UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Corso di Laurea triennale

RETI NEURALI E TECNICHE DI DATA DRIVEN PER LA CLASSIFICAZIONE DEI GUASTI

NEURAL NETWORKS AND DATA DRIVEN TECHNIQUES FOR FAULTS' CLASSIFICATIONS

Relatore: Chiar.mo
Prof. Andrea Bonci

Tesi di Laurea di:
Christian Iezzi

A.A. 2019 / 2020

Capitolo 1

INTRODUZIONE

Con l'avvento dell'Industry 4.0 uno dei settori su cui si sta investendo e ricercando maggiormente è quello della manutenzione, ossia tutti quegli interventi che hanno lo scopo di risolvere guasti delle macchine, in particolare uno degli approcci più moderni delle problematiche manutentive è quello della manutenzione predittiva, che prevede la programmazione in tempo reale degli interventi in funzione delle condizioni della macchina e dei requisiti da rispettare per contribuire a migliorare l'affidabilità del sistema. La considerazione che sta alla base dell'adozione di questo approccio è che un guasto non si manifesta all'improvviso, ma, nella maggioranza dei casi, costituisce solo il punto di arrivo di un deterioramento progressivo. La manutenzione predittiva utilizza tecniche basate sul Condition Monitoring, ovvero il processo di monitoraggio di diversi parametri nei macchinari, come ad esempio vibrazione o temperatura, al fine di identificare i cambiamenti che possono essere indicativi di un guasto in via di sviluppo. Il Condition Monitoring permette di mantenere sotto controllo lo stato del sistema attraverso analisi diagnostiche o, dove possibile, tramite ispezioni visive, riducendo il rischio di avere lunghi tempi di fermata per guasti accidentali. L'utilizzo di queste tecniche consente inoltre di progettare la manutenzione secondo le reali esigenze del sistema ed eseguire le operazioni nei momenti opportuni prevenendo, cosi, danni consequenziali e permettendo di conservare il bene materiale e di mantenerlo nel suo funzionamento ottimale. Quindi la manutenzione predittiva ha anche il vantaggio di poter prevedere il probabile andamento del guasto, in modo tale da affrontare i malfunzionamenti prima che diventino gravi. Pertanto, i guadagni che si possono ottenere dalla manutenzione predittiva sono molteplici:

- Si evitano guasti improvvisi dovuti a componenti usurati e i costi che derivano dalle le conseguenti manutenzioni;
- Si massimizzano i tempi in cui le macchine sono in funzionamento, aumentando la produzione e l'efficienza;
- Si risparmia sui componenti, sfruttando ciascuno di essi il più possibile, sostituendoli solamente quando ormai la loro vita utile residua è prossima allo zero;
- Si riduce l'occupazione del magazzino con parti di ricambio, che possono essere ordinate quando il livello di degrado di macchine o componenti supera una certa soglia.

Obiettivo e Contributo

L'obiettivo del progetto è quello di dimostrare la validità delle reti come classificatori e come mezzo utile per l'apprendimento supervisionato. Nello specifico il contributo di questa tesi è quello di realizzare una rete neurale in grado di riconoscere e classificare i guasti più comuni sugli azionamenti di movimento elettrico dei robot cartesiani (CR). Il sistema in cui è stata realizzata la rete neurale è stato già studiato e analizzato, quindi il progetto di questa tesi è un continuo di un lavoro svolto in precedenza dove sono stati organizzati e analizzati i dati e in cui si sono anche esaminati i tipi di guasti dal punto di vista fisico e meccanico. Il sistema su cui si è lavorato è applicato a un CR in cui la coppia cinghia-motore costituisce il sistema di trasmissione più comune, dove, solitamente, il sistema di trasmissione a cinghia è uno dei componenti più critici e la parte più sensibile al degrado. L'idea di base è quella di sviluppare un modo intelligente per capire quando il sistema sta passando da una situazione sicura a una critica. La soluzione si basa su dati relativi al tensionamento della cinghia, infatti conoscendo i valori corretti di questi ultimi siamo in grado di individuare diversi stati della macchina. La rete neurale proposta è in grado di rilevare lo stato in cui si trova il sistema richiedendo in fase di training i vari dati che sono stato elaborati e raccolti in precedenza durante il funzionamento della macchina nei diversi stati e permette di capire approssimativamente il valore del tensionamento della cinghia, dato che la rete si è dimostrata anche in grado di distinguere, per una stessa condizione, diversi livelli di stato di avanzamento, aumentando il grado e l'utilità delle informazioni utilizzabili in fase di pianificazione degli interventi di manutenzione.

Struttura dell'Elaborato

Il resto dell'elaborato è strutturato nel modo seguente:

Nel secondo capitolo vengono definiti i concetti di machine learning con relative tecniche per lo sviluppo dei modelli e per la raccolta dei dati. Inoltre, viene anche spiegato il funzionamento dei modelli basati sui dati e delle reti neurali.

Nel terzo capitolo è analizzato e spiegato in modo dettagliato il sistema sul quale si è lavorato e il suo funzionamento.

Nel quarto capitolo viene esposta e commentata la rete neurale che è stata creata per la diagnosi e la classificazione dei guasti e vengono riportano i risultati ottenuti.

Nel quinto capitolo si illustrano in modo approfondito tecniche di data driven per la classificazione dei guasti confrontate con la rete neurale sviluppata.

Capitolo 2

2.1 MACHINE LEARNING

Il machine learning o apprendimento automatico è lo studio e la realizzazione di algoritmi e modelli che permettono a sistemi di elaborazione di svolgere determinati compiti senza essere esplicitamente programmati per farlo, ma apprendendo tramite il riconoscimento di pattern. Tipicamente gli algoritmi di learning non usano come input direttamente il dato grezzo, ma dei valori derivati da essi, detti features che fungono da prima fase di elaborazione del dato, infatti ogni algoritmo di apprendimento supervisionato ha bisogno di un set di dati di training. L'estrazione delle features serve per facilitare il compito del modello, riducendo il numero delle variabili di input ma preservandone le informazioni e le caratteristiche, ma anche per favorire l'interpretabilità. Il machine learning permette quindi alle macchine di imparare dall'esperienza, c'è apprendimento quando le prestazioni del programma migliorano dopo lo svolgimento di un compito o il completamento di un'azione. Quindi invece di scrivere il codice di programmazione, al programma vengono forniti dei set di dati che vengono elaborati attraverso degli algoritmi permettendo al programma di ricavare modelli di apprendimento, questi ultimi permettono di costruire algoritmi per svolgere la funzione, l'attività o il compito richiesto. Ci sono diversi modelli che permettono di costruire algoritmi di apprendimento per risolvere i problemi:

• APPRENDIMENTO SUPERVISIONATO (SUPERVISED LEARNING)

Si parla di apprendimento supervisionato quando si hanno dati di input e dati di output e si utilizza un algoritmo che apprende la funzione che dall'input genera l'output. L'obiettivo è approssimare la funzione in modo che quando si ha un nuovo dato di input l'algoritmo può prevedere il valore di output generato per quel dato. L'apprendimento si interrompe quando l'algoritmo raggiunge un livello accettabile di prestazioni. Un apprendimento di questo tipo viene utilizzato per risolvere problemi di classificazione, ossia problemi in cui la macchina deve riconoscere e categorizzare oggetti da un set di dati, oppure problemi di regressione dove la macchina può predire il valore di ciò che sta analizzando in base ai dati, quindi studia la relazione tra più variabili indipendenti fra loro.

• APPRENDIMENTO NON SUPERVISIONATO (UNSEPERVISED LEARNING)

Si parla di apprendimento non supervisionato quando si ha una variabile di input, rappresentata dai dati, e nessuna variabile di output corrispondente. Si pone l'obiettivo di trovare relazioni tra i vari dati che vengono analizzati senza utilizzare una

categorizzazione. Un apprendimento di questo tipo viene utilizzato per problemi di raggruppamento, quindi quando è necessario raggruppare i dati che presentano caratteristiche simili. In questo caso l'algoritmo impara quando individua una relazione tra i dati, estraendo una regola che raggruppa i casi presentati secondo caratteristiche che ricava dai dati stessi. Inoltre, è usato anche per risolvere problemi di associazione**,** dove si vogliono scoprire regole che descrivono grandi porzioni di dati, in questo caso ci si pone quindi l'obiettivo di trovare schemi ricorrenti e associazioni tra i dati.

• APPRENDIMENTO PARZIALMENTE SUPERVISIONATO (SEMI-SUPERVISED LEARNING)

L'apprendimento parzialmente supervisionato si ha con problemi che hanno la maggior parte di dati di input e solo una parte di essi è classificata. La maggior parte dei problemi di machine learning nel mondo reale ricadono in quest'area dato che è costoso classificare i dati.

• APPRENDIMENTO CON RINFORZO (REINFORCEMENT LEARNING)

L'apprendimento con rinforzo è una tecnica di machine learning che rappresenta la versione computerizzata dell'apprendimento umano per tentativi ed errori. L'algoritmo si presta ad apprendere e ad adattarsi tramite un sistema di valutazione, che stabilisce una ricompensa se l'azione compiuta è corretta, oppure una penalità nel caso opposto. L'obiettivo è quello di massimizzare la ricompensa ricevuta, senza che venga annunciata la strada da intraprendere.

2.1.1 DEEP LEARNING

Una sottocategoria del machine learning è il deep learning o apprendimento profondo che insegna alla macchina a svolgere un'attività naturale per l'uomo: imparare con l'esempio. Nel deep learning un modello computerizzato impara a svolgere attività di classificazione direttamente da immagini, testo e suoni. Il deep learning rappresenta una tecnica di machine learning che utilizza algoritmi ispirati alla struttura e alla funzione del cervello umano chiamate reti neurali artificiali per apprendere e svolgere una determinata attività. Il deep learning sfrutta algoritmi che usano vari livelli non lineari a cascata per svolgere compiti di estrazione di caratteristiche, in modo tale che ogni livello successivo utilizza come input l'uscita del livello precedente. Inoltre, sfrutta algoritmi basati sull'apprendimento non supervisionato con livelli gerarchici multipli di caratteristiche dei dati, ovvero le caratteristiche di più alto livello vengono ottenute da quelle di più basso livello creando un rappresentazione gerarchica. Applicando il deep learning, avremo quindi una macchina che riesce autonomamente a classificare i dati ed a strutturarli in modo

gerarchico, trovando i più rilevanti e utili alla risoluzione di un problema, migliorando le proprie prestazioni con l'apprendimento continuo.

2.2 ORGANIZZAZIONE DEI DATI

In un modello di machine learning sono fondamentali il calcolo ed il ridimensionamento dei dati perché tutti gli algoritmi di apprendimento automatico richiedono un grande numero di dati per essere addestrati ed è necessario che questi dati siano validi, comprensibili, ovvero facilmente interpretabili dagli utenti, e che si trovino nella miglior forma per essere utilizzati, in modo tale che, sia per la macchina sia per l'utente, sia facile estrarre informazioni da essi. Proprio per questi motivi esistono diverse tecniche numeriche per semplificare il calcolo dei dati così da poter anche estrarre particolari caratteristiche più facilmente, insieme anche ad alcuni metodi per il ridimensionamento o per la regolarizzazione dei dati.

2.2.1 METODI PER IL CALCOLO DEI DATI

I metodi per il calcolo dei dati sono essenziali nel machine learning perché consentono, attraverso alcuni algoritmi, di risolvere più facilmente operazioni complesse e quindi rendono più agevole e rapido il calcolo di alcuni dati, permettendo anche di estrarre da questi più facilmente informazioni e caratteristiche necessarie per l'apprendimento automatico.
La tecnica più utilizzata è:

• FAST FOURIER TRANSFORM (FFT)

La trasformata di Fourier veloce, (FFT, dall'inglese Fast Fourier Transform), è un algoritmo ottimizzato per calcolare la trasformata discreta di Fourier (DFT) o la sua inversa. La trasformata discreta di Fourier è ampiamente utilizzata nel campo dell'elaborazione numerica dei segnali: per l'analisi delle frequenze contenute in un segnale, per risolvere equazioni differenziali alle derivate parziali, per il calcolo della convoluzione e della correlazione o per la compressione di dati necessaria per la memorizzazione e la trasmissione efficiente di questi. Alla base di tutti questi utilizzi c'è la possibilità di calcolare in modo efficiente la DFT attraverso la FFT. Infatti, la trasformata discreta di Fourier per essere calcolata direttamente richiede $\mathrm{O}(\mathrm{N}^2)$ operazioni aritmetiche, mentre utilizzando un algoritmo FFT si ottiene lo stesso risultato con $\mathrm{O}(\mathrm{N}\log(\mathrm{N}))$ operazioni. Inoltre, dato che l'antitrasformata discreta di Fourier è uguale alla DFT, tranne che per un esponente di segno opposto e un fattore

$\frac{1}{N}$, qualsiasi algoritmo FFT può essere facilmente invertito per calcolare anche l'antitrasformata.
L'algoritmo FFT più diffuso è l'algoritmo di Cooley-Tukey. Questo si basa sul principio di divide et impera, spezza ricorsivamente una DFT di qualsiasi dimensione N, con $N = N_1 N_2$, in DFT più piccole di dimensioni N_1e N_2, insieme a $O(n)$ moltiplicazioni per l'unità immaginaria. In generale quest'algoritmo si usa dividendo in due pezzi di $\frac{n}{2}$ ad ogni passo, ed è quindi ottimizzato solo per dimensioni che siano potenze di due, ma in generale può essere utilizzata qualsiasi fattorizzazione.
Quindi la FFT è utilizzata in una grande varietà di applicazioni in particolare per l'elaborazione di segnali digitali attraverso la quale vengono estratte caratteristiche utili per l'addestramento delle reti neurali artificiali finalizzate alla diagnosi dei guasti. Infatti, la FFT è nota come un metodo convenzionale di rilevamento guasti nelle macchine elettriche, specialmente nello stato di funzionamento stazionario.

2.2.2 METODI PER IL RIDIMENSIONAMENTO DEI DATI

Nella risoluzione dei problemi di machine learning, può capitare spesso di dover confrontare diverse unità di misura per prendere una decisione. Con le tecniche di ridimensionamento dei dati si riesce tramite delle trasformazioni e delle distribuzioni dei dati, ad uguagliare le caratteristiche dei dati al fine di poterli comparare su uno stesso livello di importanza. Fra le principali tecniche utilizzate si distinguono:
• LINEAR DISCRIMINANT ANALYSIS (LDA)
L'analisi discriminante lineare (LDA, dall'inglese Linear Discriminant Analysis) è una generalizzazione del discriminante lineare di Fisher, un metodo utilizzato nel riconoscimento dei modelli e nell'apprendimento automatico per trovare una combinazione lineare di caratteristiche che contrassegna o separa due o più classi di oggetti. Quando nei problemi di classificazione dell'apprendimento automatico, si devono confrontare ed analizzare troppe caratteristiche sulla base delle quali viene effettuata la classificazione finale, diventa difficile visualizzare il set di allenamento e lavorarci su e quindi in questi casi si utilizza la riduzione della dimensionalità. Questo metodo proietta un set di dati su uno spazio di dimensione inferiore rendendo le classi facilmente separabili al fine di evitare un eccesso di adattamento e per ridurre i costi di calcolo. La combinazione risultante può essere utilizzata come classificatore lineare o per la riduzione della dimensionalità prima della successiva classificazione. Quindi, l'obiettivo di un algoritmo LDA è di proiettare uno spazio di caratteristiche, ovvero un set di dati campioni n-dimensionali, su un sottospazio minore mantenendo le

informazioni discriminatorie di classe. In altre parole, trovare quel vettore che al meglio permette di dividere la rappresentazione delle classi.

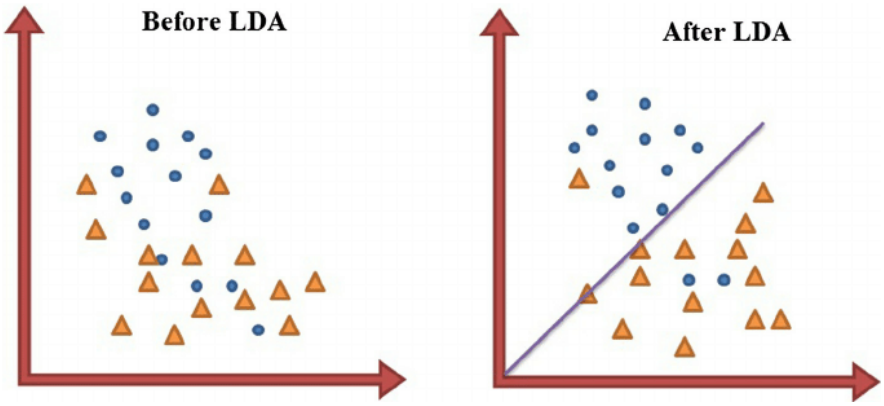


Figura 2.1: rappresentazione di un dataset prima e dopo l'applicazione dell'algoritmo LDA

Per eseguire un analisi discriminante lineare si devono seguire i seguenti passi:
1) Per ogni classe si deve calcolare la media in questo modo:

$$\mu_i = \frac{1}{N_i} \sum_{x \in w_i} x$$

dove N è il numero di campioni appartenente alla classe w i-esima e x sono gli esempi delle caratteristiche del problema.
Dato che la media delle proiezioni delle classi non è una buona misura in quanto non tiene conto della deviazione standard all'interno delle classi, per ottenere la migliore proiezione di divisione tra le due classi ci servono due matrici, quindi:
2) Si deve calcolare la matrice chiamata within scatter, che rappresenta la somma delle varie matrici di correlazione S_i delle varie classi del problema in esame, data dalla seguente formula:

$$S_W = \sum_{i=1}^{c} S_i$$

dove c è il numero di classi e S_i si ottiene da:

$$S_i = \sum_{x \in w_i} (x - \mu_i)(x - \mu_i)^T$$

Con μ pari alla media della classe i-esima, trovata al punto precedente.
3) Si deve calcolare la matrice chiamata between scatter, in cui la differenza tra le proiezioni delle medie viene espressa in termini di media dello spazio originale di caratteristiche e si calcola in questo modo:

$$S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)\,(\mu_i - \mu)^T$$

dove

$$\mu = \frac{1}{N}\sum_{\forall x} x = \frac{1}{N}\sum_{\forall x} N_i \mu_i$$

4) Si deve calcolare la miglior proiezione LDA unendo i risultati che abbiamo trovato nei punti precedenti. Per farlo si deve trovare la soluzione di quest'equazione:

$$j(w) = S_W^{-1} * S_B$$

Calcolando gli autovettori e gli autovalori corrispondenti. Ciò significa risolvere:

$$S_W^{-1} * S_B w_i = \lambda_i w_i$$

Con w che diventa il vettore proiezione, quindi w_i, vettore per la classe i-esima.

5) Si devono scegliere i migliori autovalori, ovvero una volta trovati gli autovalori, li ordiniamo in modo decrescente e selezioniamo il k superiore.

6) Si deve creare una nuova matrice W* contenente gli autovettori trovati

7) Si ottengono le nuove caratteristiche dalla Linear Discriminant Analysis dalla matrice Y:

$$Y = X \cdot W^*$$

dove X, che è la matrice delle caratteristiche del problema, è una matrice n×d con n campioni e dimensione d e Y è una matrice n×k con n campioni e dimensioni k<n. Quindi Y rappresenta il nuovo spazio delle funzionalità.

• PRINCIPAL COMPONENT ANALYSIS (PCA)

L'analisi delle componenti principali (PCA, dall'inglese principal component analisys) è un metodo utilizzato soprattutto per l'estrazione delle caratteristiche e la riduzione della dimensionalità. Il PCA permette di trovare le direzioni della massima varianza nei dati ad alta dimensione e di proiettarle su un nuovo sottospazio con dimensioni uguali o inferiori a quello originale. Il set di dati originale, che potrebbe aver coinvolto molte variabili, viene ad essere interpretato solo da poche variabili dette componenti principali. L'output del PCA sono proprio queste componenti principali, il cui numero è inferiore o uguale al numero di variabili originali. L'analisi delle componenti principali ci aiuta a identificare modelli nei dati in base alla correlazione tra funzionalità. Infatti, spesso accade che uno studio del set di dati di dimensioni ridotte consenta all'utente di individuare tendenze, modelli e valori anomali nei dati, molto più facilmente di quanto sarebbe stato possibile senza eseguire l'analisi delle componenti principali.

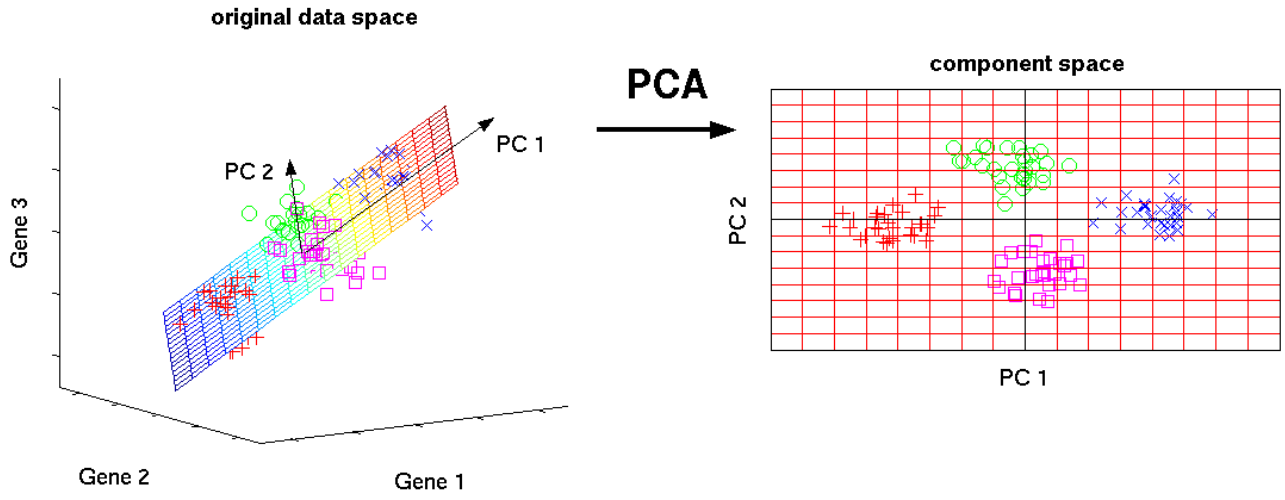


Figura 2.2: rappresentazione di un dataset prima e dopo l'applicazione di un algoritmo PCA

Gli approcci per risolvere un algoritmo PCA differiscono fra loro in base al problema che si deve risolvere, ma comunque i passaggi finali sono sempre gli stessi:

1) In alcuni casi per prima cosa si sottrae la media dalla matrice iniziale, ovvero si sottrae ad ogni colonna della matrice il valore della media ottenuta dagli elementi di quella stessa colonna.

In certe situazioni anziché sottrarre la media, si cerca di standardizzare la matrice iniziale, soprattutto quando si stanno confrontando diverse caratteristiche che hanno diverse unità di misura o un diverso ordine di grandezza, per garantire un confronto equo tra le stesse. La standardizzazione dei dati è il processo di ridimensionamento di uno o più attributi in modo che abbiano un valore medio di 0 e una deviazione standard di 1, che si ottiene calcolando il valore Z-standard per tutti gli elementi della matrice in questo modo:

$$z = \frac{x_i - u}{\sigma}$$

dove u è la media dei campioni di addestramento, σ è la deviazione standard dei campioni di addestramento e x_i è il valore che si vuole standardizzare.

2) Il secondo passaggio è quello di calcolare la matrice di covarianza. La covarianza misura la varianza tra due variabili, ovvero fornisce una misura di quanto due variabili varino assieme e questi valori tra le varie variabili sono inseriti in una matrice, chiamata appunto matrice di covarianza. Si utilizza la seguente formula:

$$Cov(x, y) = \frac{\sum(x_i - \bar{x})\,(y_i - \bar{y})}{n}$$

dove con x_i si indicano i valori della prima variabile, con y_i i valori della seconda variabile, $\bar{x}$ e $\bar{y}$ rispettivamente la media dei valori della prima e seconda variabile e con n il numero dei valori dati.

In alcuni casi, invece, viene calcolata la matrice di correlazione che analizza sempre la relazione tra due variabili, ma mostrando i coefficienti di correlazione tra le varie variabili.
3) Si calcolano gli autovalori e gli autovettori della matrice ottenuta
4) Si ordinano gli autovalori e si scelgono le componenti principali: l'obiettivo è quello di ridurre la dimensionalità del nostro spazio delle funzioni, proiettandolo tramite PCA su un sottospazio più piccolo, in cui gli autovettori formeranno gli assi di questo nuovo sottospazio di caratteristiche. Per decidere quale autovettore (i) vogliamo eliminare per il nostro sottospazio di dimensione inferiore, dobbiamo esaminare i loro corrispondenti autovalori, perché gli autovettori con gli autovalori più bassi recano il minor numero di informazioni sulla distribuzione dei dati, e quelli sono quelli che vogliamo eliminare. L'approccio comune è quello di classificare gli autovalori, dal più alto al più basso, e scegliere i primi k autovettori corrispondenti.
5) L'ultimo passaggio è quello di formare le componenti principali, per farlo moltiplichiamo la trasposizione della matrice di autovettori appena creata e la moltiplichiamo con la trasposizione della versione ridimensionata del set di dati originale, ossia:

$$CP = W^T B^T$$

dove CP è la matrice costituita dalle componenti principali; W^T è la matrice che abbiamo formato usando gli autovettori che abbiamo scelto di mantenere dallo step precedente; B^T è la versione ridimensionata (o standardizzata) del set di dati originale.

Inoltre, fra i metodi di ridimensionamento dei dati troviamo anche quelli di regolarizzazione. La regolarizzazione implica l'introduzione di ulteriori informazioni allo scopo di risolvere un problema mal condizionato o per prevenire l'eccessivo adattamento. In un modello di machine learning, la regolarizzazione riduce significativamente la varianza del modello, attraverso l'introduzione di un parametro λ, si punta a ridurre alcuni coefficienti in modo da ridurne la varianza. Fra le principali tecniche si evidenziano:

• REGRESSIONE RIDGE

Con regressione ridge ci si riferisce a un modello di regressione lineare i cui coefficienti sono stimati dallo stimatore ridge, che riduce i coefficienti di regressione, in modo che le variabili, con un contributo minore al risultato, abbiano i loro coefficienti vicini allo zero. Invece di forzarli a essere esattamente zero, li penalizziamo con un termine chiamato norma L2, costringendoli così a essere piccoli in modo continuo, in questo modo, diminuiamo la complessità del modello senza eliminare nessuna variabile. Attraverso l'introduzione della costante λ si può scegliere al meglio l'ammontare di questa penalità utilizzando la seguente equazione:

$$\hat{B}_\lambda: arg\, min_b \sum_{i=1}^{n} (y_i - x_i b)^2 + \lambda \sum_{k=1}^{K} b_k^2$$

Questa equazione è risolta da:

$$\hat{B} = (X^T X + \lambda I)^{-1} X^T y$$

dove I è la matrice identità di dimensione kxk, y è la matrice dei valori osservati della variabile dipendente e X è la matrice dei valori della variabile indipendente. Ovviamente la scelta di un buon valore per λ è fondamentale; nello stimatore ridge all'aumentare di λ l'impatto della penalità aumenta e i coefficienti di regressione si avvicinano allo zero, mentre se λ è vicino allo zero il termine penalità avrà un effetto troppo basso. Un importante vantaggio della regressione ridge è che si comporta bene anche in una situazione in cui si hanno molti dati multivariati con il numero di predittori (p) maggiore del numero di osservazioni (n). Uno svantaggio, invece, è che includerà tutti i predittori nel modello finale, invece di selezionare solo un insieme ridotto di variabili tra quelle disponibili, proprio perché la regressione ridge riduce i coefficienti verso zero, ma non ne imposta nessuno esattamente a zero.

• LEAST ABSOLUTE SHIRNKAGE AND SELECTION OPERATOR (LASSO)

Lasso è l'acronimo di Least Absolute Shrinkage and Selection Operator ovvero una tecnica che riduce i coefficienti di regressione verso lo zero penalizzando il modello di regressione con un termine di penalità chiamato norma L1, che è la somma dei coefficienti assoluti. l'equazione di una regressione lasso è la seguente:

$$\hat{B}_\lambda: arg\, min_b \sum_{i=1}^{n} (y_i - x_i b)^2 + \lambda \sum_{k=1}^{K} |b_k|$$

Nel caso della regressione lasso, la penalità, che dipende dalla scelta di λ, ha l'effetto di forzare alcune delle stime dei coefficienti, con un contributo minore al modello, a essere esattamente uguale a zero, ciò significa che il lasso esegue anche una selezione delle variabili al fine di ridurre la complessità del modello. Anche qui è fondamentale la scelta di un buon valore di λ: quando è piccolo, il risultato è molto vicino alla stima dei minimi quadrati, all'aumentare di lambda, invece, si verifica una contrazione in modo da poter eliminare le variabili che sono a zero.

Un ovvio vantaggio della regressione lasso rispetto alla regressione ridge è che produce modelli più semplici e più interpretabili che incorporano solo un insieme ridotto di predittori. In generale, il lasso potrebbe funzionare meglio in una situazione in cui alcuni predittori hanno coefficienti elevati e i restanti predittori hanno coefficienti molto piccoli. La regressione ridge funzionerà meglio quando il risultato è una funzione di molti predittori, tutti con coefficienti di dimensioni approssimativamente uguali.

2.3 DATA DRIVEN

I modelli basati sui dati, ovvero i data-driven, applicano tecniche di statistica o di machine learning ai dati raccolti relativi alle macchine, con lo scopo di poter poi riconoscere lo stato dei componenti. L'idea è quella di riuscire ad ottenere il maggior numero di informazioni riguardo lo stato dei macchinari in tempo reale e di correlarli con il livello di degrado dei singoli componenti o con le performance del sistema nel suo complesso. Questi modelli hanno il grande vantaggio di non richiedere conoscenze approfondite specifiche per il dominio di applicazione. La scelta di un modello di tipologia data-driven è fortemente dipendente dall'obiettivo che si vuole ottenere dal sistema, in base a quest'ultimo, infatti, il problema viene modellato in modo differente. Le principali opzioni sono riportate di seguito:

• CLASSIFICAZIONE BINARIA: in cui ogni singolo input rappresentante lo stato del sistema deve essere etichettato con uno fra due possibili valori mutuamente esclusivi. Ad esempio, decidere se la macchina sta funzionando correttamente o non correttamente oppure decidere se la macchina può guastarsi entro un intervallo di tempo fissato.

• CLASSIFICAZIONE MULTICLASSE: La versione multiclasse è una generalizzazione della classificazione binaria, in cui viene incrementato il numero di possibili etichette fra cui scegliere. Ad ogni input deve comunque essere associata una sola etichetta.

• REGRESSIONE: rappresenta il fatto che una macchina può predire il valore di ciò che sta analizzando in base a dati attuali. In altre parole, studia la relazione tra due o più variabili una indipendente dall'altra. La regressione può essere utilizzata per stimare la vita utile rimanente di un componente in termini di un numero continuo, fornito appunto dal modello di regressione, di unità di tempo prefissata.

• ANOMALY DETECTION: il modello deve essere in grado di stabilire se il funzionamento della macchina rientra in uno stato normale o se si discosta da esso, rientrando cioè in un caso di anomalia. Questa metodologia si differenzia dalla classificazione dato che rientra nei casi di learning semi-supervisionato infatti necessita solamente di imparare da input che rappresentano stati di funzionamento corretti.

2.4 RETI NEURALI

Le reti neurali artificiali sono modelli matematici composti da neuroni artificiali e vengono utilizzate per risolvere problemi ingegneristici di intelligenza artificiale legati a diversi ambiti tecnologici. Volendo dare una definizione più dettagliata potremmo dire che le reti neurali sono modelli di calcolo matematico-informatici basati sul

funzionamento delle reti neurali biologiche, ossia modelli costituiti da interconnessioni di informazioni; tali interconnessioni derivano da neuroni artificiali e processi di calcolo basati su un'elaborazione parallela e distribuita delle informazioni, allo stesso modo del cervello umano, che elabora le informazioni ricevute dai vari sensi in modo parallelo e le distribuisce in vari nodi della rete. Una rete neurale artificiale di fatto si presenta come un sistema in grado di modificare la sua struttura basandosi sia su dati esterni sia su informazioni interne che si connettono e passano attraverso la rete neurale durante la fase di apprendimento e ragionamento. Le reti neurali artificiali sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione: ricevono segnali esterni su uno strato di nodi d'ingresso, ognuno di questi è collegato a svariati nodi interni della rete che, tipicamente, sono organizzati a più livelli in modo che ogni singolo nodo possa elaborare i segnali ricevuti trasmettendo ai livelli successivi il risultato delle sue elaborazioni. In generale le reti neurali sono formate da tre strati:

• Lo strato degli ingressi (I – Input): livello progettato per ricevere le informazioni provenienti dall'esterno al fine di imparare, riconoscere e processare le stesse informazioni ricevute; è quello che ha il compito di ricevere ed elaborare i segnali in ingresso adattandoli alle richieste dei neuroni della rete;

• Lo strato nascosto (H – Hidden): è quello che ha in carica il processo di elaborazione vero e proprio e solitamente è strutturato con più livelli di neuroni; collegano il livello di ingresso con quello di uscita e aiutano la rete neurale ad imparare le relazioni complesse analizzate dai dati;

• Lo strato di uscita (O – Output): livello finale che mostra il risultato di quanto il programma è riuscito a imparare, qui vengono raccolti i risultati dell'elaborazione dello strato H e vengono adattati alle richieste del successivo livello della rete neurale.

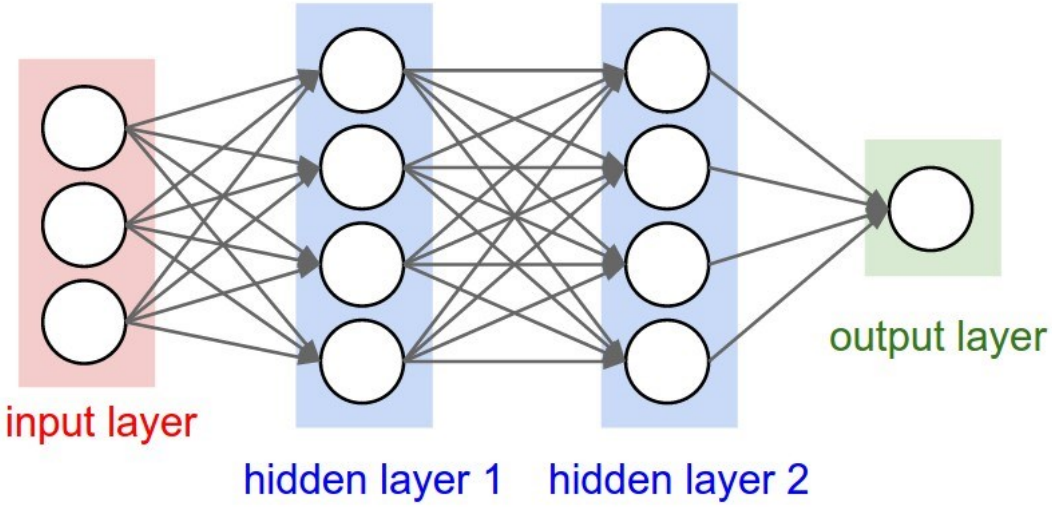


Fig.2.3: Rete neurale a 3 strati, con 2 strati nascosti, 3 ingressi, 1 uscita

Partendo dalla struttura biologica delle reti neurali umane, quelle artificiali possono essere modellizzate come delle funzioni non lineari che trasformano degli input $(x_1, x_2, \dots, x_i, \dots, x_n)$ in un output y, ad ogni input è associato un peso $(w_1, \dots w_n)$, in più c'è un altro parametro w_0 il bias, il quale può essere visto come il peso dell'input x_0 che viene posto costante a 1. Possiamo ora definire l'attivazione del neurone come la somma pesata dei vari input:

$$a = \sum_{i=0}^{N} w_i x_i$$

dove $x_0 = 1$

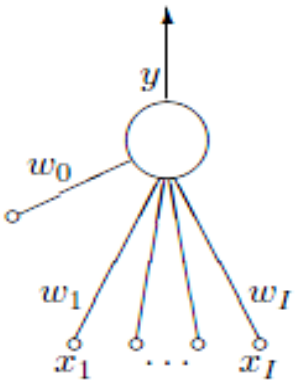


Figura 2.4: modello matematico del neurone

Il modello non è completo solo con l'attivazione, è presente anche una funzione di attivazione $y = f(a)$, che da l'output finale del neurone. Si può scegliere questa funzione in molti modi, tra cui le principali sono:

- funzione lineare

$$a = \sum_{k=1}^{N} w_i x_i$$

- funzione soglia

$$y(a) = \begin{cases} 1, & \text{se } a > 0 \\ -1, & \text{se } a \leq 0 \end{cases}$$

- funzione tangente iperbolica:

$$y(a) = \tanh(a)$$

- Funzione ReLU:

$$y(a) = \max(0, a)$$

Quindi gli elementi che, insieme a un algoritmo di apprendimento, danno vita a una rete neurale sono i vari input, cioè le informazioni proveniente dall'esterno che vengono fornite allo strato degli ingressi, attraverso l'elaborazione dei quali si arriva ai vari output della rete neurale i quali, in una rete formata da due livelli di unità di elaborazione, con d ingressi, m uscite per il primo strato e c uscite per il secondo, potrebbero essere dati da un equazione di questo tipo:

$$y_k = y\left(\sum_{j=0}^{m} w'_{kj}\, z\left(\sum_{i=0}^{d} w'_{ji} x_i\right)\right)$$

dove y_k è l'ouput finale, w'_{kj} sono i pesi per ogni unità di elaborazione, z è il segnale inviato dalle unità nascoste, x_i sono gli input e w'_{ji} sono i pesi di ogni input combinati con ogni output. Per definire meglio cosa sono questi ultimi elementi fondamentali sia nell'attivazione che nell'output delle reti neurali consideriamo che la maggior parte delle reti sono completamente connesse, ovvero ciascun neurone è collegato a tutti i neuroni dello strato successivo tramite connessioni pesate e i pesi rappresentano proprio queste connessioni che sono quindi dei valori numerici che vengono moltiplicati per il valore del neurone collegato. Ciascun neurone somma i valori pesati di tutti i neuroni ad esso collegati e aggiunge un valore di bias. A questo risultato viene applicata una funzione di attivazione, che determina il valore del segnale in uscita in base alla somma dei valori in ingresso moltiplicati per i relativi pesi prima di passarlo allo strato successivo. In questo modo i valori di input vengono propagati attraverso la rete fino ai neuroni di output, l'obiettivo è quello di regolare pesi e bias opportunamente in modo da arrivare ad ottenere il risultato voluto. Il termine bias deve essere inteso come caratterizzante di ogni singolo neurone della rete, perciò, nel complesso, nella rete ce ne saranno diversi. Il valore associato ai biases è variabile, così come lo è per i pesi, e viene corretto nella fase di apprendimento. Per comprendere il significato e l'utilità dei bias si deve considerare ogni singolo bias come una soglia che determina se l'informazione entrata ed elaborata dal neurone debba o meno uscirne procedendo in avanti nella rete. In altre parole, il bias determina se e in quale misura il neurone debba attivarsi, inoltre la presenza di questo parametro incrementa la flessibilità del modello, consentendogli di adattarsi meglio al dataset. Come parametro di un modello statistico, il bias è 'appreso' o 'stimato' trovando il punto di minimo di una particolare funzione detta funzione di perdita che dipende dal nostro dataset. È importante avere un parametro come il bias nella propria rete perché ogni neurone riceve, dal precedente livello, una somma di input pesati che passa attraverso una funzione di attivazione e invece di passare alla funzione direttamente la somma pesata, si aggiunge un parametro in più

ovvero il bias, per capire meglio l'importanza dell'aggiunta di questo parametro si consideri il seguente esempio:

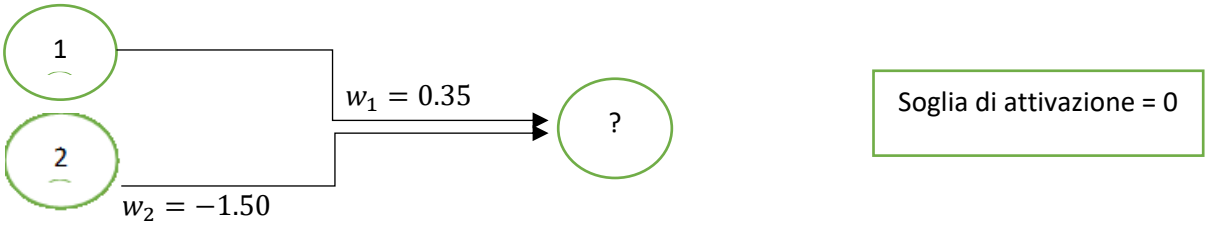


Figura 2.5: schematizzazione di un neurone nascosto con 2 input e soglia di attivazione pari a 0

L'input del primo neurone (1) e l'input del secondo (2), nell'input layer, vengono moltiplicati per il peso della rispettiva connessione al neurone dell'hidden layer. I due risultati parziali vengono poi sommati tra loro e passati alla funzione di attivazione, la ReLU ($f(x) = \max(0, x)$) ad esempio, quindi la funzione per calcolare il valore del neurone dell'hidden layer è:

$$g(x_1 w_1 + x_2 w_2) =$$
$$g\big(1 \cdot 0.35 + 2 \cdot (-1.50)\big) =$$
$$g(-2.65) \rightarrow \max(0, -2.65) = 0$$

Quindi l'input ricevuto dal terzo neurone sarà dunque pari a zero, in questo modo il neurone sarà disattivato e il suo peso sarà irrilevante nella rete. Se volessimo invece abbassare la soglia di attivazione del neurone, ovvero il livello oltre al quale il neurone risulta attivo, dovremmo introdurre il bias, il cui valore dovrà essere opposto a quello della soglia desiderata. Se quindi volessimo abbassare la soglia a -3 sceglieremmo come bias il suo opposto, ovvero 3, e lo aggiungiamo subito prima di calcolare la funzione di attivazione, quindi la funzione diventa:

$$g(x_1 w_1 + x_2 w_2 + \beta) =$$
$$g(0.35) \rightarrow \max(0,0.35) = 0.35$$

Quindi adesso con un valore di 0.35 il neurone risulterà attivo.

La parte complessa e fondamentale per una rete neurale è il suo apprendimento, infatti questo avviene quando c'è un qualche tipo di feedback, ossia una risposta che permette di verificare se si è appreso quello che si sta imparando. Le reti neurali, solitamente, utilizzano per imparare un algoritmo chiamato back propagation (BP), Il termine back propagation è legato essenzialmente alla tecnica utilizzata per il calcolo delle derivate della funzione di errore, basata sulle regole di derivazione delle funzioni composte. L'algoritmo confronta il valore in uscita del sistema con il valore desiderato, calcolando cosi l'errore, e modifica i pesi delle connessioni tra i livelli della rete partendo dal livello output, poi procedendo a ritroso prevede di modificare i pesi dei livelli nascosti e infine quelli dei livelli di input della rete, facendo quindi convergere progressivamente il set dei valori di uscita verso quelli desiderati.

Più in particolare data una rete neurale con un solo nodo N e una sola entrata X, il sistema ha l'obiettivo di raggiungere un determinato livello di uscita $\mathrm{Y_D}$. Al termine di ogni ciclo, l'algoritmo confronta il risultato ottenuto, Y, con quello desiderato, $\mathrm{Y_D}$, e calcola un errore $e = Y_D - Y$. La retroazione, ovvero il feedback, consente all'algoritmo di utilizzare l'errore per aggiustare il peso sinaptico p_t ed eseguire una nuova iterazione.

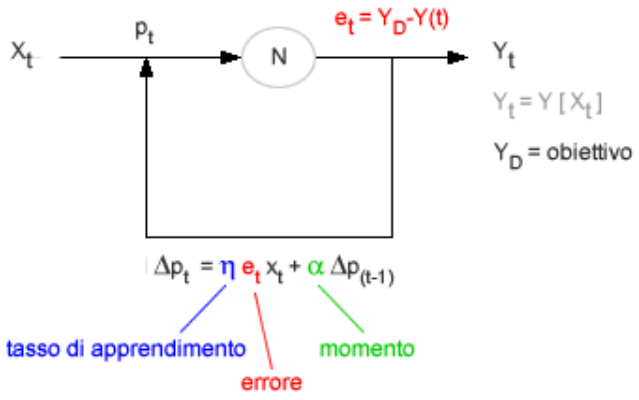
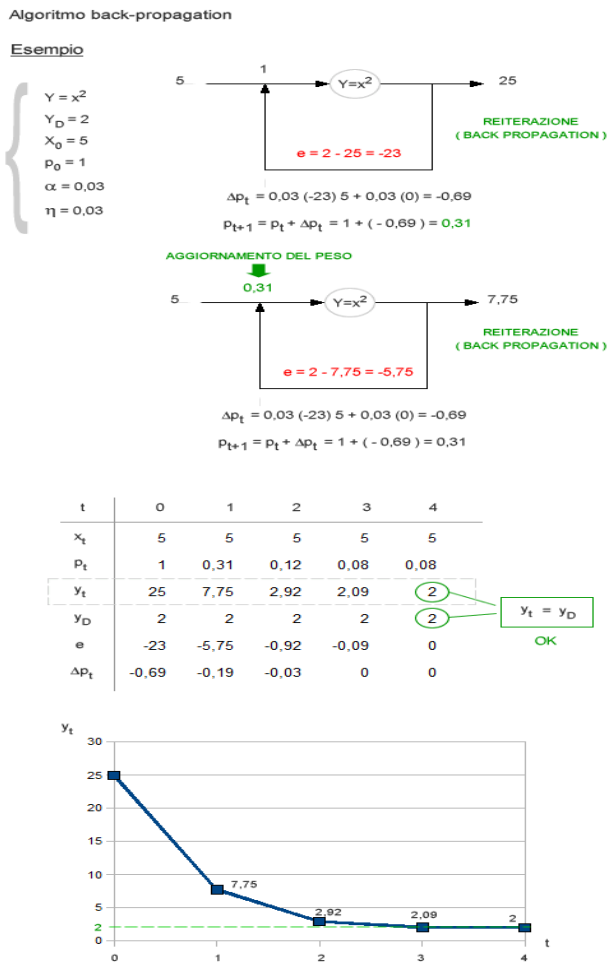


Figura 2.6: schematizzazione del funzionamento di un algoritmo di back propagation

L'aggiustamento del peso sinaptico è determinato dall'errore e da due parametri del sistema: il tasso di apprendimento e il momento. Entrambi i parametri possono variare da zero a uno: il tasso di apprendimento influenza la velocità del processo di apprendimento. Quanto più è elevato il tasso di apprendimento, vicino al valore massimo uno, tanto più veloce è il processo di apprendimento, in quanto sono più grandi le variazioni da apportare sul peso sinaptico. Tuttavia, un tasso di apprendimento molto alto aumenta anche la probabilità che il risultato oscilli troppo e, quindi, l'instabilità del sistema. Per ridurre le oscillazioni e favorire la convergenza dei risultati, nella formula di aggiustamento del peso sinaptico si utilizza un secondo parametro α , detto momento.
Nel seguente esempio è mostrato un processo dinamico di apprendimento utilizzando l'algoritmo di back propagation:
Si parte da una situazione iniziale in cui l'input è 5, $x = 5$, che supponiamo rimanga costante per tutto il processo di aggiustamento del peso sinaptico, e il peso sinaptico è fissato casualmente ed è pari al valore uno $\mathrm{p_t} = 1$. In cinque reiterazioni l'algoritmo modifica progressivamente il peso sinaptico $\mathrm{p_t}$ fino a raggiungere l'obiettivo desiderato, $\mathrm{Y_D} = 2$. Dopo l'ultima iterazione la rete neurale è stabile e il peso sinaptico rimane costante al valore di equilibrio $\mathrm{p_t} = 0.08$.



t	0	1	2	3	4
x_t	5	5	5	5	5
p_t	1	0,31	0,12	0,08	0,08
y_t	25	7,75	2,92	2,09	2
y_D	2	2	2	2	2
e	-23	-5,75	-0,92	-0,09	0
Δp_t	-0,69	-0,19	-0,03	0	0

Figura 2.7: esempio di un applicazione di algoritmo di back propagation

Non è però detto che l'algoritmo raggiunga sempre le condizioni di equilibrio. Ad esempio, impostando un tasso di apprendimento più alto, a parità di condizioni il processo di aggiustamento diventa divergente e l'algoritmo non raggiunge l'equilibrio, poiché le oscillazioni diventano sempre più grandi.

Strettamente legato all'algoritmo di back propagation c'è il metodo di discesa del gradiente che è una tecnica che consente di determinare i punti di massimo e minimo di una funzione di più variabili. La discesa del gradiente è ampiamente utilizzata nell'apprendimento automatico soprattutto per l'addestramento tramite apprendimento supervisionato delle reti neurali, ma in generale anche per altri modelli di machine learning. La discesa del gradiente sfrutta il principio chiamato regola delta che permette di aggiornare i pesi dei segnali di input di una rete neurale valutando il modello su un input il cui corrispondente output esatto sia noto, e correggendo ciascun parametro del modello in quantità proporzionale, ma di segno opposto, rispetto al suo contributo all'errore sul risultato. La back propagation implementa proprio questo principio dato che il contributo di ciascun parametro all'errore del modello è dato dalla derivata parziale della funzione di perdita rispetto

al parametro stesso. La regola delta può anche essere applicata a reti neurali ad esempio di tipo feedforward, cioè con propagazione unidirezionale dei segnali, e permette di calcolare la differenza tra i valori di output che la rete ottiene e quelli che invece dovrebbe apprendere.
Data una rete neurale di questo tipo l'obiettivo che ci si prefigge è minimizzare la diversità tra i valori di attivazione degli output della rete, ottenuti sommando i segnali provenienti dai diversi input moltiplicati per i pesi delle connessioni in ingresso, e i valori della risposta desiderata; questa differenza viene quantificata attraverso una funzione di perdita. La funzione che si vuole minimizzare è il valore atteso della perdita, per applicare il metodo del gradiente, la funzione di perdita deve essere derivabile rispetto ai valori di output. Nella fase di addestramento, variando i pesi sinaptici, si può aumentare o diminuire la funzione da minimizzare, la prestazione della rete sarà funzione delle variabili e sarà massima quando si raggiunge il minimo della funzione di perdita, il che si ottiene applicando il metodo della discesa del gradiente e aggiornando iterativamente i valori dei pesi della rete.

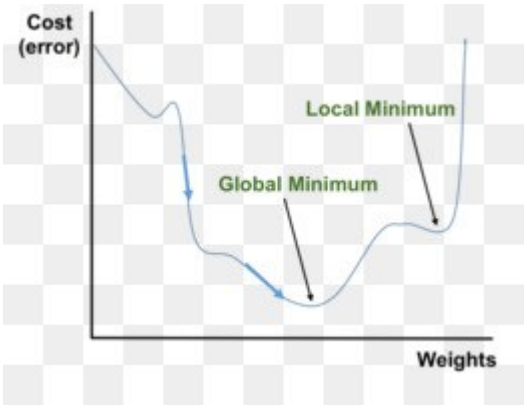


Figura 2.8: metodo di discesa del gradiente che permette di raggiungere il minimo della funzione di costo

Il metodo di discesa del gradiente si basa sul fatto che, per una funzione $f(x)$, la direzione di discesa massima, ovvero un vettore che spostandosi nella direzione indicata da esso permette di avvicinarsi a un minimo locale della funzione, in un punto $\bar{x}$ assegnato corrisponde a quella determinata dall'opposto del suo gradiente in quel punto $p_k = -\nabla f(\bar{x})$, questa scelta per la direzione di discesa garantisce che la soluzione tenda a un punto di minimo della funzione. L'algoritmo di discesa del gradiente necessita soltanto di due cose: il gradiente, per quella posizione, e la larghezza del passo da compiere, ovvero il tasso di apprendimento. Con queste informazioni, il valore corrente di ciascun parametro viene aggiornato e con i nuovi valori dei parametri, il gradiente viene ricalcolato e il processo viene ripetuto. Dopo molti passaggi, una volta che ci si rende conto che il costo non migliora di molto ed è bloccato vicino a un punto particolare, si ha la convergenza e i valori dei parametri

nell'ultima fase costituiscono il migliore insieme che si può ottenere. Quindi il metodo del gradiente prevede di partire da una soluzione iniziale x_0 scelta arbitrariamente e di procedere iterativamente aggiornandola in questo modo:

$$x_{k+1} = x_k + a_k p_k$$

dove a_k corrisponde alla lunghezza del passo di discesa, ovvero il tasso di apprendimento, importante per determinare la velocità con cui l'algoritmo convergerà alla soluzione richiesta.

In conclusione, per quanto riguarda i vantaggi delle reti neurali queste permettono di avere un addestramento statistico meno formale, una capacità di rilevare implicitamente relazioni non lineari complesse tra variabili dipendenti e indipendenti, una capacità di rilevare tutte le possibili interazioni tra variabili predittive e una disponibilità di algoritmi di allenamento multipli. Tuttavia, la sua natura di "scatola nera", il suo elevato costo computazionale nella fase di addestramento e la sua natura empirica per lo sviluppo di modelli sono considerati degli svantaggi per le reti neurali.

CAPITOLO 3

3.1 SISTEMA DI STUDIO

La rete neurale, obiettivo principale di questa tesi, è stata progettata per rilevare e classificare i guasti di un sistema composto da robot cartesiani (CR), su quest'ultimo era già stato effettuato un lavoro in precedenza in cui si analizzavano i guasti più comuni relativi agli azionamenti di movimento elettrico di questi robot e cercare di sviluppare una manutenzione predittiva (Pdm) per prevedere tali guasti ed è proprio da questa analisi che ci si è basati per la realizzazione della rete neurale.

I CR sono robot in grado di muoversi sui tre assi cartesiani, x, y e z in maniera lineare e quindi sono composti dalla combinazione di due o tre assi lineari. La configurazione a due assi permette al robot di lavorare su un piano cartesiano mentre la soluzione con tre assi identifica tre piani nello spazio, in cui ogni asse ha un solo grado di libertà e con la combinazione del movimento di tutti gli assi si raggiunge ogni punto dell'area di lavoro richiesta. Quindi si tratta di un robot industriale i cui assi di controllo principali sono lineari, ovvero si muovono in linea retta, e perpendicolari tra loro, questa disposizione meccanica semplifica la soluzione del braccio di controllo del robot, quindi ha un'elevata affidabilità e precisione quando si opera in uno spazio tridimensionale.

Figura 3.1 robot cartesiano a 3 assi

Ogni CR viene realizzato, a seconda delle necessità, con moduli a cinghia, a vite o con motore lineare tenendo in considerazione la massa del carico, la precisione, l'area ed il ciclo di lavoro richiesto, nel CR considerato il sistema di trasmissione è costituito da una coppia cinghia-motore, in cui si ha un motore sincrono a magneti permanenti (PMSM), questi ultimi sono caratterizzati, generalmente, da uno statore con avvolgimenti trifase simmetrici e da un rotore per il quale viene utilizzato un magnete permanente di forma speciale, costituito da un elemento chiamato terre rare al posto degli avvolgimenti di campo. Quindi l'obiettivo è quello di sviluppare un sistema Pdm proprio per il sistema di trasmissione a cinghia dato che è uno dei componenti più critici e più sensibile al degrado, infatti l'allentamento della cinghia è la causa principale dei guasti comuni degli azionamenti elettrici di movimento dei CR. L'idea di base è quella di trovare le cause, per poter poi sviluppare un modo intelligente per capire quando il sistema passa da una situazione di sicurezza a una situazione critica. Per evidenziare funzionalità difettose si può effettuare un analisi dello spettro di corrente dello statore per rivelare guasti nei sistemi a cinghia, dove multipli della frequenza naturale della cinghia nel segnale elettrico indicano problemi. In questo caso l'analisi spettrale, però, non può essere effettuata per mezzo della trasformata veloce di Fourier dato che non è adatta ai sistemi robotici industriali come i CR, perché spesso essi lavorano ad alta velocità e solitamente sono adottati per operazioni di "pick and place", ovvero utilizzati nella produzione industriale per il prelievo e il deposito, quindi dove il profilo di riferimento della velocità dei motori cambia nel tempo e gli rpm del motore cambiano di conseguenza. Quindi qui il segnale della corrente cambia in frequenza e tempo e l'analisi spettrale tramite FFT dà luogo a una media di tutte le frequenze dello spettro, rendendo impossibile capire in quale momento si verificano, quindi estrarre le funzionalità tempo-frequenza da questa analisi si rivela inutile. Nel lavoro analizzato invece sono state applicate tecniche di analisi spettrale della corrente del motore (motor current signature analysis, MCSA),

più precisamente è stata utilizzata la decomposizione wavelet che permette l'analisi tempo-frequenza di segnali sia stazionari che non stazionari. Il metodo MCSA è stato utilizzato per valutare il segnale elettrico del motore, acquisendolo quando il sistema funziona in condizioni di sicurezza e quando potrebbe verificarsi l'errore, in modo tale da poter evidenziare il diverso comportamento.

3.2 PREELABORAZIONE DEI DATI

3.2.1 MOTOR CURRENT SIGNATURE ANALYSIS (MCSA)

L'MCSA è una tecnica di monitoraggio delle condizioni utilizzata per diagnosticare problemi nei motori e può essere utilizzato come strumento di manutenzione predittiva per rilevare i guasti comuni del motore nella fase iniziale e quindi prevenirli. L'MCSA monitora la corrente di alimentazione del motore, viene comunemente utilizzato un sistema di monitoraggio della corrente dello statore singolo, ovvero un monitoraggio solo di una delle tre fasi della corrente di alimentazione del motore. Gli avvolgimenti dello statore del motore sono usati come trasduttori nel MCSA, raccogliendo i segnali dal rotore, ma rivelando anche informazioni sullo stato dello statore. La corrente del motore viene rilevata da un sensore di corrente con derivatore di corrente attraverso la sua uscita e registrata nel dominio del tempo, il segnale di corrente prelevato viene quindi portato a un analizzatore di spettro o ad uno strumento MCSA specializzato. Nel caso ideale la corrente del motore dovrebbe essere un'onda sinusoidale pura, ma in realtà sono presenti molte armoniche, diverse possibili condizioni di guasto elettrico e meccanico presenti nel motore modulano ulteriormente il segnale di corrente e contribuiscono ad aumentare le armoniche di banda laterale. I guasti nei componenti del motore producono, quindi, corrispondenti anomalie nel campo magnetico che appaiono nello spettro della corrente di alimentazione del motore come bande laterali attorno alla frequenza di linea. Quindi sulla base dello spettro di corrente è possibile identificare se ci sono guasti del motore ed eventualmente accedere alla relativa gravità. Ci sono diversi metodi che possono essere utilizzati, sfruttando l'MCSA, per estrarre le caratteristiche della corrente di alimentazione del motore e per confrontarla con lo spettro dei guasti del motore noti. Nel caso di studio il segnale della corrente dei motori del CR è stato analizzato utilizzando la decomposizione wavelet.

3.2.2 TRASFORMATA WAVELET

Una wavelet è una forma d'onda oscillante, di lunghezza finita, con valore medio nullo ed energia diversa da zero, che tende ad avere un andamento irregolare e asimmetrico. La wavelet è uno strumento utilizzato per analizzare segnali sia stazionari che non stazionari, quindi consente l'analisi localizzata nel dominio della frequenza o della scala temporale.

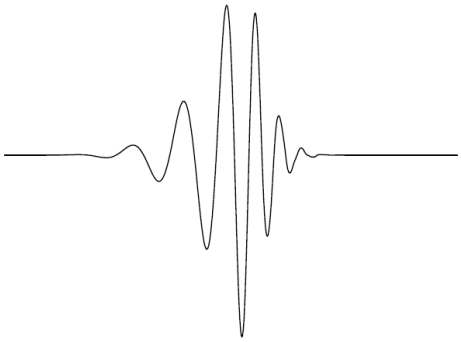


Figura 3.2 Un esempio di onda wavelet

Nella trasformata wavelet (WT), il segnale viene decomposto utilizzando una famiglia di funzioni, le quali, singolarmente, sono la versione scalata e traslata della funzione wavelet originale, chiamata wavelet madre, quindi si rappresenta il segnale mediante l'uso di quest'ultima. Quindi la trasformata wavelet mediante una scomposizione del segnale di corrente in diversi livelli, scelto un adeguato wavelet madre, fornisce una rappresentazione tempo-frequenza del segnale e restituisce, attraverso l'analisi multi-risoluzione, il segnale in diverse bande d'ottava di frequenza trovando così il contenuto energetico in ciascuna di esse. In tal caso, ci si aspetta che i segnali con guasto, dovuti nel nostro caso ad allentamenti della cinghia, mostrino energia diversa nei livelli di decomposizione. La wavelet può essere dilatata o contratta in base al valore del fattore di scala utilizzato, in particolare la trasformata wavelet possiede una buona risoluzione nel tempo per valori bassi del fattore di scala, a scapito di quella in frequenza, mentre presenta una buona risoluzione in frequenza e peggiore risoluzione nel tempo per fattori di scala elevati. Esiste, quindi, una relazione frequenza-scala, per cui la WT permette di avere una buona risoluzione in frequenza alle basse frequenze, ma, contemporaneamente, una buona localizzazione temporale alle alte frequenze. Quando si effettua una decomposizione del segnale multi-risoluzione con la WT si decompone iterativamente il segnale a risoluzioni diverse, ciascuna corrispondente ad un differente valore di scala, per tutta la sua lunghezza. Come risultato si avrà una collezione di rappresentazioni tempo-frequenza del segnale, tutte ad una differente risoluzione, questo approccio offre la possibilità di studiare discontinuità, aperiodicità e altre caratteristiche non stazionarie del segnale. Le trasformate wavelet possono essere classificate a livello generale in: trasformata

wavelet continua (Continuous Wavelet Transform, CWT) e trasformata wavelet discreta (Discrete Wavelet Transform, DWT). La CWT confronta il segnale oggetto di studio con le versioni traslate e scalate della wavelet madre per valutare la somiglianza tra le due, in questo caso la wavelet madre è:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)$$

dove a e b, rispettivamente, sono i parametri di scala e traslazione, con a, b ϵ R e a>0. Ogni versione scalata della wavelet madre deve trasportare la stessa quantità di energia, per cui si normalizza tramite $\frac{1}{\sqrt{|a|}}$. Confrontando il segnale con le wavelet a differenti scale e posizioni, si ottiene una funzione a due variabili. Le informazioni fornite dalla CWT a scale e localizzazioni temporali ravvicinate, però, sono altamente correlate, per cui si avrà una rappresentazione ridondante del segnale, richiedendo un importante onere di calcolo. In particolare, la CWT di un segnale s(t) è:

$$C_{a,b,\psi(t)} = \int_{-\infty}^{+\infty} s(t)\frac{1}{\sqrt{a}}\psi^*\left(\frac{t-b}{a}\right)dt$$

dove * indica il complesso coniugato. Dunque, per ogni valore di a e b, quanto più il segnale e la wavelet sono simili, quanto più il coefficiente restituito dalla CWT avrà un valore elevato. Inoltre, quanto più il fattore di scala è elevato, ossia la wavelet risulta più dilatata, tanto più è larga la porzione di segnale a cui è confrontata, quindi più approssimative saranno le caratteristiche del segnale misurate dai coefficienti wavelet.

In realtà, però, si ha sempre a che fare con segnale discreto, campionato, quindi viene sempre utilizzata la trasformata wavelet discreta, la DWT, questa permette anche di superare il problema della forte ridondanza nei coefficienti delle CWT. Nei DWT la rappresentazione della wavelet viene modificata dato che prevede la discretizzazione dei parametri di scala e traslazione, quindi si ha che la wavelet madre è:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{a_0^j}}\psi\left(\frac{t-kb_0a_0}{a_0^j}\right)$$

dove j e k ϵ Z, a_0>1 è un fissato step di dilatazione e b_0 è un fattore di traslazione che dipende da a_0. Di solito si fissa a_0=2 e b_0=1, in modo da avere un campionamento diadico sia nelle frequenze che nel tempo, quindi:

$$\psi_{j,k}(t) = 2^{-\frac{j}{2}}\psi\left(2^{-j}t-k\right)$$

La DWT può essere scritta come:

$$d_{j,k} = \int_{-\infty}^{+\infty} s(t)2^{-\frac{j}{2}}\psi^*\left(2^{-j}t-k\right)dt$$

dove $d_{j,k}$ sono denominati coefficienti dettagliati alla scala j e alla posizione k. Dai coefficienti dettagliati è possibile ricavare il segnale originale s(t) se è soddisfatta una condizione necessaria e sufficiente, ossia che l'energia dei coefficienti wavelet deve trovarsi fra due limiti positivi, se i due limiti sono uguali, allora le wavelet discrete si comportano come una base ortonormale, questo, inoltre, comporta che venga rimossa totalmente la ridondanza dalla WT. Quindi il segnale originale può essere ricostruito tramite tale formula:

$$s(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{j,k} 2^{-\frac{j}{2}} \psi\left(2^{-j}t - k\right)$$

I coefficienti wavelet che sono stati già nominati più volte all'interno di questo paragrafo, sono elementi fondamentali su cui si basa sia il criterio wavelet del rilevamento predittivo dei guasti, ma anche la rete neurale perché i risultati su cui poi si è concentrato tutto lo studio derivano dalla valutazione di un indice i cui termini principali sono ricavati proprio dai valori dei coefficienti, come verrà spiegato dettagliatamente nei paragrafi successivi.

Questi coefficienti sono di due tipi: coefficienti approssimativi e coefficienti dettagliati, i primi sono le componenti a bassa frequenza del segnale, mentre i secondi sono le componenti a frequenze elevate. Per dividere le due componenti, quelle a bassa frequenza e quella ad alta frequenza, si usano due filtri: un passa-alto e un passa-basso. Se si filtra una sola volta il segnale d'interesse, si ottengono due segnali differenti: uno estratto dal filtraggio passa-alto, che rappresenta i dettagli, ovvero le variazioni rapide; il secondo ottenuto dal filtro passa-basso, che rappresenta le approssimazioni, ovvero l'andamento grossolano. Tuttavia, il segnale risultato del filtraggio passa-basso potrebbe contenere ancora dei dettagli, quindi si può procedere ad un ulteriore fase di filtraggio, ottenendo ulteriori due segnali. Questo processo può essere iterato finché non si è soddisfatti della suddivisione delle bande. Inoltre, nel caso in cui si stiano analizzando dei segnali digitali, oltre al filtraggio, bisogna attuare un'ulteriore operazione, infatti, in questo caso, se si effettuasse soltanto il filtraggio, come risultato si avrebbero il doppio dei campioni rispetto al segnale originale, questo perché ogni segnale risultato dal rispettivo filtraggio ha lo stesso numero di campioni del segnale originale. Per ovviare a questo problema bisogna effettuare un'operazione di down-sampling, che permette di mantenere solo un campione ogni due, mantenendo il contenuto informativo invariato. Questi coefficienti, inoltre, permetto di ricostruire il segnale originale, quindi l'idea della trasformazione wavelet è che, usando un'espansione wavelet, qualsiasi funzione può essere espressa come la somma degli elementi base:

$$s(t) = \sum_{k=-\infty}^{\infty} a_k \varphi(t-k) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} 2^{\frac{j}{2}} \psi\left(2^j t - k\right)$$

dove appunto gli a_k sono i coefficienti di approssimazione, i $d_{j,k}$ sono i coefficienti dettagliati, il parametro k determina la traduzione temporale, ψ(t) è la wavelet madre e φ(t) è chiamata funzione di scala ed è legata alla wavelet madre. Anche questa funzione può essere scalata e traslata a formare una famiglia di funzioni di scala, nel dominio discreto:

$$\varphi_{j,k}(t) = 2^{\frac{j}{2}} \varphi\left(2^j t - k\right)$$

La funzione di scala ha il ruolo di coprire l'intera porzione di spettro lasciata libera dalla wavelet e viene definita a partire da quest'ultima, perché per una wavelet, al crescere del parametro a, che aumenta di volta in volta di un fattore due nel dominio del tempo, in accordo con il campionamento diadico, si ha che la sua larghezza di banda, invece, si dimezza. Questo significa che, teoricamente, ci vorranno un numero infinito di wavelet, per coprire lo spettro del segnale fino a zero, perché, all'aumentare del fattore di scala, si coprirà solo la metà dello spettro rimanente, cosa che viene appunto evitata grazie all'utilizzo della funzione di scala. Infatti, i filtri passa-alto sono associati alle funzioni wavelet, mentre i filtri passa-basso sono associati alle funzioni di scala.

La caratteristica che definisce un sistema wavelet o multi-risoluzione è che ψ(t) soddisfa un'equazione di scala come:

$$\varphi(t) = \sum_{k=-\infty}^{\infty} h[k] \sqrt{2} \varphi(2t-k)$$

per qualche sequenza h[k] che di solito è finita. La funzione wavelet ψ(t) deriva da φ(t). Ogni coefficiente, inoltre, viene calcolato come prodotto interno tra s(t) e il rispettivo elemento base:

$$a[k] = a_k = \int_{-\infty}^{\infty} s(t)\, \varphi(t-k) dt$$

$$d_j[k] = d_{j,k} = \int_{-\infty}^{\infty} s(t)\, 2^{\frac{j}{2}} \psi\left(2^j t - k\right) dt$$

Nel caso di studio la decomposizione wavelet viene implementata usando sia il filtraggio che il down-sampling. Inoltre, nel nostro caso la madre wavelet scelta è "db5", ovvero la madre Wavelet Daubechies, dove 5 è il grado del filtro, questa famiglia di onde è la più adatta al segnale elettrico. La selezione di un wavelet adeguato è fondamentale per identificare le caratteristiche del guasto, la wavelet ottimale, per l'estrazione di un dato segnale, è quella che può generare la maggior parte dei coefficienti con valori massimi all'interno del dominio della scala temporale

per rappresentare la caratteristica del segnale. Tuttavia, i livelli di decomposizione totali necessari per analizzare un segnale completo possono essere calcolati utilizzando la seguente equazione:

$$L \geq \frac{\log \frac{f_s}{50}}{\log 2} + 1$$

dove L è il numero minimo di livelli di decomposizione e f_s la frequenza di campionamento. Queste bande non possono essere cambiate a meno che non venga effettuata una nuova acquisizione con f_s diversi.

Inoltre, nel caso di studio il segnale elettrico è stato acquisito direttamente dal servomotore, tramite il software fornito dal produttore del motore, e prima di applicare la decomposizione wavelet, è stata effettuata una preelaborazione al fine di amplificare le differenze tra segnale difettoso e segnale normale. La fase di preelaborazione riduce il rumore permettendo una migliore interpretazione del segnale originale, accentuando le caratteristiche difettose che si vogliono studiare, consentendo in seguito una migliore estrazione con la decomposizione wavelet.

3.3 METODOLOGIA

Lo scopo di questo lavoro è quello di eseguire un sistema predittivo di rilevamento guasti in grado di capire quando ci si sta spostando da condizioni di sicurezza verso condizioni critiche del sistema di trasmissione a cinghia. Nella fase di raccolta e preelaborazione dei dati c'è stata una valutazione del segnale elettrico del motore, usando il metodo MCSA, acquisendolo in condizioni di sicurezza e quando potrebbe verificarsi l'errore. L'effetto dell'allentamento della cinghia si riflette sulla corrente dello statore del PMSM e quindi sono state estratte le caratteristiche utilizzando l'analisi di decomposizione wavelet. Le ipotetiche condizioni difettose si trovano analizzando sperimentalmente il campo di lavoro del sistema di cinghie, iniziando dal suo valore di pretensionamento. La frequenza naturale delle cinghie è stata misurata con un misuratore di micro-tensione, questo dispositivo è costituito da una piccola testa di rilevamento, che viene tenuta sulla cinghia da misurare. Quindi la cinghia viene fermata per indurla a vibrare alla sua frequenza naturale, le vibrazioni vengono rilevate e la frequenza delle vibrazioni viene visualizzata sullo strumento di misura. Al fine di trovare un intervallo di funzionamento in cui la cinghia lavora in condizioni di sicurezza, è stato utilizzato questo misuratore di micro-tensione per scegliere il giusto pretensionamento, che viene suggerito dal produttore della cinghia considerando la forza periferica (PF). Quando la tensione statica della cinghia scende al di sotto del valore, corrispondente al PF, la tensione non è sufficiente per spostare tutti i carichi

e la cinghia è deformata durante i movimenti del robot, ciò provoca i vari guasti, come ad esempio cinghia rotta o abrasione. A partire da questo vengono identificate sperimentalmente tre diverse condizioni : Condizione di sicurezza, Critico di sicurezza, ovvero prima di raggiungere il valore corrispondente al PF, e Critico, ovvero quando la tensione scende al di sotto del valore della PF.

3.3.1 PROCEDURA DI RILEVAMENTO GUASTO PREDITTIVO

Il criterio wavelet del rilevamento predittivo dei guasti si basa sulla valutazione del seguente indice:

$$W_x = abs\left(\frac{Ed_x}{Ea}\right)$$

dove il termine Ea contiene la somma della percentuale di energia di tutti i livelli dei coefficienti di approssimazione, mentre Ed è un vettore di riga con il numero di colonne uguale al numero dei livelli, ognuno di questi rappresenta i coefficienti dettagliati dal livello uno all'ultimo, e otteniamo per ognuno la singola percentuale di energia. Invece x è il livello in cui l'allentamento della cinghia mostra la sua influenza sul segnale di corrente. Questo indice rafforza le informazioni ottenute dalla decomposizione wavelet e viene utilizzato nel caso di studio come soglia per condizioni sane. Quando aumenta l'allentamento della cinghia, aumenta la percentuale di energia del coefficiente x^{th} della decomposizione. Dal punto di vista fisico la frequenza naturale della cinghia diminuisce in condizioni di allentamento e ciò introduce vibrazioni non previste e oscillazioni di coppia che si riflettono sulla corrente dello statore, quindi si presenta uno stato critico e in questo modo si possono scegliere le soglie entro le quali individuare la condizione in cui sta lavorando il sistema. Quindi in base ai valori di W_x ottenuti utilizzando i segnali elettrici acquisiti nelle varie condizioni si è in grado di distinguere, predire e identificare lo stato del sistema ed eventualmente agire se ci si trova in una situazione critica.
Questa tecnica è stata testata offline su un CR a tre assi.

Figura 3.3 CAD (a sinistra) e foto (a destra) del robot cartesiano

Tutti i test sono stati effettuati sull'asse Z del robot; tutti gli assi hanno installato lo stesso modello PMSM, la frequenza di vibrazione della cinghia è derivata dalla misura della sua tensione statica. Nel nostro caso il pretensionamento corretto misurato su 0,5 m era 64Hz, considerando che il carico massimo trasportato dal CR è di 10 chili. In questo modo è possibile riconoscere tre diversi intervalli di lavoro: Sicurezza, quando il tensionamento della cinghia, misurato su 0,5 m, è compreso tra 64Hz e 45Hz, Critico di sicurezza, quando l'intervallo è compreso tra 44Hz-32Hz, e Critico sotto i 32Hz.

3.3.2 CODICE PER ANALISI DEL MOTORE

I segnali acquisiti sul sistema sono segnali elettrici non-stazionari campionati a 8KHz per 1.25 sec, con 10000 campioni a segnale, calcolati nelle diverse possibili condizioni del sistema. Il profilo inseguito dal motore durante il campionamento è un profilo trapezoidale da 0 a 4830 rpm. Questi segnali sono stati inseriti in dei file xlsx, che verranno utilizzati nel codice per calcolare la Wx. In questi file sono presenti quattro colonne: nella prima è indicato il tempo in cui è estratto il segnale, nella seconda la corrente degli avvolgimenti del motore, nella terza il riferimento della velocità e nella quarta la velocità di feedback.
Di seguito è illustrato e spiegato il codice Matlab utilizzato per l'analisi del motore

```
% Analisi transitorio motore asse Z

[filename1] = uigetfile('*.xlsx');

inizio ='B02';
fine='B10001';
due_punti=':';
range=strcat(inizio,due_punti,fine);
[y1]=xlsread(filename1,range); % inserisce in NUM tutti i dati
% in y1 viene salvato il segnale elettrico acquisito
```

Questa prima parte il codice seleziona il file xlsx dove ci sono inseriti i segnali acquisiti e prende tutti gli elementi della seconda colonna, cioè quelli contenenti i valori della corrente nella matrice y1.

```
n=length(y1);
fr=8000; %input('Frequenza di campionamento (Hz): ');
ts=1/fr;
T=n*ts;          %Tempo di acquisizione    sec
t=(0:ts:T-ts)'; %vettore dei tempi
```

```
% PREPROCESSING: returns the envelopes of x determined using the
magnitude of its analytic signal.
%The analytic signal is computed by filtering x with a Hilbert FIR
filter of length fl.
%This syntax is used if you specify only two arguments.
env1=envelope(y1); %,40,'analytic'); % parameter to change the
envelope structure
```

Questa funzione restituisce gli inviluppi del segnale, determinati utilizzando l'ampiezza del suo segnale analitico, e questo segnale viene calcolato filtrando x con un filtro FIR Hilbert di lunghezza fl.

```
%% analisi del transitorio

figure
plot(t,y1,'r',t,env1,'k');
title(filename1)


%% wavelet decomposition of the signals

[c1,l1] = wavedec(env1,10,'db5'); % c1 coeff, l1 number of coeff,
level 10, 'db5' grado del filtro
[a11] = appcoef(c1,l1,'db5', 1); % se voglio un particolare coeff
di approx
[a19] = appcoef(c1,l1,'db5', 9); % se voglio un particolare coeff
di approx
approx1 = appcoef(c1,l1,'db5',10); %fa un approssimazione più
grossolana dei coefficienti
%extracts the detail coefficients at level N from the wavelet
decomposition structure [C,L]
[cd11,cd12,cd13,cd14,cd15,cd16,cd17,cd18,cd19,cd110] =
detcoef(c1,l1,[1 2 3 4 5 6 7 8 9 10]);
[Ea1,Ed1] = wenergy(c1,l1); % energia del segnale nei diversi
livelli
```

In questa parte di codice c'è la decomposizione wavelet off-line del segnale elettrico. Nella prima riga, C è il vettore dei coefficienti di approssimazione e dei dettagli e L contiene i numeri dei coefficienti. La decomposizione wavelet viene applicata sul segnale pre-elaborato, per 10 livelli, dove db5 indica la wavelet madre. La variabile "approx1" contiene tutti i coefficienti di approssimazione della decomposizione, mentre i coefficienti dettagliati sono memorizzati nei vettori [cd11, ..., cd110]. Nell'ultima riga , viene calcolata l'energia del segnale nelle diverse sotto bande, tramite i termine Ea1 ed Ed1.

```
figure
subplot(4,1,1)
plot(env1)
title('env1')
subplot(4,1,2)
plot(approx1)
title('Approximation Coefficients')
subplot(4,1,3)
plot(cd110)
title('Level 10 Detail Coefficients')
subplot(4,1,4)
plot(cd19)
title('Level 9 Detail Coefficients')

%% Index
%numeri puri percentuale det. coeff. liv 9 e 10 su percentuale
energia
%livello approssimativo Ea

W9_1= abs(Ed1(1,9)/Ea1); % indice non direttamente correlato alla
situazione critica

W10_1= abs(Ed1(1,10)/Ea1); % l'indice W10 cresce più aumenta
l'allentamento della cinghia
% in particolare la situazione critica si verifica quando il
tensionamento
% è inferiore ai 30 Hz
```

Questa è l'ultima parte in cui si calcolano i due coefficienti W.

Capitolo 4

4.1 PRELABORAZIONE DEI DATI PER LA RETE NEURALE

La rete neurale progettata in quest'elaborato si basa sui risultati ottenuti e analizzati nel lavoro descritto nel capitolo precedente, infatti i dati su cui è stata addestrata la rete sono i valori dei due indici W_{9_1} e W_{10_1}, che sono quelli che permettono di studiare lo stato del sistema. L'obiettivo era quello di progettare una rete neurale in grado d'individuare i guasti del sistema di trasmissione dei motori CR e riconoscerli, ovvero essere in grado di classificarli, capendo quindi in quale dei tre stati si trova il motore: stato di sicurezza, critico di sicurezza o stato critico. La rete neurale è stata programmata usando il software Matlab che attraverso il comando nprtool consente

di utilizzare l'applicazione per il riconoscimento e la classificazione di modelli, che permette di creare una rete neurale che serve proprio per la classificazione, l'applicazione inoltre aiuta a selezionare i dati ed a valutare le prestazioni della rete. La prima cosa da fare è quella di inserire i dati di input e i target, infatti per definire un problema di riconoscimento del modello è necessario disporre di due insiemi di vettori: un insieme di vettori di input come colonne in una matrice e uno di vettori target in modo che indichino le classi a cui sono assegnati gli input. Nel nostro caso gli input sono i valori dei due indici W_{9_1} e W_{10_1} che sono stati calcolati utilizzando i segnali acquisiti nel codice Matlab mostrato nel capitolo precedente, quindi gli input formano una matrice 2x49 in cui nella prima riga ci sono tutti i valori di W_{9_1} ottenuti con i diversi segnali e nella seconda riga tutti i valori di W_{10_1}. Invece la matrice dei target dipende dal numero di classi possibili in cui possono essere raggruppati gli input, nel nostro caso gli stati possibili in cui si può trovare il sistema sono tre, quindi i vettori di target costruiscono una matrice che sarà di dimensione 3x49. Inoltre, questi vettori sono tali che per ognuno di essi un elemento è 1 mentre tutti gli altri sono 0, più precisamente si avrà l'unita in corrispondenza della classe a cui appartengo quei dati di input e 0 in tutte le altre.
Una volta inseriti i dati su cui dovrà essere addestrata la rete neurale questi verranno divisi in tre sottoinsiemi: il primo sottoinsieme è quello di addestramento, solitamente contiene circa il 70% dei campioni, il secondo è quello della validazione, che contiene solitamente il 15% dei campioni, e i rimanenti saranno utilizzati per testare la rete. I campioni destinati all'addestramento sono quelli su cui la rete viene regolata, vengono utilizzati per calcolare il gradiente e aggiornare i pesi e le distorsioni della rete. I dati impiegati per la validazione servono per misurare e convalidare la generalizzazione della rete e per interrompere la formazione prima dell'overfitting, ovvero quando la generalizzazione smette di migliorare. Quindi i campioni utilizzati per la validazione servono appunto ad impedire che ci sia overfitting, che è il rischio di sovradattamento, in questo caso della rete, durante il processo di apprendimento, ovvero la rete neurale si adatta troppo bene ai dati di addestramento perdendo in generalità, quindi risulta perfetta per i dati di allenamento, ma quando si verifica il comportamento della rete con i dati di test si registrano molti errori. Più in particolare l'errore sul set di validazione viene monitorato durante il processo di formazione, questo errore normalmente diminuisce durante la fase iniziale dell'addestramento, ma, quando la rete inizia a sovradattare i dati, l'errore sul set di convalida in genere inizia a salire e quindi i pesi della rete vengono salvati al minimo dell'errore del set di validazione. Infine, l'ultimo sottoinsieme è quello che serve per testare la rete che appunto viene utilizzato come test completamente indipendente di generalizzazione della rete.

L’ultima cosa necessaria è quella di impostare il numero di neuroni nel livello nascosto della rete, nel nostro caso si hanno buoni risultati utilizzando quattro livelli nascosti. Una volta completati questi passaggi si può addestrare la rete sui dati inseriti e quella che si ottiene sarà una rete a due livelli, con una funzione di trasferimento sigmoideo nello strato nascosto e una funzione di trasferimento softmax nello strato di output.
Sia la funzione di trasferimento sigmoideo che la funzione di trasferimento softmax sono funzioni logistiche, ovvero una funzione che descrive una curva ad S di crescita di un insieme di elementi, all'inizio la crescita è quasi esponenziale, successivamente rallenta, diventando quasi lineare, per raggiungere una posizione asintotica dove non c'è più crescita. Nel machine learning viene utilizzata perché la regressione logistica, che utilizza un algoritmo per classificare i dati in arrivo in base a dati storici, si avvale proprio di questo tipo di funzioni.
In particolare, la funzione sigmoidea è una funzione il cui output è nell'intervallo [0,1]

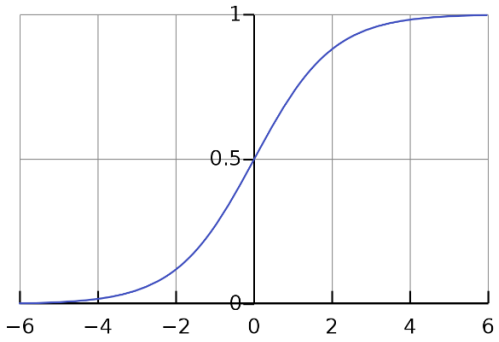


figura 4.1 funzione sigmoidea

ed è esprimibile tramite la seguente equazione:

$$y = \frac{1}{1 + e^{-h}}$$

dove nel caso applicato alle reti neurali y rappresenta l’output del neurone e h la somma degli input del neurone che può essere espressa in questo modo:

$$h = \sum_{k=1}^{n} w_k x_k$$

Risulta utile qualora si desideri trasformare una variabile con valori reali in qualcosa che rappresenta una probabilità quindi trasforma i valori reali arbitrari in numeri compresi tra 0 e 1, maggiore è il valore reale, maggiore sarà vicino a 1 il risultato e viceversa. In molte applicazioni è decisamente opportuno che il centro delle sigmoidi dipenda dal neurone stesso per garantire una maggiore flessibilità di calcolo. Si può ottenere questo modificando l’ultima formula nel seguente modo:

$$h = \sum_{k=1}^{n} w_k x_k - S_k$$

dove S_k è il punto in cui è centrata la sigmoide del k-esimo neurone. Affinché tale soglia sia personalizzata è necessario che essa venga appresa, ovvero si modifichi durante la fase di apprendimento, come i pesi delle connessioni tra i neuroni dei diversi strati. Possiamo considerare tale soglia come un input aggiuntivo costante di valore 1 che è collegato al neurone k con un peso da apprendere. La formula per calcolare l'attivazione del neurone diventa quindi:

$$h = \sum_{k=1}^{n+1} w_k x_k$$

Invece la funzione di trasferimento softmax è una generalizzazione di una funzione logistica che comprime un vettore k-dimensionale di valori reali arbitrari in un vettore k-dimensionale di valori compresi nell'intervallo [0,1]. La sua funzione è la seguente:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{per } j = 1, \dots, K$$

La funzione softmax è spesso usata nello strato finale dei classificatori basati su reti neurali e questa funzione per ogni neurone k esegue una sorta di normalizzazione i cui valori di uscita z_k possono essere interpretati come una probabilità tra 0 e 1

$$z_k = f(v_k) = \frac{e^{v_k}}{\sum_{i=1\dots s} e^{v_i}}$$

4.2 ELABORAZIONE E RISULTATI OTTENUTI

A questo punto è possibile allenare la rete neurale sui dati che le abbiamo fornito, l'addestramento si interrompe automaticamente quando la generalizzazione smette di migliorare, come indicato da un aumento dell'errore di entropia incrociata dei campioni di validazione.

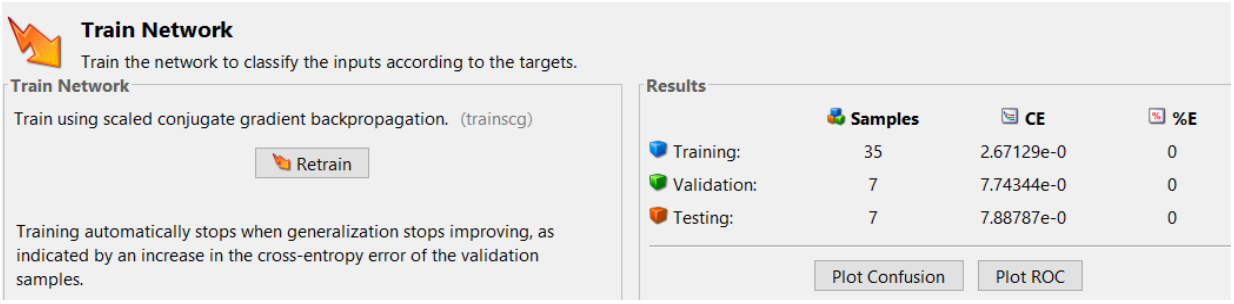


Figura 4.2 rete neurale allenata con errori di entropia incrociata ed errori percentuali sulla destra

Otterremo i valori di entropia incrociata, ovvero una delle possibili funzioni di perdita, in particolare l'entropia incrociata descrive la perdita tra due distribuzioni di probabilità; e i valori dell'errore percentuale, riducendo al minimo l'entropia incrociata si ottiene una buona classificazione. L'errore percentuale indica la frazione di campioni che sono classificati erroneamente, un valore di 0 significa nessuna classificazione errata, 100 indica classificazioni errate massime.
Alla fine, la rete neurale che otteniamo è di questo tipo:

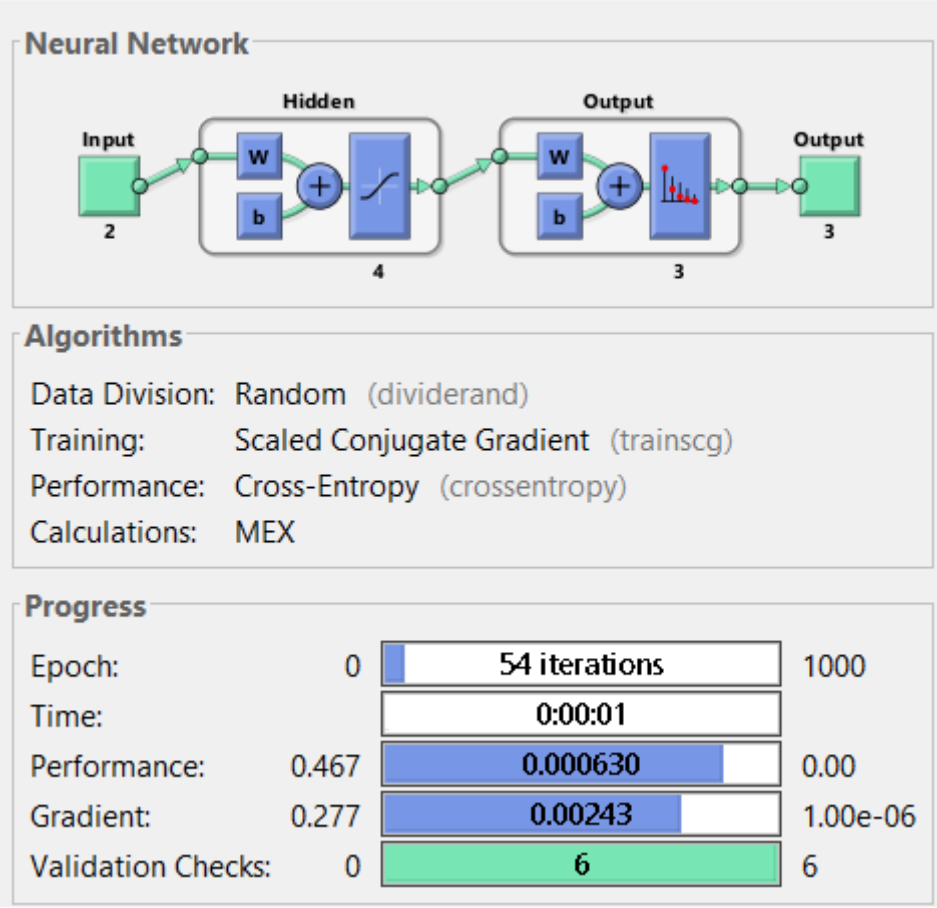


Figura 4.3 la rete neurale ottenuta

con 2 input, uguale al numero di elementi nel vettore di input, 4 neuroni nascosti, che abbiamo selezionato noi e con 3 neuroni di output, uguale al numero di classi possibili, ottenuta dopo 54 iterazioni.
Per valutare le prestazioni della rete neurale Matlab ci permette di vedere la matrice di confusione relativa alla rete che, in questo caso, è la seguente:

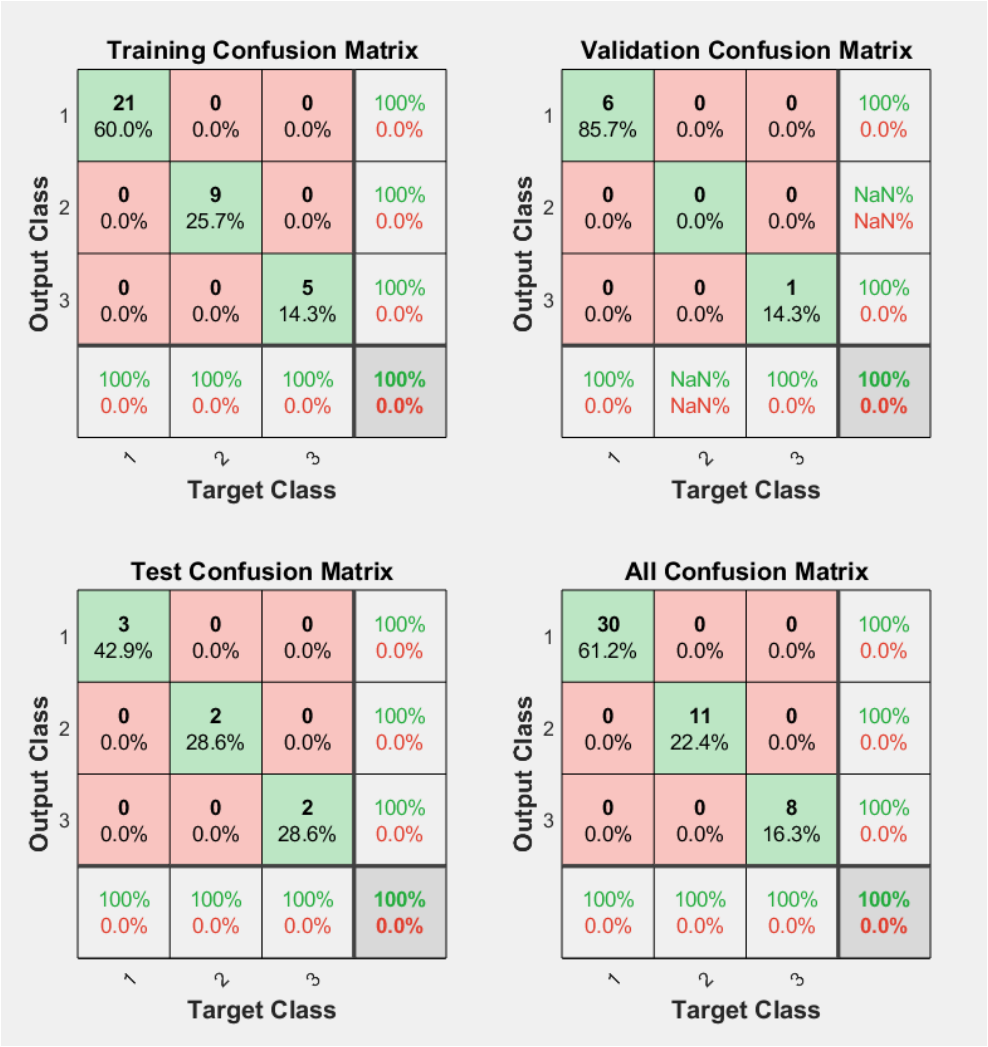


Figura 4.4 matrici di confusione relative alla rete neurale

Abbiamo quindi le matrici di confusione per addestramento, validazione e test e per i tre set di dati combinati. La matrice di confusione, detta anche tabella di errata classificazione, è molto utile perché restituisce una rappresentazione dell'accuratezza di classificazione. Questa matrice, inoltre, permette con facilità di vedere se le previsioni sono più o meno corrette ed è quindi molto semplice comprendere le performance di un modello predittivo di classificazione in modo da determinare quanto questa sia accurata ed efficace. Per analizzare la matrice di confusione in modo da poter capire l'efficienza della rete neurale è necessario definire i seguenti termini:

• Veri positivi (TP, True positive): sono i casi in cui la rete neurale ha predetto che un dato appartenesse ad una classe e ci apparteneva realmente.

• Veri negativi (TN, True negative): sono i casi in cui la rete neurale ha previsto che un dato non appartenesse ad una classe e non ci apparteneva realmente.

• Falsi positivi (FP, False positive): sono i casi in cui la rete ha previsto che un dato appartenesse ad una classe, ma in realtà non ci apparteneva.

• Falsi negativi (FN, False negative): sono i casi in cui la rete ha previsto che un dato non appartenesse ad una classe, anche se in realtà ci apparteneva.
Inoltre, insieme a questi termini è utile dare anche altre definizioni che servono per evidenziare alcune caratteristiche riguardanti le prestazioni della rete neurale, ad esempio:
- Tasso di errore (ERR) che viene calcolato come il numero di tutti i pronostici errati diviso per il numero totale del set di dati:

$$ERR = \frac{FP + FN}{TP + TN + FP + FN}$$

Il miglior tasso di errore è 0, mentre il peggiore è 1.
- Accuratezza che misura la percentuale delle previsioni esatte sul totale delle istanze ed è quindi l'inverso del tasso di errore. Pertanto, la migliore accuratezza è 1, mentre la peggiore è 0. Può essere calcolato dalla seguente formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = 1 - ERR$$

- Sensibilità che è la capacità di un classificatore di trovare tutte le istanze positive, ovvero è la percentuale delle previsioni positive corrette (TP) sul totale delle istanze positive:

$$Sensitivity = \frac{TP}{TP + FN}$$

- Specificità (SP) che è la percentuale delle previsioni negative corrette sul totale delle istanze negative:

$$SP = \frac{TN}{TN + FP}$$

La migliore specificità è 1, mentre la peggiore è 0.
Queste sono solo alcune delle definizioni riguardanti le matrici di confusioni che sono utili da analizzare per comprendere il comportamento della rete. Nella matrice di confusione, però è immediato capire se la rete ha commesso errori oppure no dato che gli elementi che sono stati classificati correttamente sono indicati sulla diagonale della matrice, mentre gli errori di previsione saranno rappresentati da valori esterni alla diagonale. Quindi è evidente che nel nostro caso la rete neurale è perfetta per la classificazione: nell'addestramento, nella validazione e nel test otteniamo il 100% di risultati esatti, con un tasso di errore pari a 0 ed accuratezza, sensitività e specificità uguale a 1.
Inoltre, Matlab permette di visionare anche la curva delle caratteristiche operative del ricevitore (ROC, Receiver Operating Characteristic).

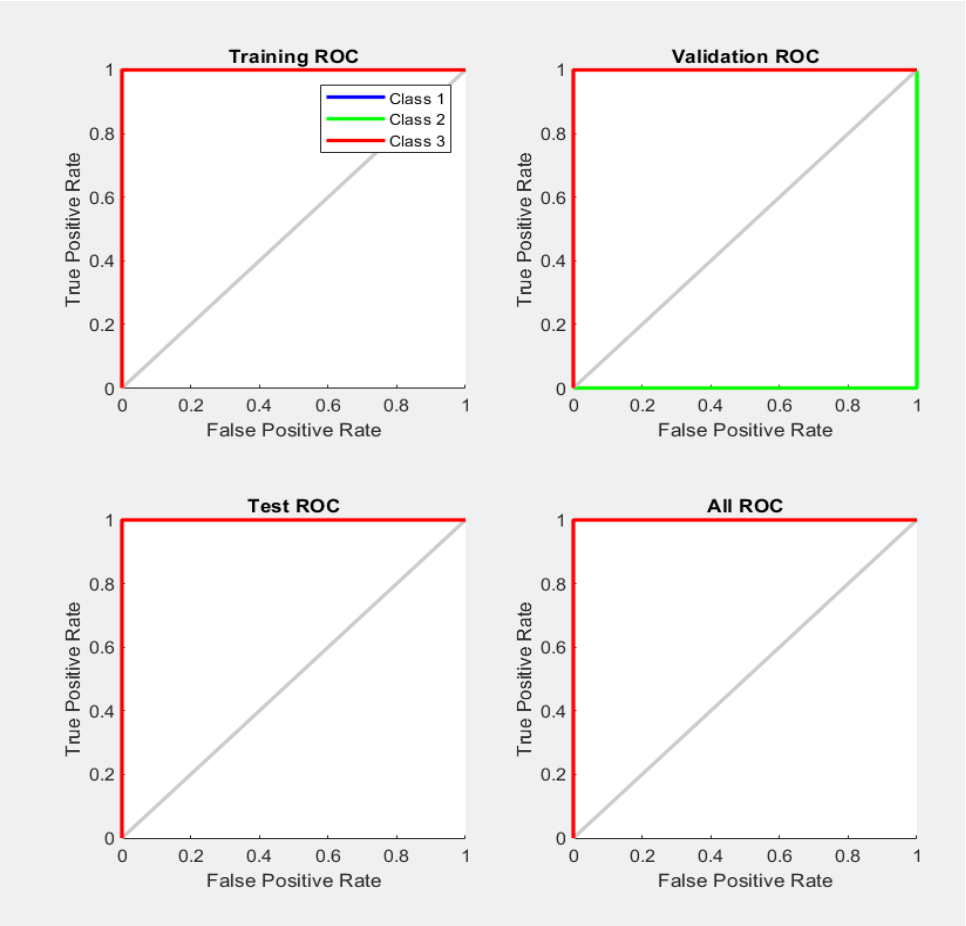


Figura 4.5 curva Roc per i diversi set e per i tre set combinati insieme

La curva ROC è un grafico che viene costruito considerando tutti i valori del test e, per ognuno di questi, si calcola la proporzione di veri positivi, ovvero la sensibilità, e la proporzione di falsi positivi, ottenuta da 1-specificità, congiungendo i punti che mettono in rapporto la proporzione di veri positivi e di falsi positivi si ottiene la curva ROC. Un test perfetto mostrerebbe punti nell'angolo in alto a sinistra, con una sensibilità del 100% e una specificità del 100% e l'area sottostante la curva ROC (Area Under the Curve, AUC), che è una misura di accuratezza avrebbe valore 1, cioè il 100% di accuratezza. Quindi per il nostro problema è confermato che la rete è perfetta, infatti nei grafici le linee rosse, verdi e blu sono sovrapposte e vanno dall'angolo in alto a sinistra fino all'angolo in alto a destra, tranne nel secondo grafico in cui la linea verde si trova nella parte opposta perché per la validazione in questo caso non è stato utilizzato alcun dato appartenente alla seconda classe.
Questi risultati perfetti ovviamente non si ottengono sempre, ma dipendono da i dati che l'applicazione sceglie per l'addestramento, per la validazione e per i test, infatti allenando nuovamente la rete neurale si possono ottenere risultati diversi che però evidenziano comunque delle ottime prestazioni della rete:

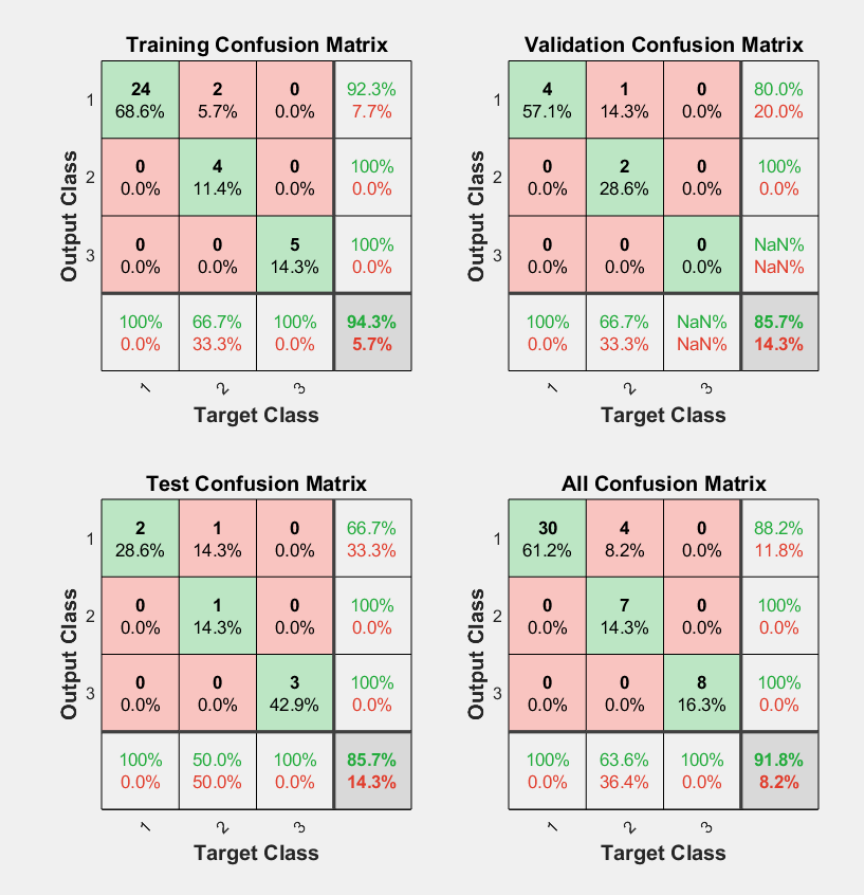


Figura 4.6 matrice di confusione dopo un altro allenamento della rete neurale

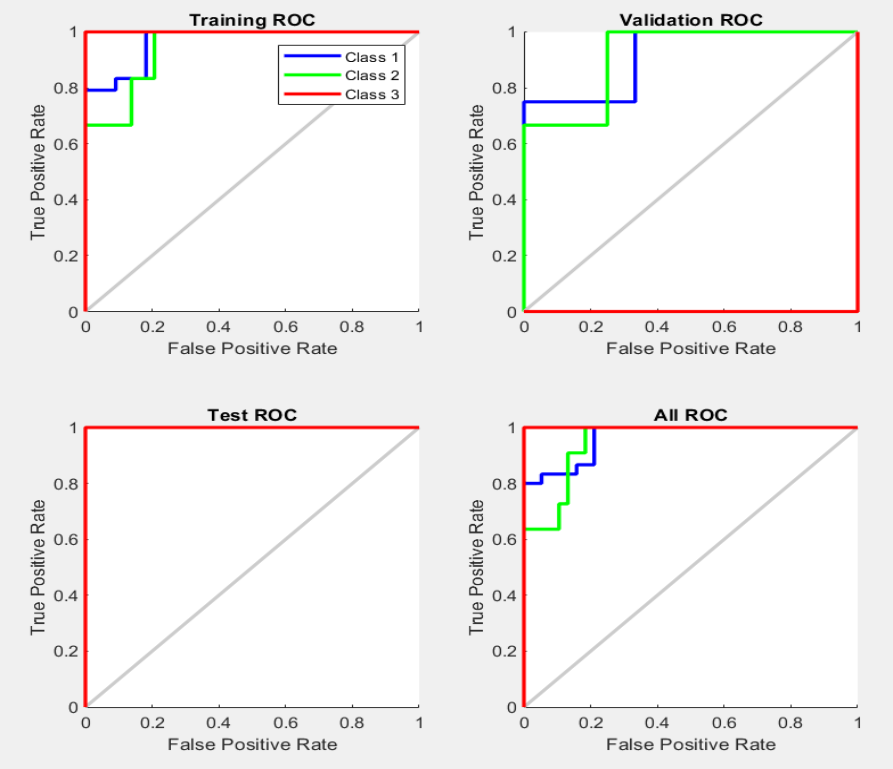


Figura 4.7 curva ROC dopo un altro allenamento della rete neurale

Quindi i risultati ottenuti sono molto spesso perfetti o comunque molto buoni, questo vuol dire che la rete neurale è in grado di individuare correttamente lo stato in cui si trova il sistema composto dal CR.
Per testare nuovi valori con questa rete neurale è sufficiente utilizzare la funzione di Matlab y = net(x), dove x sono gli input, quindi i valori degli indici W_{9_1} e W_{10_1}, e y l'output. Inoltre, nella matrice degli output quando utilizziamo nuovi valori, ma anche per i risultati della rete appena allenata, non avremo un unità e i restanti valori nulli, ma dei valori prossimi allo zero e dei valori prossimi all'unità e tanto più questi valori saranno vicini al valore unitario tanto più apparterranno a quella classe.

4.3 CODICE DELLA RETE NEURALE

Il codice relativo alla rete neurale generata è il seguente:

```
% Solve a Pattern Recognition Problem with a Neural Network

x = W;
t = targets;
```

x rappresenta la matrice contenente i dati di input, quindi i valori degli indici W_{9_1} e W_{10_1}, mentre t rappresenta la matrice contenente i targets, ovvero la classificazione degli input.

```
%trainFcn = 'trainbr'
%trainFcn = 'trainlm'
trainFcn = 'trainscg';  % Scaled conjugate gradient
backpropagation.
```

Questa è la funzione di addestramento è possibile usare una delle tre proposte per allenare la rete:
- la prima ('trainbr') richiede più tempo ma può essere migliore per problemi impegnativi.
- la seconda ('trainlm') è solitamente il più veloce.
- la terza ('trainscg') utilizza meno memoria.

```
% Create a Pattern Recognition Network
hiddenLayerSize = 4;
net = patternnet(hiddenLayerSize, trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
```

```
% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);
```

vengono divisi i dati in set di allenamento (70%), di valutazione (15%) e di test (15%), successivamente la rete viene allenata e vengono testati i risultati, y è l'output ottenuto, e indica l'errore dato dalla differenza tra t e y, vengono indicati anche il valore della performance della rete e la percentuale d'errore.

```
View the Network
view(net)

figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotconfusion(t,y)
figure, plotroc(t,y)
```

Queste funzioni infine permettono di visualizzare la rete neurale e i vari grafici che ne descrivono le caratteristiche come la matrice di confusione e la curva ROC.

Capitolo 5

Risultati simili a quelli ottenuti con la rete neurale possono essere raggiunti anche con le tecniche di data driven, l'obiettivo di questo capitolo è proprio quello di analizzare e commentare alcune tra i principali metodi di data driven, confrontarli con la rete neurale progettata evidenziando le differenze e gli aspetti in cui sono migliori, considerando il caso generale, ma soprattutto considerando lo stesso contesto analizzato per la creazione della rete. Quindi queste tecniche verranno studiate mettendo in evidenza la loro capacità di classificazione dei guasti.

5.1 SUPPORT VECTOR MACHINE (SVM)

Le support vector machines (SVM) o macchine a vettori di supporto sono dei modelli associati ad algoritmi di apprendimento per la regressione e la classificazione. L'algoritmo SVM ottiene la massima efficacia nei problemi di classificazione binari, ma viene utilizzato anche per problemi di classificazione multiclasse. L'SVM è basato sull'idea di trovare un iperpiano che divida al meglio un set di dati in due classi, dove i support vector, ovvero i vettori di supporto, sono i punti più vicini all'iperpiano, tali punti dipendono dal set di dati che si sta analizzando e, se vengono rimossi o modificati, alterano la posizione dell'iperpiano divisorio. Quindi dato un insieme di esempi per l'addestramento, ognuno dei quali etichettato con la classe di appartenenza fra le possibili classi, un algoritmo di addestramento per le SVM costruisce un modello che assegna i nuovi esempi a una delle classi. Un modello SVM è una rappresentazione degli esempi come punti nello spazio, divisi in modo tale che gli esempi appartenenti alle diverse categorie siano chiaramente separati da uno spazio il più possibile ampio. I nuovi esempi sono quindi mappati nello stesso spazio e la predizione della categoria alla quale appartengono viene fatta sulla base del lato nel quale ricadono.

5.1.1 SVM LINEARI PER LA CLASSIFICAZIONE BINARIA

Per studiare l'algoritmo SVM è necessario analizzare per prima il caso in cui si deve effettuare una classificazione binaria, quindi nel caso in cui si lavora soltanto con due classi, perché è da questo che deriva anche la classificazione multiclasse. Esamineremo prima il caso lineare in cui l'obiettivo del modello è quello di cercare un iperpiano linearmente separabile che separa i valori di una classe dall'altro, se ne esiste più di uno, di cercare quello in cui la distanza tra i vettori di supporto delle due classi differenti è maggiore, per migliorare l'accuratezza del modello.

Data set separabile

Nel caso in cui abbiamo a che fare con un set di dati in cui è facilmente possibile individuare un iperpiano che distingue le due classi il problema è trovare quale tra i possibili iperpiani sia quello ottimale ossia quello che generi il minimo errore di classificazione su una nuova osservazione. Pertanto, desideriamo che i nostri punti dati siano il più lontano possibile dall'iperpiano, pur rimanendo nella parte corretta.

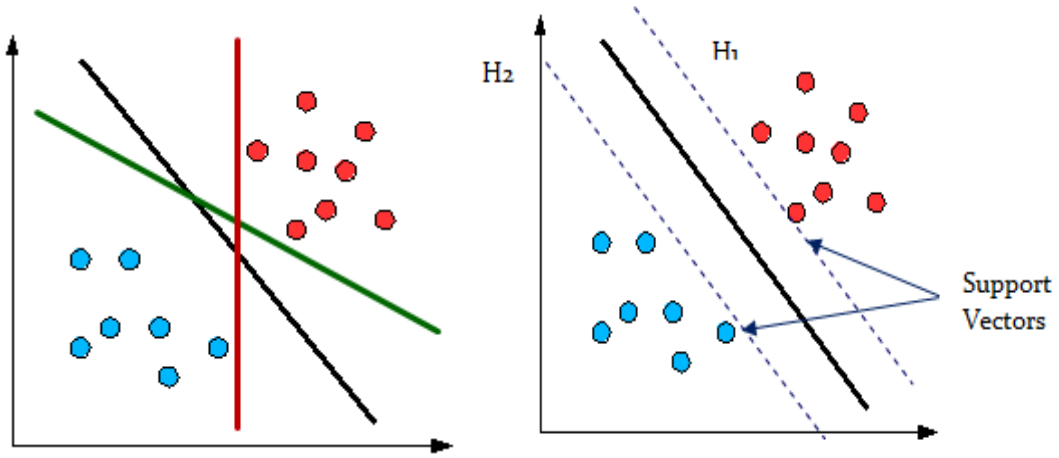


Figura 5.1 possibili iperpiani di separazione fra due classi (a sinistra), iperpiano di separazione ottimale (a destra)

Supponiamo di avere un insieme di N dati separabili con cui vogliamo addestrare la SVM e di voler effettuare una classificazione di tipo binario, cioè con sole due classi possibili. Questi dati di training sono etichettati come:

$$\{x_n, y_n\};\ n = 1, \dots, N;\ \ y_n \in \{-1,1\};\ x_n \in R^D$$

dove y_n è un'etichetta che può assumere uno dei due valori previsti per l'identificazione della classe di appartenenza, quindi positivo o negativo, mentre x_n è il vettore di caratteristiche di input. Nel caso binario, abbiamo dati con cui effettueremo il training, che possono essere positivi o negativi, e l'iperpiano di separazione ottimale che stiamo cercando, ovvero quello che divida nel miglior modo possibile le due classi, soddisfa la seguente equazione:

$$x_n \cdot w + w_0 = 0$$

dove w è il vettore di peso e w_0 è il bias. In maniera più dettagliata può essere scritto in questo modo:

$$w_0 + w_1 x_1 + \cdots + w_n x_n = 0$$

Quindi in n dimensioni un iperpiano di separazione è una combinazione lineare di tutte le dimensioni uguagliate a 0. Si ha allora che i punti che stanno sopra l'iperpiano, rappresentano una classe, mentre i punti che stanno sotto rappresentano l'altra classe. Inoltre, se includiamo in queste condizioni anche i limiti dei margini delle classi è possibile regolare i coefficienti e i pesi in modo tale da avere:

$$x_n \cdot w + w_0 \geq 1,\ \text{per } y_n\text{=1}$$
$$x_n \cdot w + w_0 \leq -1,\ \text{per } y_n\text{=-1}$$

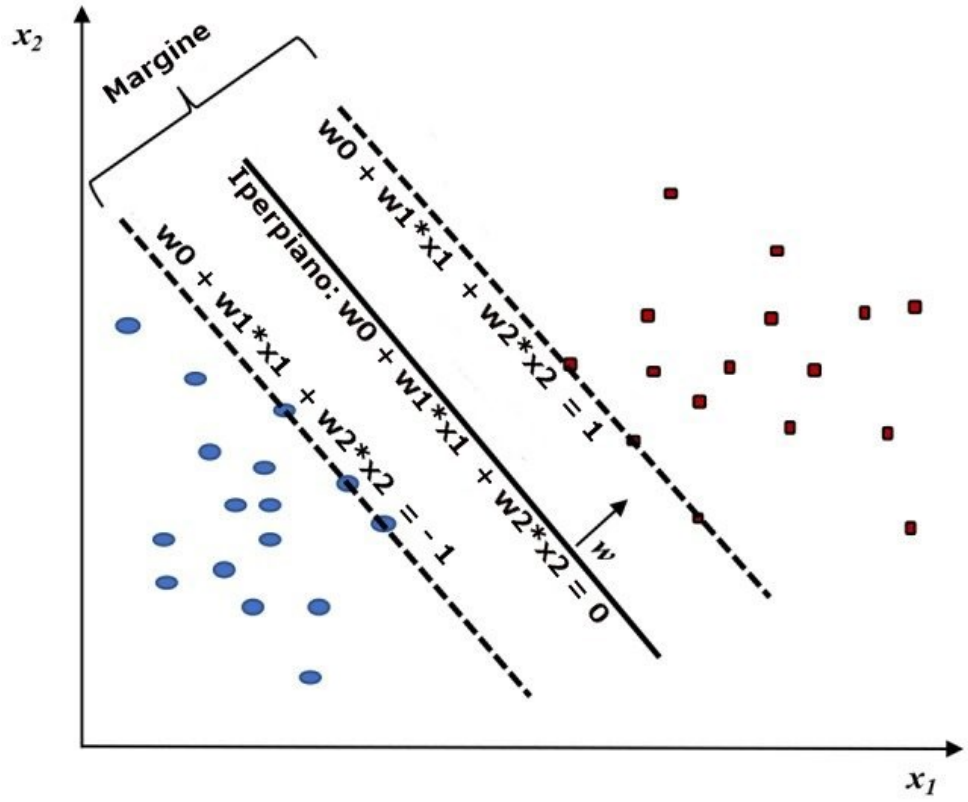


Figura 5.2 iperpiano di separazione fra due classi per un caso in due dimensioni

Questi due risultati rappresentano i confini del margine ed è possibile combinarli in un'unica diseguaglianza:

$$y_n(x_n \cdot w + w_0) - 1 \geq 0$$

Per poter trovare una formula per calcolare i pesi per ottenere l'iperpiano ottimale è necessario per prima cosa determinare la distanza tra x_n e il piano $w \cdot x + w_0 = 0$. Per trovare questa distanza è di fondamentale importanza considerare che il vettore w è perpendicolare al piano nello spazio dei dati di ingresso. Dimostrare questo è molto semplice: prendiamo due punti qualsiasi x' e x'' sull'iperpiano separatore, sappiamo che questi punti verificano le equazioni: $w \cdot x' + w_0 = 0$ e $w \cdot x'' + w_0 = 0$, questo implica che $w \cdot (x' - x'') + w_0 = 0$. Sappiamo che $x' - x''$ è un vettore che congiunge i due punti e che, essendo nullo il suo prodotto scalare con w, tale vettore deve essere per forza perpendicolare a w. Dal momento che ciò vale qualsiasi siano i punti scelti sul piano, è dimostrato che w è ortogonale a qualsiasi vettore sul piano e quindi al piano stesso.

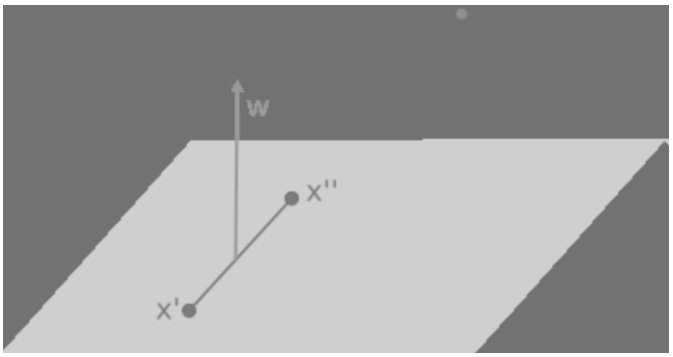


Figura 5.3 w è perpendicolare a qualsiasi vettore sul piano

Per trovare la distanza di un punto x_n dal piano, scegliamo un qualsiasi punto x appartenente al piano e cerchiamo la proiezione di $x_n - x$ su w. Per trovare tale proiezione moltiplichiamo il versore di w, ovvero $\widehat{w} = \frac{w}{\|w\|}$, per il vettore $x_n - x$:

$$distanza = \widehat{w} \cdot (x_n - x) = \frac{w}{\|w\|}(x_n - x) = \frac{1}{\|w\|}|w \cdot x_n - w \cdot x| = \frac{1}{\|w\|}$$

Una volta che conosciamo come calcolare la distanza tra un punto e il piano possiamo proseguire per determinare il valore della larghezza del margine. Consideriamo i punti che verificano la sola eguaglianza $x_n \cdot w + w_0 \geq 1$, questi sono posizionati sull'iperpiano $H_1: x_n \cdot w + w_0 = 1$ mentre, in modo simile, i punti che verificano l'eguaglianza $x_n \cdot w + w_0 \leq -1$, sono posti sull'iperpiano $H_2: x_n \cdot w + w_0 = -1$. Gli altri punti si troveranno nello spazio esterno a quello delimitato dai piani H_1 e H_2 andando a verificare la diseguaglianza stretta $y_n(x_n \cdot w + w_0) - 1 > 0$. Nessun punto si potrà trovare, invece, nello spazio interno a quello delimitato dai piani marginali. Quindi la distanza tra l'iperpiano separatore e l'iperpiano H_1 è la stessa che vi è tra l'iperpiano separatore e l'iperpiano H_2 e vale $\frac{1}{\|w\|}$, la larghezza del margine è banalmente la somma di queste due distanze, ossia $\frac{2}{\|w\|}$. Tenendo presente che gli iperpiani H_1 e H_2 sono paralleli, avendo entrambi come vettore ortogonale lo stesso vettore w, iniziamo a considerare l'obiettivo di individuare i due iperpiani che danno il margine maggiore, minimizzando $\|w^2\|$ sotto la condizione $y_n(x_n \cdot w + w_0) - 1 \geq 0$. Come abbiamo visto, i punti che verificano questa uguaglianza sono quelli che si trovano esattamente sopra uno degli iperpiani che determinano il margine. La massimizzazione del margine deve avvenire nel rispetto della diseguaglianza, per cui si tratta di

$$\text{massimizzare } \frac{1}{\|w\|} \text{ con } y_n(x_n \cdot w + w_0) - 1 \geq 0$$

Possiamo vedere questo problema anche come un problema di minimizzazione. Si rende massimo il margine minimizzando il denominatore. Il problema diventa allora equivalente a:

$$\text{minimizzare } \frac{1}{2}\|w^2\| \text{ con } y_n(x_n \cdot w + w_0) - 1 \geq 0$$

dove il termine $\frac{1}{2}$ è stato aggiunto per semplificare i calcoli, e permetterà poi di risolvere il problema come un problema di programmazione quadratica. Giunti a questo punto, per proseguire consideriamo la formulazione Lagrangiana del problema, che facilita la risoluzione infatti:

• le condizioni presenti nel problema di minimizzazione, verranno sostituite da condizioni sui moltiplicatori Lagrangiani stessi, che sono molto più facili da gestire;

• in questa riformulazione, i dati di training appariranno nella forma di prodotti scalari tra vettori.

Passando alla formulazione di Lagrange otteniamo:

$$L_P(w, b, \alpha) = \frac{1}{2}\|w^2\| - \sum_{n=1}^{N} \alpha_n \left(y_n(x_n \cdot w + w_0) - 1\right)$$

che va minimizzata rispetto a w e b, ricordando che deve valere sempre la condizione $\alpha_n > 0$ e massimizzata rispetto a α. Come si vede dalla formula che vogliamo minimizzare, abbiamo introdotto i moltiplicatori di Lagrange α_n con n = 1, … ,N, uno per ogni condizione, che devono essere tutti positivi. La regola per la formulazione del problema di Lagrange prevede che le equazioni della condizione vengano moltiplicate per dei moltiplicatori di Lagrange positivi e sottratte dalla funzione obiettivo, per formare così la Lagrangiana. Si tratta di un problema di programmazione quadratica convessa, essendo convessa la funzione che vogliamo minimizzare ed essendo convesso anche l'insieme dei punti che verificano le condizioni. Ciò significa che possiamo risolvere il problema duale: massimizzare L_P sotto la condizione che le derivate parziali di L_P rispetto a w e b siano nulle e inoltre i vari α_n siano tutti positivi, ottenendo lo stesso risultato, questa formulazione duale del problema è chiamato Problema duale di Wolfe. Porre le derivate parziali uguali a zero, permette di ottenere:

$$\nabla_w L_P = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \rightarrow w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

$$\frac{\partial L_p}{\partial w_0} = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

che essendo condizioni di eguaglianza che devono essere verificate nella formulazione duale, possono essere sostituite in LP per ottenere:

$$L_D(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n \cdot x_m$$

L_D va massimizzata rispetto alla sola variabile $\alpha = \alpha_1, \dots, \alpha_n$ sotto le condizioni $\alpha_n \geq 0$ e $\sum_{n=1}^{N} \alpha_n y_n = 0$ per n=1, … ,N. Si nota immediatamente che la forma duale richiede solo il calcolo del prodotto di ogni vettore di ingresso. L'allenamento della SVM consiste nel massimizzare L_D rispetto ai moltiplicatori α_n, sotto le condizioni di positività degli α_n e la condizione $\sum_{n=1}^{N} \alpha_n y_n = 0$. La risoluzione del problema di programmazione quadratica fornisce come risultato il vettore dei moltiplicatori di Lagrange $\alpha = \alpha_1, \dots, \alpha_n$ che sostituito nella formula $w = \sum_{n=1}^{N} \alpha_n y_n x_n$ permette di ricavare la soluzione.
Se consideriamo adesso la seguente condizione:

$$\alpha_n(y_n(x_n \cdot w + w_0) - 1) = 0$$

Notiamo che i casi affinché si annulli tale prodotto sono due:
• $(y_n(x_n \cdot w + w_0) - 1) = 0$;
• $\alpha_n = 0$;
Il primo caso si verifica quando il dato x_n si trova su H_1 o H_2, ossia su uno dei due iperpiani di margine. Nel caso di punti interni, invece, l'unico modo perché si annulli tale prodotto è che i moltiplicatori di Lagrange rispettivi a questi dati siano nulli. A questo punto possiamo affermare che se $\alpha_n \geq 0$ allora x_n è un vettore di supporto (SV). Da tutto ciò possiamo capire che i support vector sono di primaria importanza per queste macchine in quanto la soluzione del problema di training e la determinazione degli iperpiani dipende esclusivamente da essi. Se eliminassimo tutti gli altri vettori per poi ripetere nuovamente la fase di training otterremmo come risultato lo stesso iperpiano. Adesso per il calcolo del vettore w possiamo usare la seguente formulazione:

$$w = \sum \alpha_m y_m x_m$$

dove x_m è un SV dal momento che, nella formulazione precedente, il valore di α era 0 per vettori x che non fossero support vector. Giunti a questo punto possiamo calcolare il valore del termine w_0, inserendo il vettore w appena calcolato nella formula:

$$y_s(w \cdot x_s + w_0) = 1$$

Sostituendo, otteniamo:

$$y_s\left(\sum \alpha_m y_m x_m \cdot x_s + w_0\right) = 1$$

Moltiplicando entrambi i membri per y_s e considerando $y_s^2 = 1$ si ottiene:

$$w_0 = y_s - \sum \alpha_m y_m x_m \cdot x_s$$

Invece di usare un vettore di supporto arbitrario x_s, è consigliato usare una media dei valori di w_0 calcolati a partire da tutti i support vector, ossia:

$$w_0 = \frac{1}{N_s}\sum\left(y_s - \sum \alpha_m y_m x_m \cdot x_s\right)$$

Abbiamo così ottenuto i valori di w e w_0 che definiscono l'iperpiano di separazione ottimale della Support Vector Machine.

Data set non separabili

Il problema che ci si pone nel caso di dati non separabili è quello di estendere tutto ciò che si è ottenuto a questa nuova situazione. Se infatti applicassimo l'algoritmo per dati separabili a un caso con dati non separabili, non troveremmo alcuna soluzione.

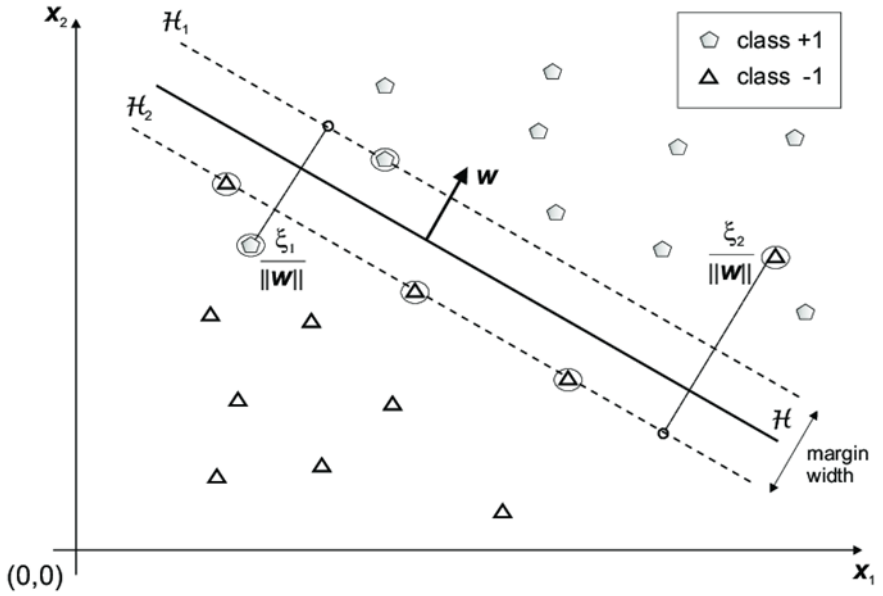


Figura 5.4 set di dati non separabili

Nell'esempio in figura 5.4 il margine viene violato da un dato che però viene comunque classificato correttamente. La condizione che era sempre vera nel caso esaminato in precedenza, ossia $y_n(x_n \cdot w + w_0) \geq 1$, fallisce. Dobbiamo allora introdurre un metodo per quantificare l'errore che si ha nella violazione del margine. Una prima idea è quella di rendere più flessibili le condizioni viste nel caso separabile, introducendo delle variabili positive dette variabili di slack (o allentamento). In questo modo saranno permesse le scorrette classificazioni dei punti. Le condizioni diventano:

$$x_n \cdot w + w_0 \geq 1 - \varepsilon_n,\ per\ y_n{=}1$$

$$x_n \cdot w + w_0 \leq -1 + \varepsilon_n,\ per\ y_n{=}{-}1$$

dove $\varepsilon_n \geq 0$ con n=1, …, N, che possono anche in questo caso essere riassunte in un'unica condizione:

$$y_n(x_n \cdot w + w_0) \geq 1 - \varepsilon_n$$

Se un generico dato x_i viene classificato in modo errato, allora ad esso corrisponde uno slack $\varepsilon > 1$ in quanto per essere classificato erroneamente deve trovarsi per forza al di là dell'iperpiano separatore. Possiamo quindi considerare la violazione totale

come $\sum_n \varepsilon_n$. Ci troviamo davanti a un nuovo problema di ottimizzazione, per molti versi simile a quello visto nel caso di dati separabili:

$$\text{minimizzare } \frac{1}{2}\|w^2\| + C(\sum_{n=1}^{N} \varepsilon_n)^k$$

dove il valore di C può essere scelto e rappresenta il peso che si vuole attribuire agli errori, a un grande valore di tale parametro corrisponde un alta penalità di errore e sarà quindi preferibile avere il minor errore possibile. Si tratta ancora una volta di un problema di programmazione convessa per qualsiasi intero positivo k. Per k = 2 e k = 1 è inoltre un problema di programmazione quadratica e la scelta di k = 1 ha il vantaggio che né ε_n, né i rispettivi moltiplicatori di Lagrange, appaiono nel problema duale di Wolfe, che diventa un problema di massimizzazione. Il problema di minimizzazione va trasformato in un problema di Lagrange, ottenendo così il problema primale:

$$L_P = \frac{1}{2}\|w^2\| - C\sum_{n=1}^{N} \varepsilon_n - \sum \alpha_n \left(y_n(x_n \cdot w + w_0) - 1 + \varepsilon_n\right) - \sum_{n=1}^{N} \beta_n \varepsilon_n$$

Tale funzione va minimizzata rispetto a w, β e ε e massimizzata rispetto ad ogni $\alpha_n \geq 0$ e $\beta_n \geq 0$, dove i coefficienti α_n e β_n sono moltiplicatori di Lagrange. Differenziando questa funzione rispetto a w, β e ε_n e ponendo queste derivate uguali a 0, otteniamo:

$$\nabla_w L_P = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \rightarrow w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

$$\frac{\partial L_p}{\partial w_0} = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\frac{\partial L_p}{\partial \varepsilon_n} = C - \alpha_n - \beta_n = 0 \rightarrow C = \alpha_n + \beta_n$$

a cui vanno aggiunte anche le seguenti condizioni:

$$y_n(x_n \cdot w + w_0) \geq 1 - \varepsilon_n \geq 0$$
$$\varepsilon_n \geq 0\ , \alpha_n \geq 0, \beta_n \geq 0, \beta_n \varepsilon_n = 0$$
$$\alpha_n[y_n(x_n \cdot w + w_0) - 1 + \varepsilon_n] = 0$$

Sostituendo queste nella formula di L_P otteniamo il problema duale di Wolfe, esprimibile come:

$$\text{massimizzare } L_D(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n \cdot x_m$$

rispetto a α, con $0 \leq \alpha_n \leq C$ per n=1, … ,N e $\sum_{n=1}^{N} \alpha_n y_n = 0$, la condizione $0 \leq \alpha_n \leq C$ è necessaria perché altrimenti, affinché sia verificata la condizione $C - \alpha_n - \beta_n = 0$ servirebbe un valore di β_n negativo ma ciò non è possibile dovendo essere sempre $\beta_n \geq 0$. La soluzione di questo problema è data da:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

che può essere ristretta alla sola sommatoria dei vettori di supporto, come visto in precedenza. Osservando i dati del problema notiamo che l'unica differenza con il caso di dati linearmente separabili è che i moltiplicatori di Lagrange α_n sono limitati anche superiormente dalla costante C. L'equazione $C = \alpha_n + \beta_n$ combinata con $\beta_n \varepsilon_n = 0$ mostra che $\varepsilon_n = 0$ se $\alpha_n \leq C$. Possiamo quindi prendere un qualsiasi punto per cui valga $0 \leq \alpha_n \leq C$ per usare $\alpha_n[y_n(x_n \cdot w + w_0) - 1 + \varepsilon_n] = 0$, con $\varepsilon_n = 0$, e calcolare w_0. Come valeva per il caso separabile, è conveniente usare la media di tutti i w_0 calcolati tramite questa equazione.

5.1.2 SVM NON LINEARI PER LA CLASSIFICAZIONE BINARIA

Abbiamo considerato finora solo il caso di funzioni di decisione lineari, ma è utile generalizzare quanto ottenuto al fine di risolvere il problema di ottimizzazione anche nel caso in cui la funzione di decisione non sia funzione lineare dei dati. Il problema di base è che l'insieme dei dati di training non è separabile e che vogliamo trovare un metodo più performante rispetto a quello lineare con slack per suddividere le due classi. Per ovviare a questo problema possiamo utilizzare il metodo del kernel, che ci consente di modellare modelli non lineari di dimensioni superiori.

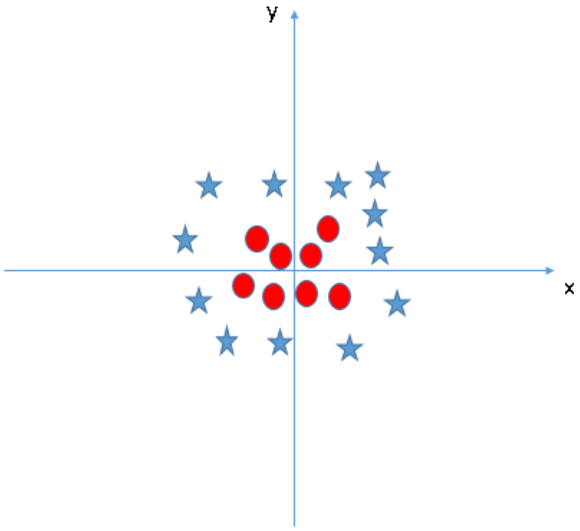


Figura 5.5 set di dati non possono essere separati con un iperpiano lineare

Quindi per risolvere il problema di non linearità dei dati è possibile aggiungere una terza dimensione, fino ad ora abbiamo avuto due dimensioni: x e y, creiamo una nuova dimensione z, e stabiliamo che venga calcolata in questo modo: $z = x^2 + y^2$. La terza dimensione ci darà uno spazio tridimensionale.

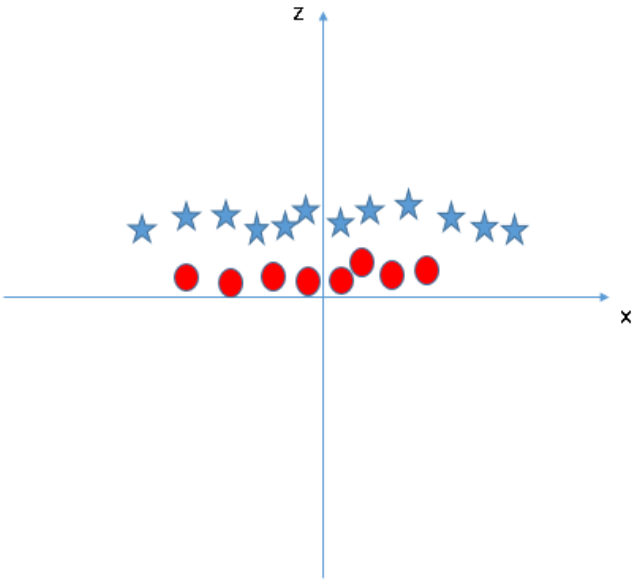


Figura 5.6 set di dati in tre dimensioni

Si noti che poiché ora siamo in tre dimensioni, l'iperpiano è un piano parallelo all'asse x a una certa quota z. Quando guardiamo l'iperpiano nello spazio di input originale, esso sembra un cerchio.

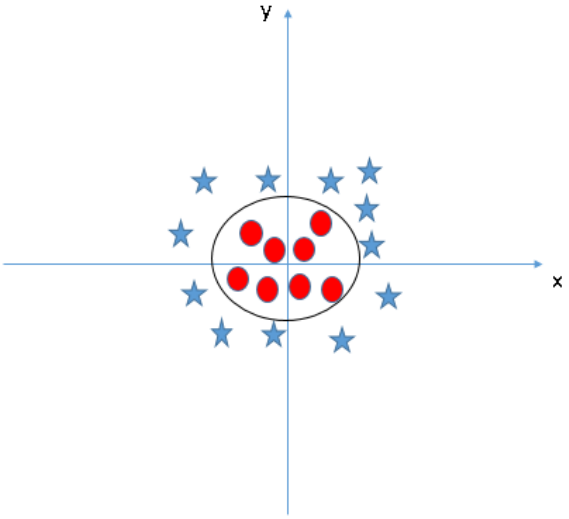


Figura 5.7 set di dati non lineari separati dall'iperpiano

Quindi abbiamo trovato un modo per classificare i dati non lineari mappando il nostro spazio a una dimensione tridimensionale, utilizzando il metodo Kernel.
Gli algoritmi SVM utilizzano un insieme di funzioni matematiche definite come kernel. Il suo scopo è di prendere i dati come input e trasformarli nella forma richiesta qualora non sia possibile determinare un iperpiano linearmente separabile, come avviene nella maggior parte dei casi. In generale il Kernel può essere definito come:

$$K(x, y) = < f(x), f(y) >$$

dove K è la funzione del kernel, x, y sono vettori di input a dimensione n, f è usato per mappare l'input dallo spazio n dimensionale a quello m dimensionale, di livello più

alto rispetto a quello di n. Mentre <x, y> indica il prodotto scalare. Applicare la funzione Kernel all'esempio visto nel paragrafo precedente significa calcolare il prodotto scalare per la terza dimensione (z). Se il nuovo spazio che vogliamo è $z = x^2 + y^2$, il prodotto scalare in tale spazio risulterà come dalle seguenti espressioni:

$$x \cdot y = x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2$$
$$x \cdot y = x_1 \cdot x_2 + y_1 \cdot y_2 + (x_1^2 + y_1^2) \cdot (x_2^2 + y_2^2)$$

Normalmente, il kernel è lineare e otteniamo un classificatore lineare. Tuttavia, usando un kernel non lineare possiamo ottenere un classificatore non lineare senza trasformare completamente i dati: modifichiamo solo il prodotto scalare a quello dello spazio che vogliamo e l'algoritmo SVM ci aiuterà a trovare il miglior iperpiano. La scelta della funzione del kernel tra gli altri fattori potrebbe influire notevolmente sulle prestazioni di un modello SVM, ma, l'unico modo per scegliere il miglior kernel per uno specifico problema di riconoscimento di pattern è attraverso le prove, infatti, a seconda della natura del problema, è possibile che un kernel sia migliore degli altri. Le funzioni Kernel più diffuse sono le seguenti:

• Kernel lineare, il tipo di kernel più semplice, che è definito come:

$$K(x_i, y_j) = x_i \cdot y_j$$

• kernel polinomiale, che è definito come:

$$K(x_i, y_j) = (x_i \cdot y_j + c)^d$$

Questo kernel contiene due parametri: una costante c e un grado di libertà d. Un valore d con 1 rappresenta il kernel lineare. Un valore maggiore di d renderà il limite decisionale più complesso e potrebbe comportare anche un overfitting dei dati.

• kernel RBF(Radial Basis Function), definito come:

$$K(x_i, y_j) = e^{(-\gamma \|x_i - y_j\|^2)}$$

Il kernel RBF è anche chiamato il kernel gaussiano, risulterà in un limite decisionale più complesso. Il kernel RBF contiene un parametro γ: un piccolo valore di γ farà sì che il modello si comporti come un SVM lineare, mentre un grande valore di γ renderà il modello fortemente influenzato dagli esempi dei vettori di supporto.

Quindi in questo caso per trovare l'iperpiano ottimale, dato che nello spazio di ingresso i dati non sono linearmente separabili, supponiamo di lavorare su uno spazio diverso, in cui siano linearmente separabili. Chiameremo questo spazio Z e la sua dimensione può anche essere infinita.

Andiamo a modificare il problema di Lagrange ottenuto per dati separabili e adattiamolo al nuovo caso considerato, ossia all'operare nello spazio Z:

$$L_D(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n \cdot x_m$$

dove $0 \leq \alpha_n \leq C$ per n=1, … ,N e $\sum_{n=1}^{N} \alpha_n y_n = 0$. A questo punto possiamo definire la funzione che restituirà la classe di appartenenza:

$$g(x) = sign(w \cdot z + w_0)$$

dove i valori di w e w_0 possono essere facilmente calcolati:

$$w = \sum \alpha_n y_n z_n$$

$$y_m(w \cdot z_m + w_0) = 1$$

Come si può notare, nel calcolo di g(x) abbiamo bisogno del prodotto scalare $z_n \cdot z$, mentre nel calcolo di w_0 abbiamo bisogno del prodotto scalare $z_n \cdot z_m$. Per calcolare questo prodotto scalare basta mappare i dati dallo spazio X a un altro spazio euclideo Z, usando una funzione che chiamiamo ϕ:

$$\phi: R^D \rightarrow Z$$

Ovviamente, in questa situazione, l'algoritmo di training dipende solo dai prodotti interni in Z, che sono: $z \cdot z' = \phi(x) \cdot \phi(x')$.

Inoltre, dati i punti x,x'ϵ X ,vogliamo conoscere il valore di $z \cdot z'$. A questo punto, sia $z \cdot z' = K(x, x')$ dove K, il Kernel, è il prodotto interno di $\phi(x)$ e $\phi(x')$ in Z, la condizione di Mercer consente di stabilire se un dato kernel sia ammissibile, ossia se corrisponda effettivamente al prodotto interno di due vettori in uno spazio: esiste una funzione ϕ che mappa i dati in uno spazio Z e una espansione:

$$K(x, y) = \sum_i \phi(x)_i \phi(y)_i$$

se e solo se, per ogni g (x) tale che $\int g(x)^2 dx$è finito, vale:

$$\int K(x, y)\, g(x) g(y) dx dy \geq 0$$

Questa equazione deve valere per qualsiasi g con norma finita. In alcuni casi specifici potrebbe non essere semplice verificare che la condizione di Mercer sia rispettata. Ad ogni modo è possibile dimostrare che questa condizione è soddisfatta per integrali positivi di potenze del prodotto interno: $K(x, y) = (x \cdot y)^p$. Una semplice conseguenza di questo fatto è che ogni kernel che possa essere espresso come $K(x, y) = \sum_{p=0}^{\infty} c_p (x \cdot y)^p$, dove c_p sono coefficienti reali positivi e la serie è uniformemente convergente, soddisfa la condizione di Mercer. Nel caso di dati separabili quello che vogliamo fare è esprimere la funzione che assegna la classe a un dato, $g(x) = sign(w \cdot x + w_0)$ in funzione del Kernel. Tenendo presente che:

$$w = \sum \alpha_n y_n z_n$$

definiamo la nuova funzione come:

$$g(x) = sign\left(\sum \alpha_n y_n K(x_n, x) + w_0\right)$$

dove

$$w_0 = \frac{1}{N_s}\sum\left(y_s - \sum \alpha_m y_m K(x_m, x_s)\right)$$

Finora abbiamo detto che non ci interessa cosa sia e cosa accada nello spazio Z ma è ovvio che deve esistere il prodotto interno di due vettori in esso. Possiamo affrontare il problema dell'esistenza usando due diversi approcci:

- per costruzione, ad esempio nel caso del kernel polinomiale
- usando proprietà matematiche, Condizioni di Mercer

5.1.3 SVM PER LA CLASSIFICAZIONE MULTICLASSE

Per quanto riguarda l'utilizzo delle SVM per problemi di classificazione multiclasse, si utilizza una tecnica che vale per tutti gli algoritmi e che permette di estendere le tecniche della classificazione binaria al caso multiclasse.

Supponiamo dunque di avere i dati di addestramento siano coppie $T = \{(x^i, y^i): x^i \in R^n, y^i \in \{1, \dots, k\}\, i = 1, \dots, l\}$ dove k è il numero totale di classi. Ci sono due approcci possibili per utilizzare le SVM per la classificazione multiclasse: l'approccio one-against-all e l'approccio one-against-one.

One-against-all

In questo approccio si costruiscono k classificatori in cui k è il numero di classi. Il j-esimo classificatore binario separa i vettori della classe j da quelli di tutte le altre classi. Quindi è addestrato considerando gli elementi della classe j-esima come positivi e tutti gli altri negativi. Si considerano i k problemi di classificazione binaria in cui i dati di training sono definiti come:

$$T_j = (x^i, y_j^i): y_j^i = \begin{cases} 1\ se\ \ y^i = j \\ -1\ se\ \ y^i \neq j \end{cases}\ con\ i = 1, \dots, l$$

Per ogni j = 1, . . . , k si deve addestrare una SVM^j e dunque si devono risolvere per ogni j = 1, . . . , k i problemi di massimo margine. Una volta ottenuta la soluzione dei k problemi j = 1, . . . , k sono disponibili k funzioni di decisione. Per decidere la classe di appartenenza di un generico input x si individua la funzioni di decisione a cui corrisponde il maggior valore dell'argomento, ovvero

One-against-one

In questo caso si costruiscono $\frac{k(k-1)}{2}$ classificatori ognuno addestrato sui dati relativi a due sole classi. Il generico classificatore è addestrato considerando gli elementi di due classi la m−esima e la n-esima come "antagonisti". Dunque, per ogni coppia di

indici m, n ∈ {1, . . . , k} si costruisce un insieme di dati di training che è un sottoinsieme dei dati originari. In particolare:

$$T_{mn} = (x^i, y^i_{mn}) : y^i_{mn} = \begin{cases} 1 \; se \;\; y^i = m \\ -1 \; se \;\; y^i = n \end{cases} \quad con \; i : y_i = m \; o \; y_i = n$$

Per ogni coppia di indici m, n ∈ {1, . . . , k} si deve addestrare una SVM^{mn} e dunque si devono risolvere per ogni m, n ∈ {1, . . . , k} i problemi di massimo margine. Una volta ottenuta la soluzione per ogni coppia di indici m, n ∈ {1, . . . , k} sono disponibili $\frac{k(k-1)}{2}$ funzioni di decisione. Per decidere la classe di appartenenza di un generico input x si adotta una strategia di voting, per ogni classe j si conta il numero di volte che il generico x è assegnato alla classe j. La classe di appartenenza di un input x è scelta come quella che ottiene il massimo numero di voti. Nel caso di classi con lo stesso numero di voti si può scegliere quella con indice più piccolo. In pratica si risolvono $\frac{k(k-1)}{2}$ problemi duali con un numero di variabili pari al numero di dati nelle due classi. Assumendo che in media ogni classe abbia ℓ/k dati di input, la dimensione dei problemi duali `e in media 2ℓ/k.

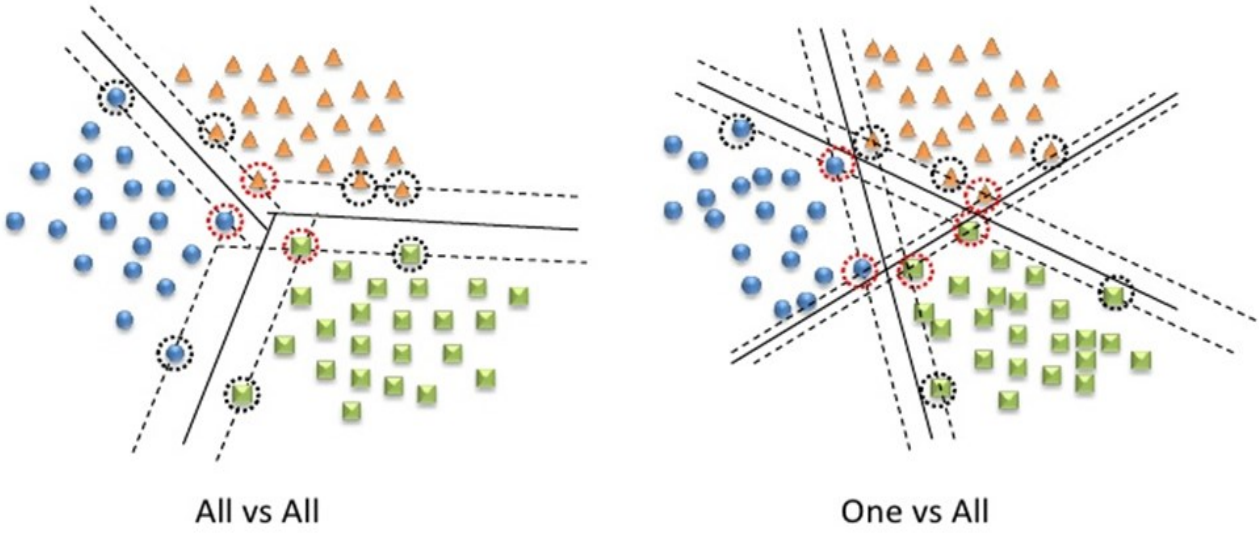


Figura 5.8 rappresentazione dell'algoritmo SVM nell'One-aganist-all e nell' One-against-one

5.2 K-NEAREST NEIGHBORS (KNN)

Il k-nearest neighbors (KNN) è un algoritmo di machine learning supervisionato utilizzato nel riconoscimento di pattern e per la classificazione. È chiamato lazy learner, in quanto non apprende una regola su come discriminare le classi dei vettori, ma memorizza il data set di apprendimento tutto intero, infatti per la classificazione di oggetti si basa sulle caratteristiche degli elementi vicini a quello considerato. Lo spazio viene partizionato in regioni in base alle posizioni e alle caratteristiche degli oggetti di apprendimento, un elemento sarà quindi classificato in base alla maggioranza dei voti dei suoi k elementi vicini, che sono presi da un insieme di oggetti per cui è nota la classificazione corretta. Più in particolare un punto, che rappresenta un oggetto, è assegnato a una classe se questa è la più frequente fra i k esempi più vicini all'oggetto sotto esame.

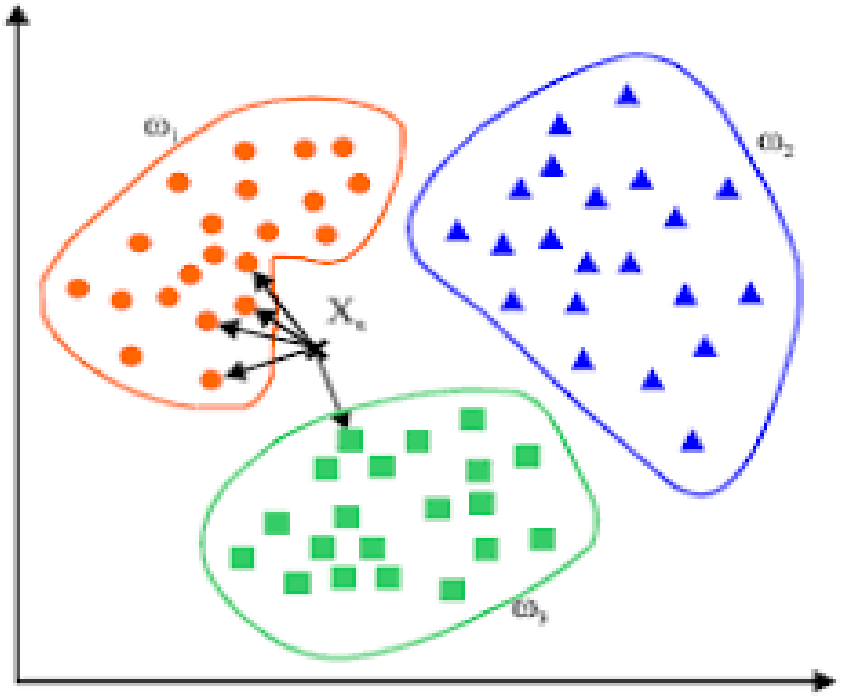


Figura 5.9 esempio di classificazione di un nuovo punto con il KNN

Ai fini del calcolo della distanza gli oggetti sono rappresentati attraverso vettori di posizione in uno spazio multidimensionale e la vicinanza si misura in base alla distanza fra punti. L'algoritmo può essere riassunto nei seguenti passaggi:

- si sceglie il numero k e una metrica per la distanza;
- si trovano i k elementi più vicini al campione da classificare;
- si assegna l'etichetta di classe con un voto a maggioranza.

Per quanto riguarda il calcolo della distanza fra i vari punti, in genere per stimare la vicinanza tra due elementi viene impiegata la distanza euclidea:

$$d_E(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

La scelta del valore di k, invece, dipende dai dati a disposizione per l'addestramento ed il test: valori più grandi riducono l'effetto del rumore nella classificazione, ma rendono i confini tra le classi meno distinti, quindi un buon valore di k può essere scelto eseguendo più prove modificando i dati del training-set.

Il KNN quindi si basa sul principio secondo cui gli oggetti sono simili se la distanza geometrica tra i vettori che li descrivono è piccola, ma, in alcune situazioni, la distanza geometrica può essere fuorviante. Infatti non tutti gli attributi sono uguali e nell'apprendimento automatico ce ne possono essere alcuni che sono irrilevanti, ovvero tali che i loro valori non hanno nulla a che fare con la classe che vogliamo identificare, ma in questo caso comunque influenzeranno la distanza geometrica tra i vettori. Ovviamente gli errori causati da attributi irrilevanti dipendono da quanti vengono utilizzati per descrivere gli esempi: è improbabile che pochi di essi distorcano il valore di $d_E(x,y)$ in modo significativo, se, invece, la maggioranza degli attributi non ha nulla a che fare con la classe che vogliamo riconoscere, la distanza geometrica diventerà quasi priva di significato e le prestazioni del classificatore saranno pessime. Un modo per risolvere questo problema è normalizzare gli attributi, ovvero ridimensionarli in modo da far rientrare tutti i valori nello stesso intervallo [0,1]. Tra i vari meccanismi che sono stati utilizzati per tale scopo, il più semplice è quello che identifica per primo, per l'attributo dato, il suo massimo e il suo minimo, e poi sostituisce ogni valore, x, di questo attributo usando la seguente formula:

$$x = \frac{x - MIN}{MAX - MIN}$$

In modo tale che se abbiamo un set di allenamento che comprende diversi elementi e sottraiamo ad ognuno di questi il valore minimo e poi dividiamo per MAX-MIN otteniamo una situazione in cui tutti i valori sono compresi nell'intervallo [0,1]. La normalizzazione riduce il tasso di errore in molte applicazioni pratiche, soprattutto se le scale degli attributi originali variano in modo significativo, però, lo svantaggio è che la descrizione degli esempi in questo modo diventa distorta e inoltre la scelta di far rientrare tutti i valori tra 0 e 1 in alcuni casi potrebbe non essere giustificata.

5.3 PRESTAZIONI E CONFRONTO CON LE RETI NEURALI

Dal punti di vista delle prestazioni queste tecniche di data driven presentano, ovviamente, tra di loro diversi vantaggi e svantaggi. Nel caso generale le SVM sono degli algoritmi di apprendimento promettenti nel campo dell'apprendimento automatico per via della loro semplicità. Possono essere addestrate in modo semplice sia nel caso lineare che nel caso non lineare grazie all'utilizzo di funzioni kernel, anche lavorando in spazi di enormi dimensioni, infatti, usando un kernel, una SVM può lavorare in uno spazio a dimensionalità potenzialmente infinita utilizzando sempre gli stessi algoritmi. Con l'algoritmo SVM non ci sono neanche problemi di curse of dimensionality, che di solito si presentano con spazi troppo grandi, perché l'algoritmo è indipendente dalla dimensione dello spazio finale e lavora nello spazio originale dei punti. Con curse of dimensionality ci si riferisce a vari fenomeni che sorgono durante l'analisi e l'organizzazione dei dati in spazi ad alta dimensione. Il tema comune di questi problemi è che quando la dimensionalità aumenta il volume dello spazio aumenta così velocemente che i dati disponibili diventano scarsi. Per ottenere un risultato statisticamente valido e affidabile, la quantità di dati necessari a supportare il risultato spesso cresce in modo esponenziale con la dimensionalità.

Quando vengono usati come classificatori, quindi, sono molto efficienti dato che sono solo i vettori di supporto che svolgono un ruolo nel determinare il limite di classificazione e sono sufficienti per dare una rappresentazione compatta del training set. Il loro numero fornisce un'idea della capacità di generalizzazione, infatti tutti gli altri punti del set di allenamento non devono essere memorizzati. La difficoltà principale consiste nella scelta della funzione di kernel ottimale e del parametro di regolarizzazione C che consente una certa varianza del modello, infatti le prestazioni della SVM sono fortemente dipendenti da queste scelte. Inoltre, uno svantaggio di SVM è la complessità di classificazione piuttosto elevata che si ridimensiona linearmente con il numero di vettori di supporto. Ciò è dovuto al fatto che per la classificazione di un campione, la funzione del kernel è valutata per tutti gli SV, quindi anche oneroso in termini di calcolo e tempo. Infine, quando si tratta di un classificatore binario i suoi risultati possono essere ottimi, cosa che invece non accade nel caso multiclasse, in cui la SVM perde molto in termini di efficienza.

Invece nel modello KNN oltre ad essere semplice ed intuitivo ha il vantaggio principale è che il classificatore si adatta immediatamente man mano che aggiungiamo vettori di apprendimento. Dall'altro lato, il difetto è che la complessità computazionale per classificare nuovi campioni cresce linearmente con il numero di vettori di apprendimento. Per di più, non possiamo ignorare a priori alcun vettore di training dato che non c'è un vero e proprio apprendimento. Così, lo spazio di archiviazione e

il numero di distanze da calcolare possono diventare un problema quando si lavora con grandi data set.
L'algoritmo KNN è robusto per i dati di allenamento rumorosi ed è efficace se i dati di allenamento sono grandi, ma in questi casi può avere scarse prestazioni di run time perché è molto sensibile alle funzionalità irrilevanti o ridondanti dato che tutte le funzionalità contribuiscono alla somiglianza e quindi alla classificazione, infatti anche gli attributi irrilevanti vengono pesati come gli altri e questo può causare instabilità. Infine, l'apprendimento basato sulla distanza non è chiaro per definire quale tipo di distanza usare e quale attributo usare per produrre i migliori risultati.
Rapportandoli alle reti neurali le SVM sia nella classificazione binaria, che nella multiclasse, sono soltati in grado di individuare la classe di appartenenza di un dato, privandoci di tutte le informazioni in più che forniscono le reti neurali. Infatti, come è già stato detto le reti neurali quando analizzano un dato di input ci forniscono in output la classe di appartenenza per quegli input, ma se si esaminano gli output si è in grado anche di capire il grado di appartenenza a quella classe. Nel caso di studio gli output della rete neurale indicavano anche la distanza che gli input avevano rispetto alle altre classi. Ad esempio, quando venivano classificati degli input ottenuti da i segnali ricavati con un pretensionamento della cinghia a 45 hz questi venivano classificati nella condizione di sicurezza, ma analizzando gli output si poteva notare che si trovavano in una condizione limite o comunque molto vicini anche alla condizione critica di sicurezza. Queste informazioni invece non ci vengono fornite dalle SVM che permettono soltanto di individuare la classe di appartenenza, mentre questo aspetto nell'algoritmo KNN potrebbe essere un po' diverso, infatti anche se, come nelle SVM non ci vengono forniti dettagli sul grado di appartenenza, ad un classe quando si inseriscono nuovi input si può notare che questi potrebbero avere come vicini anche elementi di altre classi.

5.4 CONCLUSIONI

Per i nuovi robot e sistemi automatizzati delle fabbriche intelligenti, per evitare inconvenienti in termini di produttività, è quasi necessario disporre di un sistema PdM in grado di rilevare e classificare i diversi guasti. Questo elaborato propone una tecnica per la classificazione dei guasti sui CR, utilizzando le reti neurali di cui sono stati evidenziati i punti di forza anche rispetto ad altre tecniche di classificazione. Il metodo proposto permette di estrarre caratteristiche dai segnali CR ed eseguire un algoritmo che classifica con precisione i guasti. Per concludere, i risultati ottenuti identificano al meglio i guasti degli azionamenti dei CR e permettono di capire quando si lavora in una condizione di sicurezza e quando ci si sta allontanando da questo stato.

BIBLIOGRAFIA

[1] Yi Lu Murphey*, Senior Member, IEEE*, M. Abul Masrur*, Senior Member, IEEE*, ZhiHang Chen and Baifang Zhang, "Model-based fault diagnosis in electric drives using machine learning", 2006

[2] S.S. Moosavia, A. Djerdira, Y. Ait-Amiratb and D.A. Khaburi "ANN based fault diagnosis of permanent magnet synchronous motor under stator winding shorted turn", 2015

[3] Jose Gregorio Ferreira and Adam Warzecha "An application of machine learning approach to fault detection of a synchronous machine", 2017

[4] Miroslav Kubat "An Introduction To MachineLearnin", 2017

[5] Yong Chen , Siyuan Liang, Wanfu Li, Hong Liang and Chengdong Wang "Faults and Diagnosis Methods of Permanent Magnet Synchronous Motors: A Review", 2019

[6] Andrea Giantomassi, Francesco Ferracuti, Sabrina Iarlori, Gianluca Ippoliti and Sauro Longhi, "Signal Based Fault Detection and Diagnosis for rotating electrical machines: Issues and Solutions"

[7] Dubravko, miljkovic, "Brief review of motor current signature analysis"

[8] Andrea Bonci, Sauro Longhi, Giacomo Nabissi, Federica Verdini "Predictive Maintenance System using motor current signal analysis for Industrial Robot"