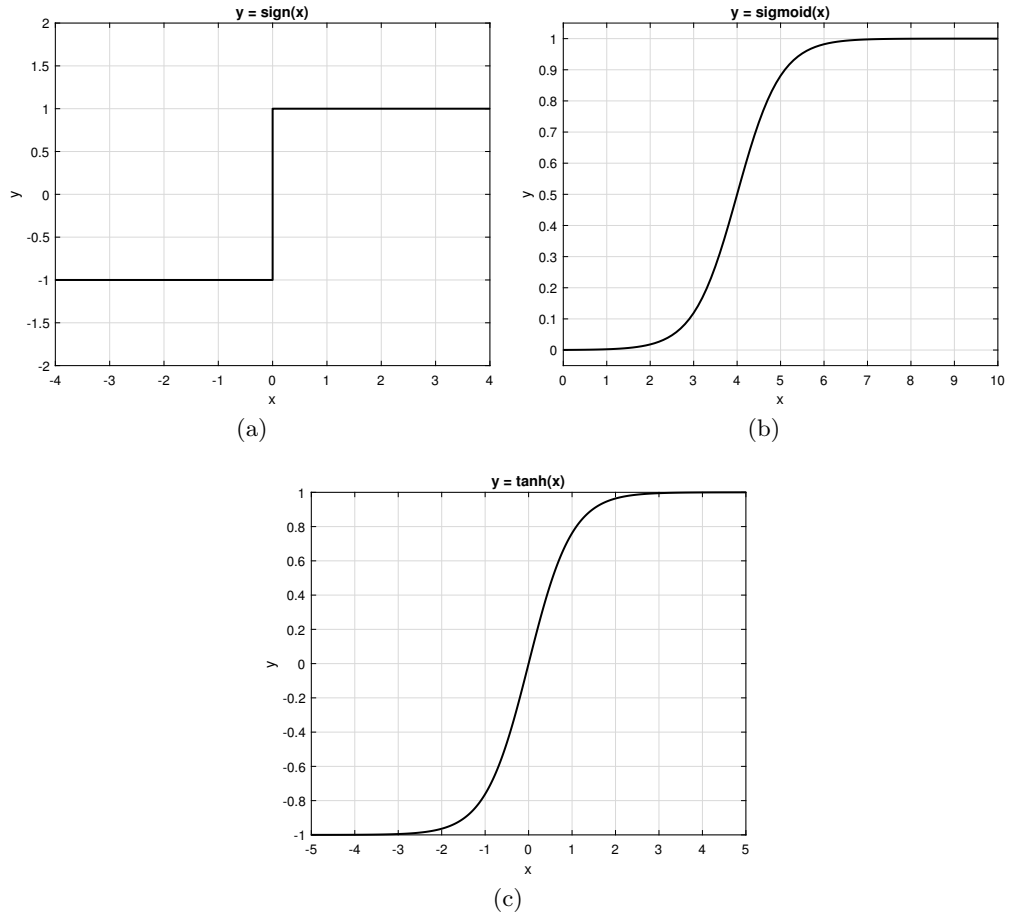
Ulteriori esempi di funzioni di attivazioni si possono trovare nel libro *Neural Networks and Deep Learning* (C.C.Aggarwal, [1]).

5.3 Minimizzazione dell'errore e aggiornamento dei pesi

L'obiettivo dell'algoritmo riguardante le reti neurali è di minimizzare l'errore della predizione rispetto al valore originale. Questo problema è spesso riconducibile al metodo dei minimi quadrati, e si traduce in:

$$\text{minimize}_w L = \sum (y - \hat{y})^2 \tag{5.1}$$

questo tipo di funzione di minimizzazione è detta **funzione di perdita**. Anche in questo caso le scelte sono molteplici, per semplicità se ne specificano soltanto due.



(a) (b) (c)

Figura 5.6: Esempi di funzione di attivazione: funzione segno (a), sigmoide (b), tangente iperbolica (c)

5.3.1 Metodo di Gauss-Newton

Considerando il vettore dei residui, con P pattern

$$E(w) = (e_1, \ldots . e_P)$$

il problema dei minimi quadrati assume la forma:

$$\min_w E = \frac{1}{2}||e(w)||^2$$

Le derivate dell'errore E si esprimono con la matrice Jacobiana $J = (\nabla e_1, \ldots, \nabla e_p)^T$, e il gradiente:

$$\nabla E = \sum_{p=1}^{P} e_p(w) \nabla e_p(w) = J^T(w) e(w)$$

Considerando inoltre la matrice Hessiana:

$$\nabla^2 E = \sum_{p=1}^{P} \nabla e_p(w) e_p^T(w) + \sum_{p=1}^{P} e_p(w) \nabla^2 e_p(w) = J(w) J^T(w) + Q(w)$$

Dove il termine Q_w è trascurabile rispetto al primo.
La regola di apprendimento e quindi di aggiornamento di pesi è la seguente

$$w^{k+1} = w^k + d^k$$

dove d^k è la soluzione del sistema:

$$(J^k)^T J^k d^k = -(J^k)^T e^k$$

5.3.2 Metodo di Levenberg Marquardt

L'algoritmo di Levenberg Marquardt (LMA) è un algoritmo di ottimizzazione usato per la risoluzione di problemi in forma dei minimi quadrati non lineari. È molto utilizzato per la sua velocità di convergenza e stabilità e il suo ciclo iterativo si svolge come segue:

1. si calcola la matrice Jacobiana dell'errore
2. si calcola il termine $(J^k)^T e^k$
3. si risolve il sistema $(J^k)^T J^k d^k = -(J^k)^T e^k$, per trovare il passo di iterazione d^k
4. si aggiornano i pesi $w^{k+1} = w^k + d^k$
5. si calcola il nuovo vettore dei pesi E
6. si ripete fino al raggiungimento della convergenza.

5.4 Overfitting

Il problema dell'overfitting si manifesta nel momento in cui il modello funziona molto bene nella previsione dei dati per cui è stato addestrato ma perde di validità per altri set di dati, mai visti prima. Questo accade specialmente quando il modello è poco generale e di conseguenza è estremamente specifico per i dati di input. Aumentare il numero di dati per l'addestramento certamente aiuta ma bisogna considerare anche che se i dati di training sono molti, il modello ha difficoltà a catturare le complessità insite nei dati. In generale si assume che i dati di training siano almeno 2 o 3 volte superiori rispetto al numero dei parametri della rete.
La tecnica più semplice per evitare questo fenomeno è quello di suddividere il database originale in due parti: una parte di training, che prende circa il 75% dei dati e uno di validazione che contiene il restante 25%. Si addestra la rete utilizzando i dati di training e successivamente si testa il modello sul database di validazione per valutarne l'efficacia. Un tipico esempio di overfitting è mostrato in figura 5.7.

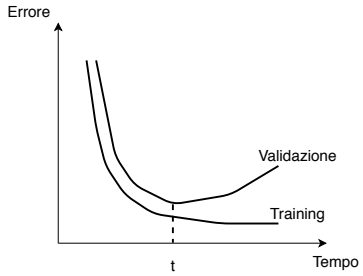


Figura 5.7: Andamento dell'errore nei due dataset

Si può interrompere l'addestramento al tempo t, non appena l'errore nel set di validazione inizia ad aumentare, mentre quello di training continua a diminuire.

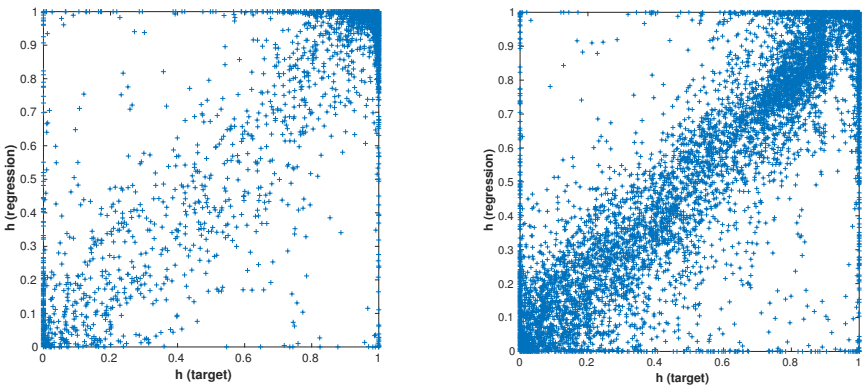
5.5 Risultati

Il database utilizzato per l'addestramento della rete è quello presentato nella sezione 3.2, e i seguenti quattro input vengono considerati:

Input	Descrizione
$\chi = \frac{\tilde{\nu}}{\nu}$	rapporto tra viscosità turbolenta e cinematica
f_d'	shedding function modificata
$\frac{\nabla p \cdot \mathbf{u}}{p\lvert\mathbf{u}\rvert} d$	gradiente di pressione adimensionale
$\tilde{M}_c$	Numero di Mach convettivo locale

Tabella 5.1: Input addestramento rete neurale

In particolare sono stati considerati 3 sottoinsieme del database: il primo contiene il 75% dei dati utilizzato come training, il secondo contiene un 15% per la validazione e per controllare che l'errore in questo sottoinsieme non aumenti mentre il restante 15% è stato utilizzato come sottoinsieme di test, per testare la rete su un insieme di dati che non hanno fatto parte dell'addestramento e non hanno inciso sull'errore quadratico medio. I grafici di regressione sono stati considerati proprio su quest'ultimo. L'addestramento è stato effettuato considerando l'algoritmo di Levenberg-Marquardt e come errore la funzione MSE (Mean Square Error), la rete neurale considerata è costituita da 2 strati nascosti da 20 neuroni ciascuno.
In figura 5.8 si può notare un confronto effettuato tra due diversi database. Il primo è il database totale, che non ha subito il processo di "pulitura" con il mach convettivo e successivamente di pesatura, mentre il secondo è quello ridotto con $W_H = 1$ e $W_L = 10$. Confrontando anche il coefficiente di regressione



(a) (b)

Figura 5.8: Regressione con database originale (a), e pesato (b)

$$R_a^2 = 0.846 \qquad R_b^2 = 0.940$$

si nota un miglioramento e cosa ancora più importante è che la riduzione del database comporta un minor costo computazionale.

Capitolo 6

Random Forest

Gli algoritmi di foreste casuali (Random Forest, RF) sono attualmente i più potenti tra le tecniche di Machine Learning. Una forma semplificata di questi è apparsa per la prima volta nel 1996 sviluppata da Leo Breiman, considerato l'inventore, con l'aiuto di Adele Cutler, e presero il nome di *algoritmi di bagging*. Nel corso degli anni sono stati migliorati fino ad arrivare alla versione definitiva di Random Forest. In generale, i due algoritmi funzionano come collezione di alberi decisionali, ognuno dei quali effettua una stima del valore da predire. Infine viene effettuata una media (in caso di regressione) o un sistema di voti maggioritari (problemi di classificazione) per ottenere la previsione finale. Prima di concentrarsi sugli algoritmi annunciati e le loro principali caratteristiche è utile descrivere brevemente il processo di costruzione di un albero decisionale e i suoi vantaggi.
Consideriamo un vettore contenente p variabili predittive (che potremmo indistintamente chiamare predictors da qui in poi) $X = (x_1, \ldots, x_p)^T$ e una variabile Y in cui ci sono tutti i valori di risposta. L'obiettivo è trovare una funzione di predizione $f(X)$ che riesce a predire i valori presenti in Y noto il set di variabili predittive X. La funzione di predizione è determinata attraverso l'utilizzo di una funzione di perdita $L(Y, f(X))$ già analizzata nello scorso capito, che ha lo scopo di minimizzare l'errore in previsione $E(L(Y, f(X)))$. Tipicamente la scelta della funzione di perdita ricade sull'errore quadratico per i problemi di regressione, così formulato

$$L(Y, f(X)) = (Y - f(X))^2$$

La funzione $f(X)$ è costruita come media tra le stime risultanti da ogni albero, considerando un numero totale di alberi pari a J, $h_1(X), \ldots, h_J(X)$, ed è formulata nel seguente modo:

$$f(x) = \frac{1}{J} \sum_{j=1}^{J} h_j(X)$$

Gli alberi decisionali campionano lo spazio delle variabili predittive utilizzando una sequenza di partizione binaria su ogni variabile. La radice dell'albero (punto più in alto) comprende l'intera variabile e da lì in poi ogni nodo si divide in altri due nodi in accordo con una delle variabili predittive. I nodi non più ramificabili vengono detti terminali. Per problemi di regressione è presente un particolare punto detto *split-point*: i valori minori di questo continuano sull'albero ramificato a sinistra, in caso contrario a destra. Un esempio è visualizzabile in figura 6.1, dove c è il valore dello split-point e X_i è la generica variabile predittiva. La divisione ad ogni nodo è scelta considerando tutte le possibili ramificazioni e selezionando la migliore secondo un criterio prestabilito. È proprio in questo punto che gli algoritmi di bagging e di random forest si differenziano.

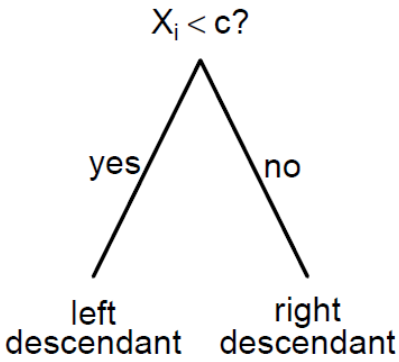


Figura 6.1: Schema di costruzione di un albero decisionale [12]

Per un problema di regressione il tipico criterio di splitting è l'errore quadratico medio al nodo:

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

dove $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ è il valore predetto al nodo, cioè la media dei valori di risposta.
Gli alberi decisionali godono di alcuni vantaggi che li rendono largamente utilizzati in algoritmi di questo tipo:

- non sono parametrici, possono modellare qualsiasi relazione di tipo input/output senza il bisogno di alcuna assunzione;
- implementano la selezione delle variabili importanti, evitando quelle ridondanti;
- sono facilmente interpretabili;
- sono molto robusti, anche di fronte a valori anomali.

6.1 Algoritmo di bagging

Come già anticipato, nel 1996 Breiman propone l'algoritmo di bagging: costruzione degli alberi decisionali indipendenti tra di loro. Ogni albero è sviluppato considerando un sottoinsieme dei dati di training disponibili e alla fine attraverso un sistema di media si ottiene la previsione desiderata. Ad esempio, ipotizziamo di avere un database di N variabili predittive. Prendiamo un sottoinsieme del database e costruiamo un albero decisionale, ogni divisione al nodo viene effettuata considerando la variabile migliore tra le N esistenti, quando l'albero non è più ramificabile otteniamo una previsione del dato di uscita. Ripetiamo questo procedimento un numero finito di volte, cambiando il sottoinsieme di dati di training e prendendolo in modo totalmente casuale. Tipicamente questo procedimento si stoppa una volta ottenuto un numero di alberi decisionali tra 200 e 500. Come già detto ogni albero fornisce una stima del valore di uscita, ma la previsione definitiva è ottenuta come media tra tutte le possibili stime ottenute.
Per facilità di osservazione si può prendere un esempio di classificazione in figura 6.2. I bootstrap indicano il sottoinsieme dei dati di training necessario per formare ogni albero. Se ad esempio ci sono 3 classi: blu, giallo, rosso e la costruzione di 5 alberi su un campione di input ha portato alla previsione giallo, giallo, rosso, giallo, giallo allora come valore di uscita di prende il giallo, il più diffuso tra i vari output.

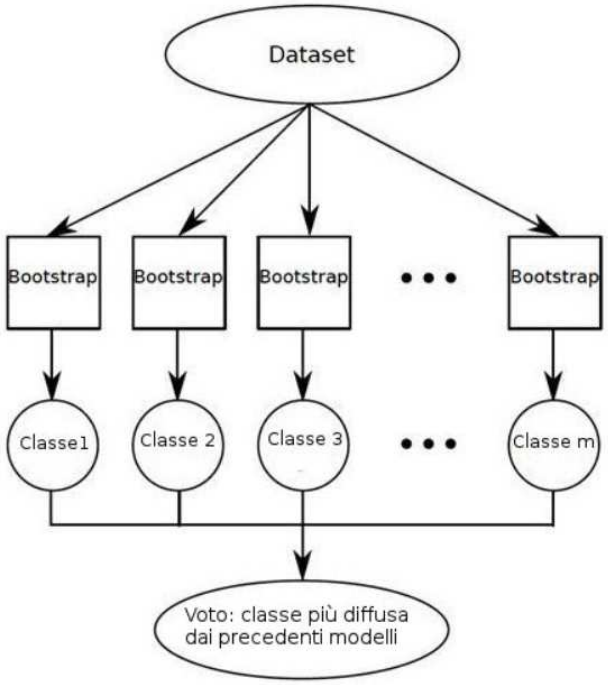


Figura 6.2: Esempio di algoritmo di bagging

Tuttavia, sebbene questo algoritmo sembri molto promettente, Breiman sperava in risultati migliori. Analizzando i vari algoritmi sviluppati notò che gli alberi risulanti sono tutti molto simili tra di loro e pensò che un rimedio efficace potesse essere quello di trovare un metodo per differenziarli quanto più possibile. L'idea di base di Breiman fu quella di introdurre dei fattori di casualità non solo nella scelta del sottoinsieme di dati di training da utilizzare nella costruzione dell'albero, ma anche all'interno dell'albero stesso, nella scelta delle variabili predittive da utilizzare per la divisione ad ogni nodo. Questa idea portò alla nascita nel 2001 degli algoritmi di Random Forest: si seleziona un nuovo sottoinsieme di variabili predittive ad ogni nodo dell'albero.

6.2 Random Forest

Random forest è un algoritmo di apprendimento supervisionato: esso considera numerosi alberi decisionali (una foresta di alberi) e li unisce per ottenere una predizione dei dati di uscita. Un grande vantaggio di questa tecnica è che può essere utilizzata sia per problemi di classificazione che di regressione. Come già anticipato, rispetto all'algoritmo di bagging che per effettuare lo split al nodo utilizza la variabile migliore tra tutte le variabili predittive presenti, quello di random forest seleziona un sottoinsieme casuale di queste e ne sceglie la migliore. Questo comporta una grande diversificazione tra i vari alberi che riesce a fornire un modello di previsione migliore e più affidabile. Le principali caratteristiche vantaggiose degli algoritmi di random forest si possono riassumere di seguito:

- hanno un'efficienza insuperabile tra gli algoritmi di machine learning attuali;
- funzionano molto bene con database di dimensione elevate;
- utilizzabili sia per problemi di regressione che di classificazione;

- implementano intrinsecamente un metodo per la selezione di variabili più importanti;
- forniscono una misura di prossimità tra le varie coppie di dati;
- non soffrono del problema di overfitting.

6.2.1 Algoritmo di random forest

Considerando un dataset di training D cosi formato

$$D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$$

dove $x_i = (x_{i,1}, \ldots, x_{i,p})^T$ sono le p variabili predittive e y_i è la risposta ad ogni x_i, l'algoritmo utilizza gli alberi decisionali $h_j(x, \Theta_j)$ come base per l'apprendimento. Una realizzazione particolare θ_j di Θ_j viene indicata nell'albero come $\hat{h}_j(x, \theta_j, D)$. Questa è la generica formulazione di Breiman anche se θ_j non appare in modo esplicito ma si manifesta in due modi:

1. ogni albero è costruito considerando un sottoinsieme random del dataset originale;
2. al momento della divisione al nodo vengono selezionate solamente m variabili predittive tra le p disponibili ($m < p$).

A livello matematico l'algoritmo ([12]) è formulato nel modo seguente:

sia $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ il dataset completo di training, con $x_i = (x_{i,1}, \ldots, x_{i,p})^T$.

Per j=1 fino a j=J:

1. prendi un sottoinsieme di dati D_j di dimensione N, dal dataset originale D.
2. usando l'insieme D_j come nuovo dataset di training, costruisci un albero con una partizione binaria al nodo, come segue:
   a. inizia con tutte le osservazioni in un singolo nodo.
   b. Ripeti i seguenti step in maniera ricorsiva per ogni divisione al nodo fino a quando non si raggiunge il criterio di stop:
      i. seleziona m variabili predittive tra le p disponibili.
      ii. trova la migliore per eseguire la divisione al nodo.
      iii. dividi il nodo in due nodi discendenti usando lo "split-point" scelto al punto ii.

Per effettuare la predizione in nuovo punto di x si calcola:

- $\hat{f}(x) = \frac{1}{J} \sum_{j=1}^{J} \hat{h}_j(x)$ per regressione
- $\hat{f}(x) = argmax_y \sum_{j=1}^{J} I(\hat{h}_j(x) = y)$ per classificazione

dove $\hat{h}_j(x)$ è la predizione della variabile in x usando il j-esimo albero.

6.2.2 Dati di Out-Of-Bag

Quando si costruisce un sottoinsieme di training per lo sviluppo dell'albero, come illustrato al punto 1. dell'algoritmo precedente, necessariamente una parte dei dati vengono scartati e questi rientrano a far parte dell'insieme chiamato Out Of Bag (OOB). Ovviamente ogni albero possiede un diverso insieme di OOB perché la scelta del sottoinsieme di training è totalmente casuale. Questi dati "scartati" sono molto utili sia per il calcolo dell'errore in previsione che per la stima delle variabili più importanti.
Per quanto riguarda il calcolo dell'errore, si segue l'algoritmo proposto da A. Cutler, D. R. Cutler e J.R. Stevens ([12]):

Sia D_j il j-esimo sottoinsieme di training e $\hat{h}_j(x)$ la predizione del j-esimo albero per $j = 1, \ldots, J$.
Per $i = 1, \ldots, N$:

1. sia $G_i = \{j : (x_i, y_i) \notin D_j\}$ l'insieme di dati OOB e sia J_i il numero di elementi in G_i (algoritmo di Random Forest).
2. si definisce la predizione out-of-bag ad x_i:
   - $\hat{f}_{oob} = \frac{1}{J_i} \sum_{j \in G_i} \hat{h}_j(x_i)$ per regressione
   - $\hat{f}_{oob} = arg\max_y \sum_{j \in G_i} I(\hat{h}_j(x_i) = y)$ per classificazione

   dove $\hat{h}_j(x_i)$ è la predizione in x_i usando l'albero j-esimo.

In particolare, per problemi di regressione con funzione di perdita misurata con l'errore quadratico, l'errore generalizzato è tipicamente stimato con l'errore quadratico medio dei dati di out-of-bag:

$$\text{MSE}_{oob} = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \hat{f}_{oob}(x_i) \right)^2$$

dove $\hat{f}_{oob}(x_i)$ è la predizione out-of-bag per l'osservazione i.

6.2.3 Importanza delle variabili

La stima dell'importanza delle variabili non è un problema banale, perché spesso una variabile è importante nella sua interazione con le altre. L'algoritmo di Random Forest implementa automaticamente questo aspetto analizzando l'andamento dell'errore in predizione quando viene apportata una modifica ad una variabile lasciando invariate tutte le altre. In generale per misurare l'importanza della variabile k si prendono i valori OOB e si analizzano attraverso un albero, in seguito si modificano i valori corrispondenti alla variabile k e si lasciano immutati tutti gli altri. Quando si modificano i valori c'è la nascita di un nuovo albero e quindi si ottengono due insiemi diversi di predizione. Questi possono essere confrontati e con il calcolo dell'errore quadratico medio si può stimare l'importanza della variabile. A livello matematico si può seguire l'algoritmo presente in ([12]):

Per trovare l'importanza della variabile k, per $k = 1, \ldots, p$:

1. per $i = 1, \ldots, N$:
   a. sia $G_i = \{j : (x_i, y_i) \notin D_j\}$ l'insieme di dati OOB e sia J_i il numero di elementi in G_i

b. sia $\hat{y}_{i,j} = \hat{h}_j(x_i)$ per tutti $j \in G_i$

2. Per $j = 1, \ldots, J$

   a. Sia D_j il j-esimo sottoinsieme di training

   b. sia $F_j = \{i : (x_i, y_i) \notin D_j\}$

   c. si permuti casualmente il valore della variabile k per i punti $\{x_i : i \in F_i\}$ per dare $P_i = \{x_i^* : i \in F_i\}$

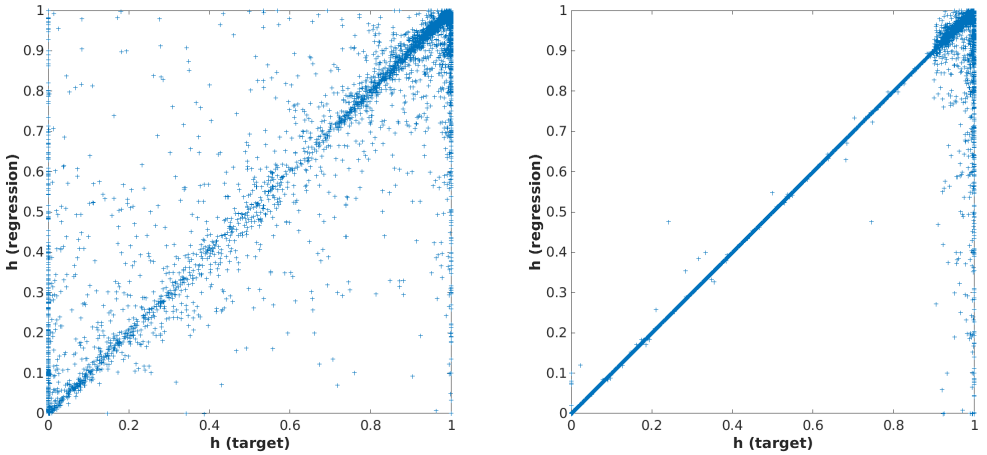
   d. sia $\hat{y}_{i,j}^* = \hat{h}_j(x_i^*)$ per ogni $i \in F_j$

3. per $i = 1, \ldots, N$

   - $\text{Imp}_i = \frac{1}{J_i} \sum_{j \in G_i} (y_i - \hat{y}_{i,j}^*)^2 - \frac{1}{J_i} \sum_{j \in G_i} (y_i - \hat{y}_{i,j})^2$ per regressione
   - $\text{Imp}_i = \frac{1}{J_i} \sum_{j \in G_i} (y_i \neq \hat{y}_{i,j}^*) - \frac{1}{J_i} \sum_{j \in G_i} (y_i \neq \hat{y}_{i,j})$ per classificazione

6.3 Risultati

Per la tecnica di Random Forest si è considerato lo stesso database utilizzato per le reti neurali, presentato in 3.2 eriassunto nel paragrafo 5.5. Anche in questo caso è utile fare un confronto tra i due diversi database, quello originale e la sua versione ridotta e pesata. Graficamente, come nel caso di reti neurali, si può analizzare la regressione effettuata per entrambi i database.



(a) (b)

Figura 6.3: Grafico di regressione con RF per database completo (a) e ridotto (b)

Anche in questo caso si possono confrontare numericamente i valori del coefficiente di regressione

$$R_a^2 = 0.926 \qquad R_b^2 = 0.995$$

Immediatamente si nota che la regressione effettuata con la tecnica di random forest risulta molto più promettente delle precedenti due analizzate, la regressione sembra essere quasi perfetta ad eccezione della zona superiore quando i valori del target sono molto prossimi all'unità.

Capitolo 7

Analisi CFD e risultati

In questo capitolo verranno effettuate delle simulazioni CFD sui risultati ottenuti dalle tecniche di apprendimento Reti Neurali e Random Forest, per le stesse condizioni operative allo scopo di capire se effettivamente la soluzione si avvicina a quella ottenuta dal metodo di inversione del campo. Come più volte ripetuto, questo aspetto è molto importante perché in caso affermativo si riuscirebbe ad ottenere una forma analitica del fattore di correzione in funzione delle variabili fisiche, permettendo una grande generalizzazione del modello a molteplici casi di studio. Si precisa che sebbene i diagrammi di regressione illustrati nella sezione risultati dei capitoli 5 e 6 siano molto promettenti, il coefficiente di regressione è sempre minore dell'unità e per tale motivo una volta effettuata la simulazione CFD con soluzione ottimale quella proveniente dal metodo dell'aggiunto, necessariamente questa evolverà verso una nuova soluzione stazionaria che si spera essere quanto più vicina possibile alla prima. Per quanto detto, in una prima parte verranno analizzate le visualizzazioni ottenute con l'analisi CFD e successivamente si porrà attenzione alla previsione considerando lo stesso caso di studio con differenti numeri di Reynolds per vedere quanto la previsione fornita dalle tecniche di machine learning si avvicini ai dati sperimentali disponibili.

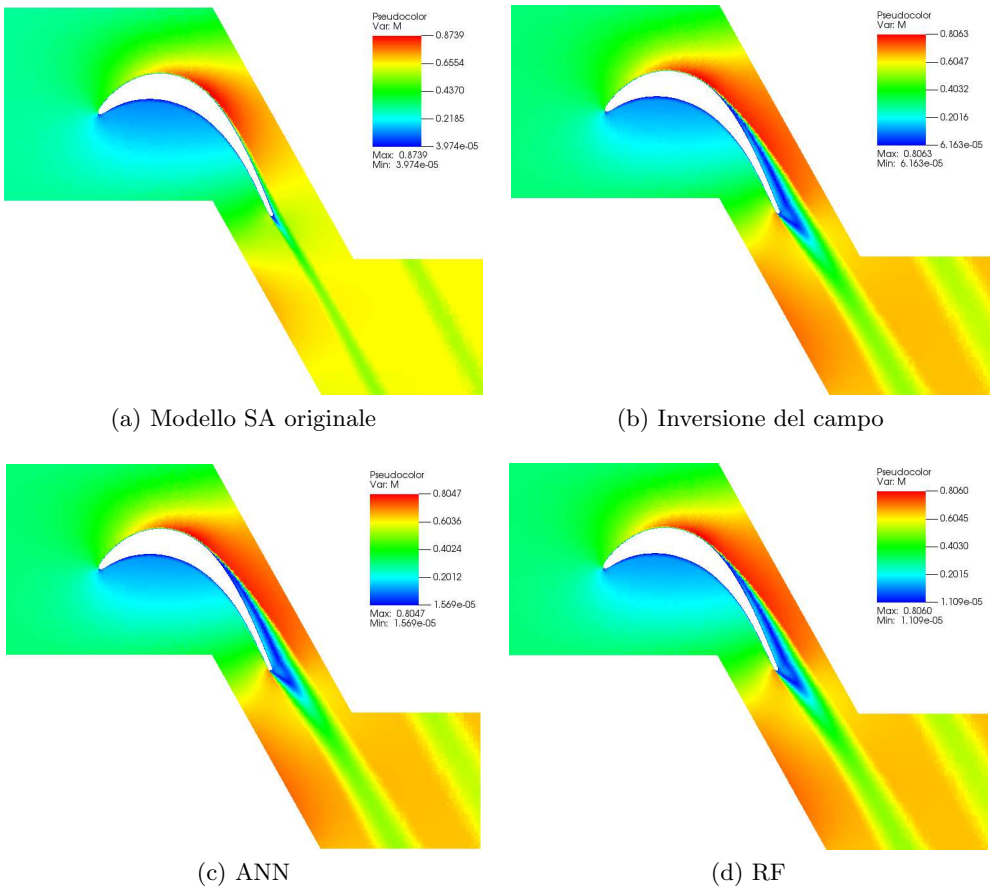
7.1 Analisi CFD

Le simulazioni CFD sono state svolte per i due numeri di Reynolds delle condizioni operative: $Re_{2_s} = (80000, 250000)$. Come più volte anticipato, all'interno di questo range sul dorso del profilo si passa da una separazione aperta ad una chiusa. I confronti verranno effettuati con due grandezze: la distribuzione del numero di Mach sulla paletta e il fattore di correzione.

7.1.1 Numero di Reynolds 80000

In figura 7.1 notiamo un confronto tra il modello di turbolenza Spalart-Allmaras originale, la sua versione modificata con il metodo dell'inversione del campo e la previsione effettuata dalle tecniche di machine learning proposte.

Si nota immediatamente la miglioria apportata dal metodo di inversione del campo al modello di turbolenza originale: la separazione è predetta con maggiore precisione. Il risultato fornito dalla rete neurale e da random forest è molto simile e conferma la giusta previsione nei diagrammi di regressione.



(a) Modello SA originale (b) Inversione del campo

(c) ANN (d) RF

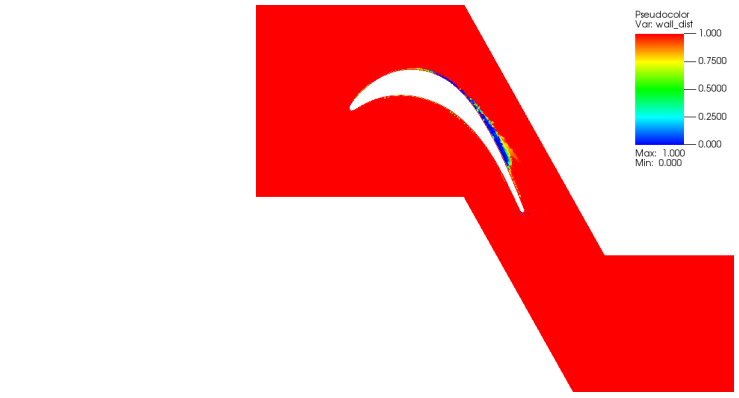
Figura 7.1: Distribuzione del numero di Mach con Reynolds 80000 nei modelli SA originale (a), Inversione del campo (b), reti neurali (c), random forest (d)

Una visualizzazione analoga si può fare considerando il fattore di correzione in figura 7.2. La principale variazione del fattore di correzione avviene nello strato limite, in cui assume valori molto bassi, rispetto al restante campo di moto in cui è uniformemente uguale a 1. La tecnica di random forest riesce a predire in modo migliore la variazione del fattore di correzione rispetto alle reti neurali. Infatti, nei diagrammi di regressione si era notato la difficoltà della tecnica di Random Forest di predire valori del fattore prossimi all'unità, mentre per valori molto bassi la retta era molto preciso. Nelle reti neurali, invece, alcune dispersioni erano presenti anche per valori più basssi.

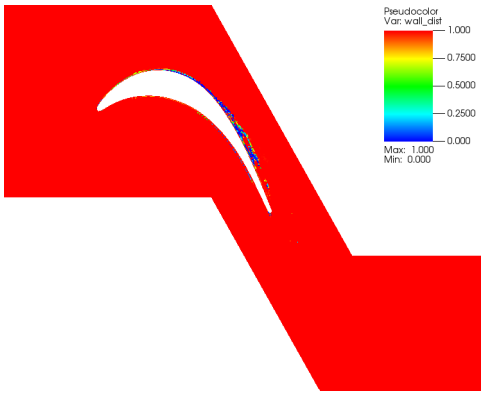
7.1.2 Numero di Reynolds 250000

Per numero di Reynolds isoentropico di uscita pari a $2.5 \cdot 10^5$ ci si aspetta una separazione chiusa seguita da un flusso riattaccato. Anche in questo visualizziamo il campo di moto per distribuzione di numero di mach (in figura 7.3) e fattore di correzione (in figura 7.4).

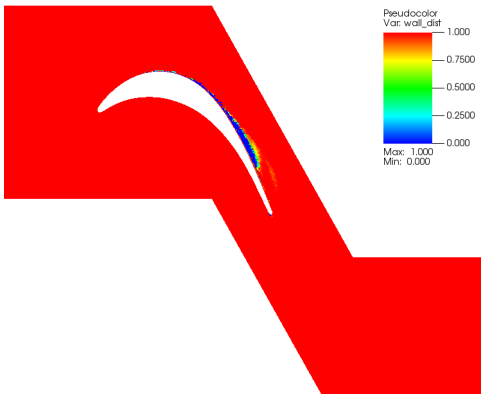
Questa separazione, totalmente assente nel modello SA originale, è invece predetta dagli altri tre modelli piuttosto bene. Considerazioni analoghe possono essere fatte sul fattore di correzione in cui si nota come la tecnica di random forest performi leggermente meglio rispetto alla rete neurale.



(a) Inversione del campo



(b) ANN

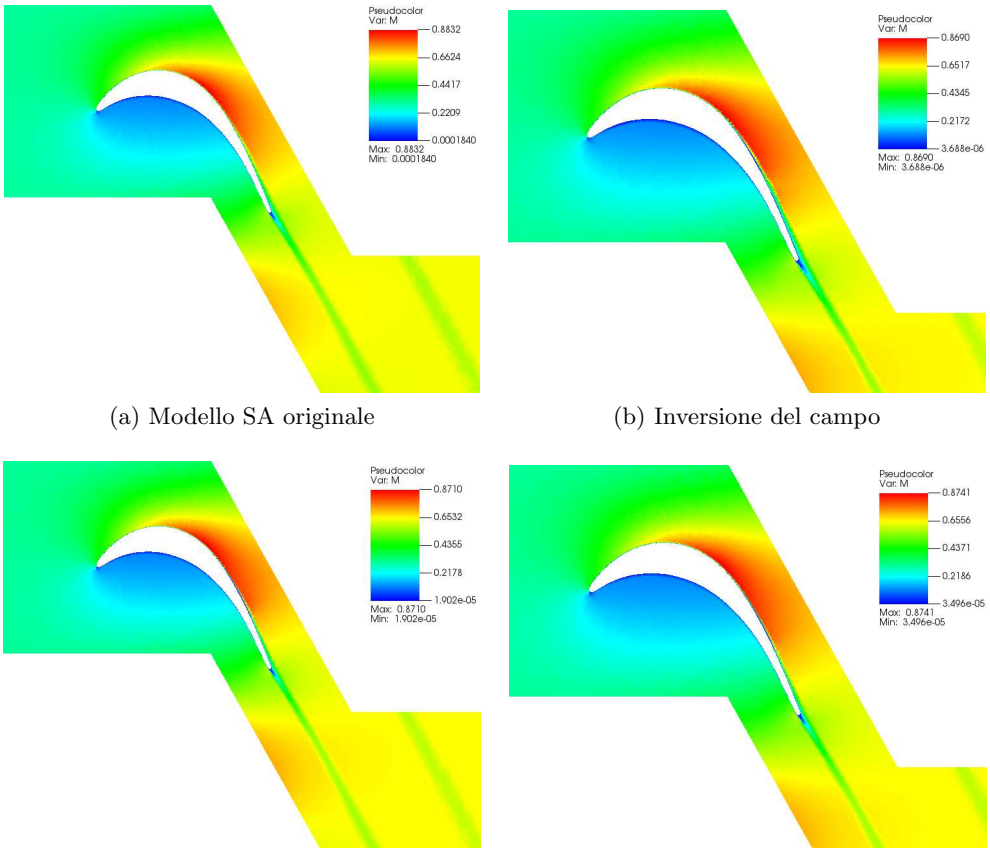


(c) RF

Figura 7.2: Distribuzione del fattore di correzione $h(\beta)$ con Reynolds 80000 nei modelli di Inversione del campo (a), reti neurali (b), random forest (c)

7.2 Previsioni

In questa sezione si vuole valutare se il modello costruito con le tecniche di machine learning è valido anche per casi di studio che non sono stati addestrati. In particolare, gli algoritmi di reti neurali e random forest hanno usato i database costruiti a Re_{2_s} per l'apprendimento, ora si vuole valutare come agisce il modello se applicato ad altri due numeri di Reynolds $(1.2 \cdot 10^5, 1.6 \cdot 10^5)$ e confrontarli con i dati sperimentali già esistenti. Questi confronti verranno effettuati per le perdite cinetiche e per l'angolo di uscita β_2 in figura 7.5 rispettivamente (a) e (b). Nei diagrammi sono presentati diversi confronti. Di particolare importanza sono i risultati sperimentali ottenuti al VKI e i tre modelli di turbolenza: quello Sapalart-Allmaras originale (SA orig.) e la sua implementazione con il metodo di reti neurali (SA-ANN) e random forest (SA-RF). A titolo di confronto sono anche presenti i risultati numerici ottenuti da Benyahia et al. (2011), Pacciati et al. (2010) e Babajee (2013).
Nelle condizioni operative di training (Reynolds pari a $8 \cdot 10^4$ e $2.5 \cdot 10^5$) si nota che il modello di reti neurali e random forest è molto prossimo ai risultati sperimentali e fornisce un risultato migliore rispetto al modello Spalart-Allmaras originale; tuttavia a diversi numeri di Reynolds i risultati non sono ottimi, specialmente per bassi valori. Si nota anche che le due tecniche di machine learning forniscono differenti previsioni sebbene siano state allenate sullo stesso database di training, a conferma del fatto che le diverse tecniche possono portare a diversi risultati.
È importante notare che le previsioni sono state effettuate in funzione delle perdite di



(a) Modello SA originale

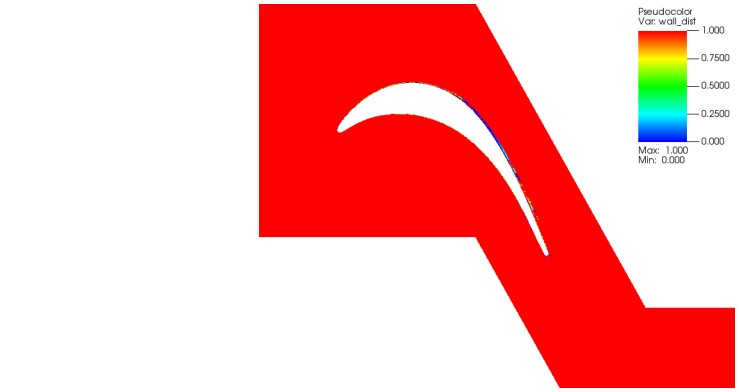
(b) Inversione del campo

(c) ANN

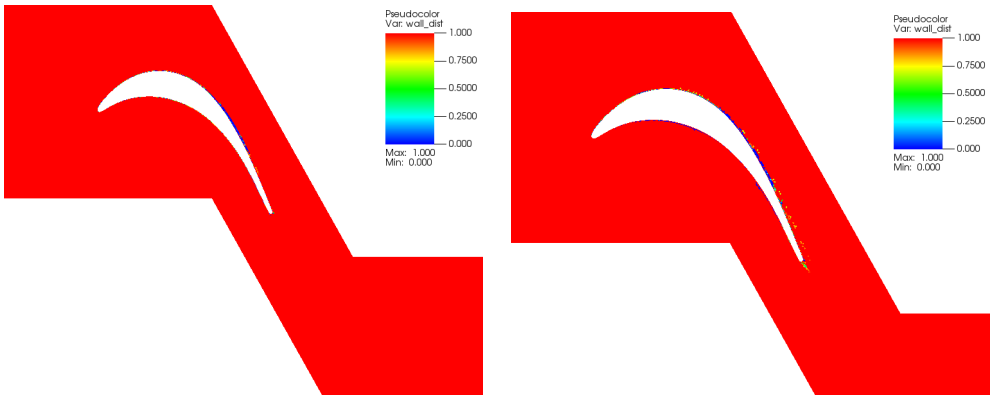
(d) RF

Figura 7.3: Distribuzione del numero di Mach con Reynolds 250000 nei modelli SA originale (a), Inversione del campo (b), reti neurali (c), random forest (d)

cinetiche e dell'angolo di uscita, e mostrano risultati piuttosto soddisfacenti anche se la funzione goal per l'approccio di inversione del campo è basata sulla distribuzione del numero di Mach isoentropico a parete.

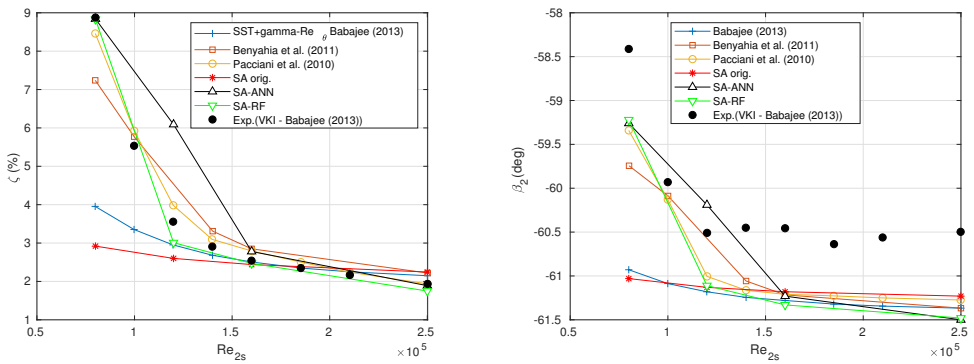


(a) Inversione del campo



(b) ANN (c) RF

Figura 7.4: Distribuzione del fattore di correzione con Reynolds 250000 nei modelli SA originale (a), Inversione del campo (b), reti neurali (c), random forest (d)



(a) Perdite cinetiche (b) Angolo di uscita

Figura 7.5: Perdite cinetiche (a) e angolo di uscita (b) per diversi valori di numeri di Reynolds

Capitolo 8

Conclusioni

In questo lavoro di tesi si è proposto il paradigma di field inversion and machine learning (FIML), con applicazione ad una schiera di pale di una turbina a bassa pressione T106c. L'approccio di inversione del campo e il metodo dell'aggiunto sono stati applicati al set di equazioni di Navier-Stokes mediate alla Reynolds (RANS) con modello di chiusura ad una equazione proposto da Spalart-Allmaras (SA). Il modello SA tende a sovrastimare l'aumento di viscosità turbolenta in questi tipi di flusso e per tale motivo è un buon punto di partenza per capire quanto l'approccio dell'inversione del campo possa migliorare la previsione. Un fattore di correzione è stato inserito all'interno del modello di turbolenza, come moltiplicatore del termine di produzione, e da questo è stato ricavato un campo di correzione che agisce come funzione di intermittenza per simulare la presenza di transizioni e separazioni. Come funzione goal è stata scelta la distribuzione del numero di Mach isoentropico per il quale sono disponibili dei risultati sperimentali in letteratura. Il campo di moto è stato analizzato a due differenti numeri di Reynolds $Re_{2_s} = (8 \cdot 10^4; 2.5 \cdot 10^5)$: all'interno di questo range sulla paletta si manifesta una transizione da una separazione aperta ad una chiusa con flusso riattaccato. Abbiamo proposto un metodo per la riduzione del database originale, basato su un sensore che misura la presenza di scorrimenti: il mach convettivo. Attraverso questo sensore si sono potute considerare le regioni di strato limite e di scia in cui è interessante analizzare gli effetti viscosi e le variazioni nel fattore di correzione, mentre al di fuori di queste zone il fattore di correzione assume valore costantemente pari ad uno. Nonostante ciò il databse risultava ancora pieno di valori $\beta \simeq 1$, non rendendo possibile una previsione ottimale, e un ulteriore metodo di scrematura del database è stato proposto: all'interno delle zone viscose identificate dal numero di Mach convettivo tutti i punti con $\beta > 0.9$ sono stati presi una volta sola, i restanti 10 volte. Questo ha permesso di ripopolare il database con moltissimi punti aventi un basso fattore di correzione. A questo punto tre diverse tecniche di Machine Learning sono state analizzate al fine di trovare una correlazione tra il fattore di correzione e alcune grandezze fisiche caratteristiche del flusso. La tecnica basata su programmazione genetica ha un grande potenziale, ed è in grande sviluppo ma non ha portato nessuna miglioria al modello, predicendo un coefficiente di regressione troppo basso, e pertanto si è preferito non analizzarla con le simulazioni CFD. Le altre due, reti neurali artificiali e random forest, si sono mostrate molto promettenti, i risultati ottenuti sono in accordo con l'approccio di inversione del campo, e dalle analisi CFD svolte si possono visualizzare graficamente le separazioni aperte e chiuse accennate, completamente mancate dal modello SA originale. Al fine di generalizzare i risultati è stata proposta una previsione a diversi numeri di Reynolds, non utilizzati nell'addestramento delle tecniche di machine learning e in funzione di due grandezze (perdite cinetiche e angolo di uscita nella scia) che non sono state considerate nella funzione obiettivo. I risultati si mostrano in accordo con

i dati sperimentali disponibili soprattutto nelle zone in cui le tecniche sono state allenate, altrove sono evidenti alcune differenze anche se è bene notare che appaiono piuttosto soddisfacenti rispetto ad un modello più complesso a quattro equazioni (SST-γ-Re_θ).
Quello proposto è un punto di partenza nella ricerca di un legame tra il campo di correzione e alcune grandezze fisiche, e molti studi dovranno ancora essere effettuati su una ottimale scelta del set di input iniziali, per evitare ridondanza e disturbi nei dati, e sulla ricerca di una architettura ideale per le tre tecniche proposte. Infine, si sottolinea che, sebbene sia stata trovata una relazione analitica che mostra ottimi risultati in addestramento e piuttosto soddisfacenti in previsione, non si assicura l'esistenza di un reale nesso causa-effetto. La presenza di un elemento indiziario, se correlato ad altri indizi, può portare alla deduzione con ragionevole probabilità dell'esistenza di una causa, ma non è esso stesso la causa.
Sarebbe interessante, inoltre, una volta ottenuta la previsione con le tecniche di machine learning, sapere qual è l'incertezza corrispondente. Questo risulta abbastanza semplice con la tecnica di Random Forest: essa è costituita da una foresta di alberi, ciascun albero è di tipo decisionale che effettua una serie di condizioni a cascata sugli input e predice l'output. Mettendo insieme tutti gli alberi nella foresta, il valore predetto finale viene calcolato come media tra tutti i valori dei singoli alberi. Se piuttosto che effettuare la media si calcola la deviazione standard si ottiene l'incertezza sulla previsione. Se gli alberi sono tutti in accordo tra di loro questa incertezza è molto piccola. Numerosi sono i futuri sviluppi associabili alle tecniche analizzate, al fine di migliorare i modelli di turbolenza.

Bibliografia

[1] C.C. Aggarwal et al. *Neural networks and deep learning*. Springer, 2018.

[2] S.R. Allmaras and F.T. Johnson. Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, pages 1–11, 2012.

[3] Jayson Babajee. *Detailed numerical characterization of the separation-induced transition, including bursting, in a low-pressure turbine environment*. PhD thesis, 2013.

[4] A. Benyahia, L. Castillon, and R. Houdeville. Prediction of separation-induced transition on high lift low pressure turbine blade. In *Turbo Expo: Power for Land, Sea, and Air*, volume 54655, pages 1835–1846, 2011.

[5] R.A. Berk. *Statistical learning from a regression perspective*, volume 14. Springer, 2008.

[6] J. Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[7] N. Boldrini. Intelligenza artificiale (ai): cos'è, come funziona e applicazioni. 2020. https://www.ai4business.it/intelligenza-artificiale/intelligenza-artificiale-cose/#Machine_Learning_e_Deep_Learning_un_po_di_chiarezza.

[8] L. Breiman and A. Cutler. Random forest for beginners. *Salford Systems*, 2014.

[9] E.A. Colbourn, S.J. Roskilly, R.C. Rowe, and P. York. Modelling formulations using gene expression programming–a comparative analysis with artificial neural networks. *European journal of pharmaceutical sciences*, 44(3):366–374, 2011.

[10] N.L. Cramer. A representation for the adaptive generation of simple sequential programs. In *proceedings of an International Conference on Genetic Algorithms and the Applications*, pages 183–187, 1985.

[11] E.M. Curtis, H.P. Hodson, M.R. Banieghbal, J.D. Denton, R.J. Howell, and N.W. Harvey. Development of blade profiles for low-pressure turbine applications. *ASME J. Turbomach.*, pages 531,538, 1997.

[12] A. Cutler, D.R. Cutler, and J.R. Stevens. Random forests. In *Ensemble machine learning*, pages 157–175. Springer, 2012.

[13] C. Ferreira. Gene expression programming: a new adaptive algorithm for solving problems. *arXiv preprint cs/0102027*, 2001.

[14] C. Ferreira. *Gene expression programming: mathematical modeling by an artificial intelligence*, volume 21. Springer, 2006.

[15] A. Ferrero. *Computational fluid dynamics for aerospace propulsion systems: an approach based on discontinuous finite elements*. PhD thesis, 2014.

[16] A. Ferrero, A. Iollo, and F. Larocca. Rans closure approximation by artificial-neural networks. In *ETC 2019-13th European Turbomachinery Conference on Turbomachinery Fluid Dynamics and Thermodynamics*, 2019.

[17] A. Ferrero, A. Iollo, and F. Larocca. Field inversion for data-augmented rans modelling in turbomachinery flows. *Computers & Fluids*, 201:104474, 2020.

[18] A. Ferrero, A. Iollo, F. Larocca, M. Loffredo, and E. Menegatti. Field inversion and machine learning strategies for improving rans modelling in turbomachinery. In *ETC 2021-14th European Turbomachinery Conference on Turbomachinery Fluid Dynamics and Thermodynamics*, 2021.

[19] L. Gilles. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

[20] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[21] J. Holland. Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence*, 1975.

[22] J.R. Holland. *Integrated Field Inversion and Machine Learning With Embedded Neural Network Training for Turbulence Modeling*. PhD thesis, 2019.

[23] J. Hourmouziadis. Aerodynamic design of low pressure turbines. *AGARD, Blading Design for Axial Turbomachines 40 p(SEE N 89-27661 22-07)*, 1989.

[24] G. Iposi. Miglioramento di modelli di turbolenza rans per flussi interni.

[25] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[26] A. Liaw, M. Wiener, et al. Classification and regression by random forest. *R news*, 2(3):18–22, 2002.

[27] S. Luo, J. Cui, M. Vellakal, J. Liu, E. Jiang, S. Koric, and V. Kindratenko. Review and examination of input feature preparation methods and machine learning models for turbulence modeling. *arXiv preprint arXiv:2001.05485*, 2020.

[28] J. Michálek, M. Monaldi, and T. Arts. Aerodynamic performance of a very high lift low pressure turbine airfoil (t106c) at low reynolds and high mach number with effect of free stream turbulence intensity. *Journal of Turbomachinery*, 134(6), 2012.

[29] M. Mohebujjaman, Rebholz L.G., and T. Iliescu. Physically-constrained data-driven correction for reduced order modeling of fluid flows. *International Journal for Numerical Methods in Fluids*, 09 2018.

[30] R. Pacciani, M. Marconcini, A. Arnone, and F. Bertini. A cfd study of low reynolds number flow in high lift cascades. In *Turbo Expo: Power for Land, Sea, and Air*, volume 44021, pages 1525–1534, 2010.

[31] R. Paciorri and F. Sabetta. Compressibility correction for the spalart-allmaras model in free-shear flows. *Journal of Spacecraft and Rockets*, 40(3):326–331, 2003.

[32] E. J. Parish and K. Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305:758–774, 2016.

[33] S.B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[34] D. R. Sandberg and V. Michelassi. The current state of high-fidelity simulations for main gas path turbomachinery components and their industrial impact. *Flow, Turbulence and Combustion*, 102(4):797–848, 2019.

[35] A. P. Singh, S. Medida, and K. Duraisamy. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA journal*, 55(7):2215–2227, 2017.

[36] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows,? la recherche aerospatiale, no. 1. 1994.

[37] P.R. Spalart, S. Deck, M.L. Shur, K.D. Squires, M.Kh Strelets, and A. Travin. A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and computational fluid dynamics*, 20(3):181, 2006.

[38] J. X. Wang, J.L. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3):034603, 2017.

[39] M. Yang and Z. Xiao. Improving the k–ω–γ–ar transition model by the field inversion and machine learning framework. *Physics of Fluids*, 32(6):064101, 2020.