

Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in
Scienze Statistiche



TESI DI LAUREA

**Classificazione di segnali EEG
per la Brain-Computer Interface.**

**Analisi delle matrici di covarianza tramite
geometria Riemanniana.**

Relatore Prof. Bruno Scarpa
Dipartimento di Scienze Statistiche

Correlatore Prof. Livio Finos
Dipartimento di Psicologia dello Sviluppo e della Socializzazione

Laureanda: Laura Masiero
Matricola N. 1183284

Anno Accademico 2018/2019

*Alla mia famiglia e a Mirella,
mi manca molto sentire il tuo "Dottoressa!".
Alla fine, come ti avevo promesso, ce l'ho fatta.
Spero che tu sia fiera di me.*

Indice

Introduzione	11
1 Dati EEG e Brain-Computer Interfaces	13
1.1 Il segnale elettroencefalografico	14
1.2 Sistemi di <i>Brain-Computer Interface</i>	19
1.2.1 <i>Brain-Computer Interfaces</i> basate su segnali EEG . . .	21
2 Geometria Riemanniana	25
2.1 Concetti e strumenti	26
2.1.1 Distanza Riemanniana	27
2.1.2 Proiezioni nello spazio tangente	28
2.1.3 Media delle matrici SPD	29
2.2 Distribuzione di Riemann-Gauss	30
3 Metodi di classificazione	35
3.1 Metodi di <i>statistical learning</i>	35
3.1.1 <i>Support Vector Machines</i>	35
3.1.2 <i>Model-based Oblique Random Forest</i>	40
3.1.3 <i>Bagging</i>	44
3.1.4 <i>Boosting</i>	45
3.2 Metodi basati sulla geometria Riemanniana	48
3.2.1 <i>Support Vector Machines</i> con kernel Riemanniano . . .	48
3.2.2 <i>Minimum Distance to Riemannian Mean</i>	51
3.2.3 Rapporto di verosimiglianza	52

4	<i>Brain Invaders</i>: colpire gli alieni col pensiero	55
4.1	Costruzione di BCI <i>P300-based</i>	55
4.2	Brain Invaders	57
5	Analisi dei dati	61
5.1	Presentazione del dataset	61
5.1.1	Paradigma sperimentale	61
5.1.2	Acquisizione dei dati	62
5.1.3	Importazione e preprocessing dei dati	63
5.1.4	Divisione in due insiemi di dati distinti	63
5.2	Creazione di <i>super trial</i>	64
5.3	Analisi esplorative	66
5.3.1	<i>t</i> -SNE	66
5.4	Risultati	71
5.4.1	Metodi di <i>statistical learning</i>	72
5.4.2	Metodi basati sulla geometria Riemanniana	79
5.4.3	Risultati salienti	86
	Conclusioni	89
A	Codice R	91
A.1	Importazione dei dati in formato <i>.mat</i>	91
A.2	Funzioni create per il pacchetto RiemannR	92
	Bibliografia	99

Elenco delle figure

1.1	Esempio di segnale rilevato e di configurazione degli elettrodi .	15
1.2	Sistema 10/20	16
1.3	Ritmi EEG	17
1.4	Tipi di BCI basati sul segnale EEG	22
2.1	Rappresentazione della mappatura esponenziale e logaritmica .	28
3.1	Classificatori basati su vettori di supporto	37
3.2	Esempio di separazione con alberi decisionali	40
3.3	Effetto della regolarizzazione sulle direzioni degli split	42
4.1	<i>Brain Invaders</i> : schermate di gioco.	58
5.1	Posizionamento degli elettrodi durante le sessioni di gioco di <i>Brain Invaders</i>	62
5.2	Esempio di matrici di covarianza, rappresentate con heatmap, costruite con e senza <i>super trial</i>	65
5.3	Rappresentazione 2D della similarità tra matrici di covarianza: soggetto 1	69
5.4	Rappresentazione 3D della similarità tra matrici di covarianza: soggetto 1	69
5.5	Rappresentazione 2D della similarità tra matrici di covarianza: soggetto 8	70
5.6	Rappresentazione 3D della similarità tra matrici di covarianza: soggetto 8	70

5.7	Diagramma di dispersione delle osservazioni del <i>test set</i> : distanze dal baricentro di ciascuna classe (<i>target</i> e <i>non-target</i>).	
	Soggetti 1 e 8	88

Elenco delle tabelle

3.1	Tipologia di funzioni <i>kernel</i> usate nelle SVM	39
5.1	Dataset 1. Risultati ottenuti tramite classificazione con i metodi di <i>statistical learning</i> . Insieme di stima generalizzato. . .	73
5.2	Dataset 1. Risultati ottenuti tramite classificazione con i metodi di <i>statistical learning</i> . Insieme di stima specifico per ogni soggetto.	74
5.3	Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi di <i>statistical learning</i> . Insieme di stima generalizzato.	75
5.4	Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi di <i>statistical learning</i> . Insieme di stima generalizzato.	76
5.5	Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi di <i>statistical learning</i> . Insieme di stima specifico per ogni soggetto.	77
5.6	Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi di <i>statistical learning</i> . Insieme di stima specifico per ogni soggetto.	78
5.7	Dataset 1. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima generalizzato.	80
5.8	Dataset 1. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima specifico per ogni soggetto.	81

5.9	Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima generalizzato.	82
5.10	Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima generalizzato.	83
5.11	Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima specifico per ogni soggetto.	84
5.12	Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima specifico per ogni soggetto.	85

Introduzione

I dati provenienti dall'elettroencefalografia (EEG) sono ampiamente utilizzati in ambito clinico e costituiscono la base per molti progetti di ricerca volti a migliorare la comprensione di alcune malattie ed alla creazione di strumenti che permettono, alle persone con scarsa mobilità, di avere una maggiore autonomia, consentendo loro un'esperienza di vita soddisfacente.

Questi dati provengono dall'attività elettrica spontanea prodotta dal cervello, in un periodo di tempo limitato, e vengono acquisiti attraverso una serie di elettrodi posizionati in diversi punti del cranio.

Una delle più recenti innovazioni nelle aree di ricerca che utilizzano segnali EEG è rappresentata dalle *Brain-Computer Interfaces* (BCIs), tecnologie che convertono i segnali elettrici prodotti dal cervello impegnato in un compito specifico, come l'immaginazione di un determinato movimento (la *Motor-Imagery BCI*), in istruzioni che possono essere comprese da alcuni software. Queste tecnologie sono impiegate in una grande varietà di contesti, come nell'ambito dei videogiochi: in Congedo et al. (2011) questa tecnologia è stata applicata a un prototipo di gioco basato sul potenziale evento-relato P300, ***Brain Invaders***, considerato nello specifico in questa tesi.

Esso si basa sul noto *Space Invaders*, la differenza è che il giocatore non ha periferiche disponibili per controllare il gioco: esso viene controllato da particolari variazioni, rappresentative dell'onda P300, dei segnali EEG rilevati. Tuttavia, la maggior parte delle BCI sono scarsamente utilizzate al di fuori degli ambienti di laboratorio controllati, perché l'identificazione di uno stato mentale a partire dal segnale EEG è un processo complesso.

Le caratteristiche di questo segnale, quindi, non favoriscono il compito: l'EEG subisce pesantemente l'effetto di rumore indesiderato, richiede lunghi tempi

di calibrazione, generalmente non è stazionario e la sua descrizione richiede un elevato volume di dati, direttamente proporzionale al numero di elettrodi utilizzato nella fase di acquisizione (Lotte et al., 2007). In definitiva, la realizzazione di BCI in grado di prevedere efficacemente richiede tecniche di elaborazione del segnale e strumenti di *statistical learning* avanzati.

Per raggiungere un'ottima classificazione e diminuire il tempo di calibrazione dei modelli, è stata proposta l'applicazione di strumenti appartenenti alla geometria Riemanniana, come le *Support Vector Machines* modificate per operare in modo appropriato con matrici di covarianza campionarie, che si trovano in una varietà Riemanniana da un punto di vista geometrico (Congedo et al. 2013), o algoritmi basati sulle distanze come il *Minimum Distance to Riemannian Mean*.

In questa tesi si cercherà di implementare alcuni metodi basati sulla geometria Riemanniana, per poi confrontare la loro *performance* a quella di metodi di *statistical learning* molto conosciuti per la loro efficacia, come il *Bagging*, l'*eXtreme Gradient Boosting* e una modifica delle *Random Forest*. In particolare, l'insieme di dati su cui verranno applicati le varie metodologie provengono da sessioni di gioco di ***Brain Invaders***.

Il Capitolo 1 verrà dedicato alla descrizione dei segnali EEG ed alle BCIs, seguito dal Capitolo 2 in cui verranno presentati i concetti principali della geometria Riemanniana.

In seguito, nel Capitolo 3 verranno presentati da un punto di vista teorico i metodi utilizzati nella classificazione, sia quelli di *statistical learning* che quelli basati sulla geometria Riemanniana, e nel Capitolo 4 sarà descritto in dettaglio il funzionamento di ***Brain Invaders***.

Per concludere, nel Capitolo 5 verranno esposti e commentati i risultati ottenuti dai diversi metodi, per cercare di comprendere quali siano le metodologie più indicate in questa tipologia di problemi di classificazione.

Le funzioni relative alle metodologie basate sulla geometria Riemanniana descritte nel Capitolo 3 sono state raccolte in un pacchetto R, **RiemannR**, scaricabile dal seguente url: <https://github.com/LauraMasiero/RiemannR>.

Capitolo 1

Dati elettroencefalografici e Brain-Computer Interfaces

Nel 1929 Hans Berger condusse diversi esperimenti, nei quali mostrò che l'attività elettrica del cervello umano poteva essere misurata piazzando elettrodi sullo scalpo, amplificando il segnale e tracciando i cambiamenti di voltaggio nel corso del tempo (Berger, 1929). La registrazione di questa attività elettrica è chiamata *elettroencefalogramma*, o EEG (Luck, 2014).

L'EEG è molto utile sia nelle applicazioni nel campo scientifico che clinico, tuttavia esso è una misura grossolana dell'attività del cervello, e non può essere usato nella sua forma grezza per misurare la maggior parte di specifici processi neurali che costituiscono il fulcro delle neuroscienze cognitive, questo perché l'attività elettrica misurata rappresenta una mistura di segnali provenienti da diverse sorgenti neurali e la loro separazione non è semplice. Con l'elettroencefalogramma è possibile anche estrarre le risposte neurali associate a specifici stimoli di natura sensoriale, cognitiva e motoria tramite diverse tecniche: queste risposte specifiche sono chiamate *potenziali evento-relati*, o ERP, per sottolineare il fatto che costituiscono variazioni di potenziale elettrico relativi a specifici eventi (Luck, 2014).

Nel corso di questo Capitolo per prima cosa verrà fornita una spiegazione più dettagliata dell'elettroencefalografia, seguita da una descrizione delle caratteristiche dei sistemi di *Brain-Computer Interface*, insieme al loro funzio-

namento. Per finire saranno presentati gli ERP, in particolare il potenziale P300 ed il paradigma *oddball* solitamente usato per misurare i potenziali evento-relati.

1.1 Il segnale elettroencefalografico

L'elettroencefalografia è una tecnica non invasiva volta alla rilevazione dell'attività elettrofisiologica del cervello, attuata tramite la misura dei potenziali sullo scalpo. L'attività prevalentemente misurata è quella dei neuroni piramidali presenti nella corteccia, che data la loro prossimità rispetto allo scalpo e alle loro caratteristiche, danno luogo a campi elettrici che si sommano facilmente e sono così rilevabili in superficie (He, 2005).

Anche se l'elettroencefalogramma gode di una buona risoluzione temporale, nello specifico nell'ordine di decine di millisecondi (Hill et al., 2006), di una semplicità e sicurezza di utilizzo e di costi contenuti, esso presenta diverse problematiche, come la scarsa risoluzione spaziale dovuta al passaggio dei segnali attraverso vari strati di tessuto e soprattutto attraverso il cranio che opera come elemento di sfocatura spaziale (Hill et al., 2006), ed una grande sensibilità al rumore, come ad esempio artefatti del movimento oculare e muscolare.

Per quanto riguarda gli elettrodi utilizzati per rilevare il segnale elettrico, essi possono essere di diverse tipologie. Nell'ambito clinico solitamente vengono usati quelli di cloruro di argento, che necessitano dell'applicazione di gel o pasta elettrolitica tra elettrodo e scalpo, per assicurare una registrazione accurata del segnale (Sinclair et al., 2007). Tuttavia l'utilizzo di questi elettrodi che richiedono la presenza di gel portano ad un allungamento dei tempi necessari alla preparazione del soggetto e necessitano una continua manutenzione, dovuta al fatto che col passare del tempo il gel tende a seccarsi e ciò influisce in maniera negativa sulla qualità delle registrazioni (Nicolas-Alonso e Gomez-Gil, 2012).

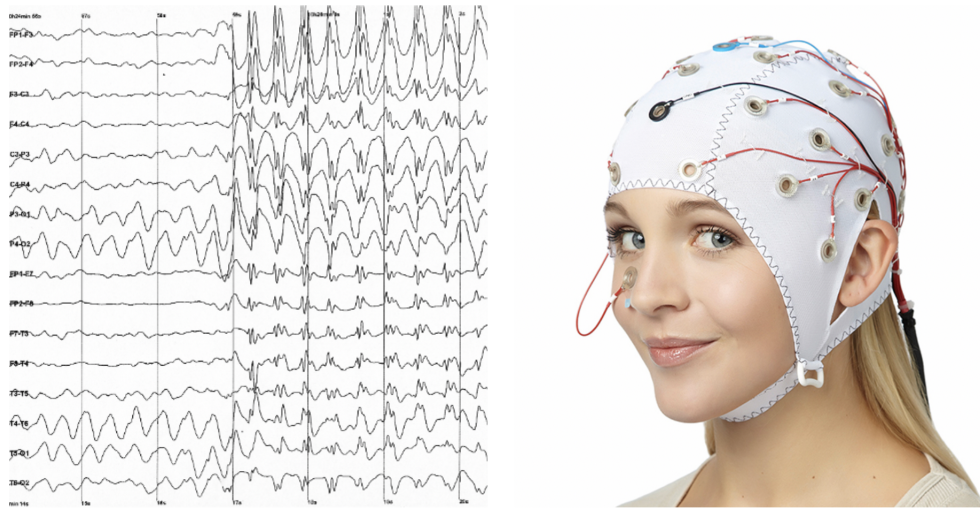


Figura 1.1: Esempio di segnale EEG rilevato (a sinistra) e di configurazione degli elettrodi sullo scalpo (a destra). Fonte: *Wikipedia, Electroencephalography* e *Brain Products UK*.

Una possibile soluzione a questi problemi è costituita dall'utilizzo di elettrodi a secco, detti anche di tipo *dry*, che come suggerisce il nome non necessitano di gel per la loro applicazione, che però hanno un costo maggiore dovuto alla loro diversa costruzione (Nicolas-Alonso e Gomez-Gil, 2012, Fonseca et al., 2007, Taheri et al., 1994). Nelle registrazioni multicanale, come quella d'esempio in Figura 1.1, solitamente si utilizza la disposizione degli elettrodi sullo scalpo data dal Sistema Internazionale 10/20 (Klem et al., 1999), visibile in Figura 1.2. Il suo nome è dovuto al fatto che le distanze effettive tra gli elettrodi adiacenti sono il 10% o il 20% della distanza totale anteriore-posteriore o destra-sinistra del cranio.

Il sistema usa come riferimenti quattro punti anatomici specifici:

- **nasion**: punto craniometrico sulla linea mediana alla radice del naso, in corrispondenza della sutura tra le due ossa nasali e l'osso frontale;
- **inion**: punto craniometrico che rappresenta la sommità della protuberanza esterna dell'osso occipitale;
- **punti preauricolari**: piccole depressioni alla radice degli zigomi (Klem et al., 1999).

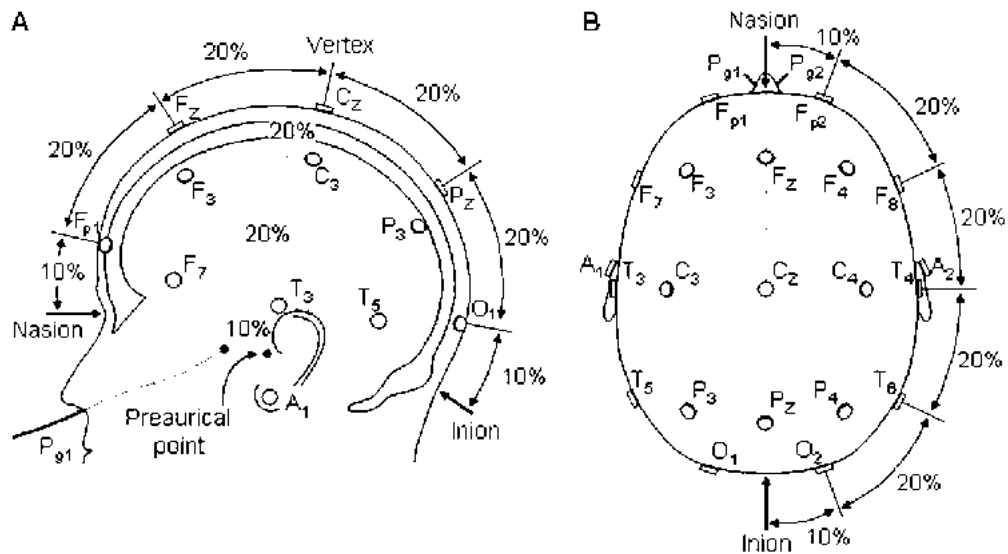


Figura 1.2: Sistema 10/20. Nel pannello A visione di lato della posizione degli elettrodi, nel pannello B visione dall'alto della stessa. Fonte: Meulman, 2015.

Gli elettrodi, secondo il Sistema Internazionale 10/20, possiedono un nome specifico dato da una o due lettere ed un numero. La lettera indica la localizzazione sullo scalpo dell'elettrodo, si ha infatti A per gli elettrodi posizionati vicino alle orecchie, C per la regione centrale, P per quelle parietali, F per le frontali, Fp per le regioni fronto-polari ed infine O per quelle occipitali; il numero invece può essere pari o dispari: numeri pari indicano gli elettrodi che stanno a destra, quelli dispari gli elettrodi di sinistra. Infine, la lettera "z", che sta per zero, si riferisce agli elettrodi sulla linea mediana (Klem et al., 1999).

Esistono altre varianti del sistema 10/20, come la versione 10/10 e 10/5, che estendono il sistema per la locazione di più elettrodi, dato che il sistema originale del 1958 tiene in considerazione solo 19 elettrodi (Klem et al., 1999): con le nuove varianti è possibile arrivare fino alla collocazione di più di 300 elettrodi (Nuwer et al., 1998, Oostenveld e Praamstra, 2001).

Per raggiungere una collocazione precisa e ridurre i tempi di preparazione vengono utilizzate delle cuffie su cui gli elettrodi sono già premontati nelle posizioni definite dallo standard preso in considerazione.

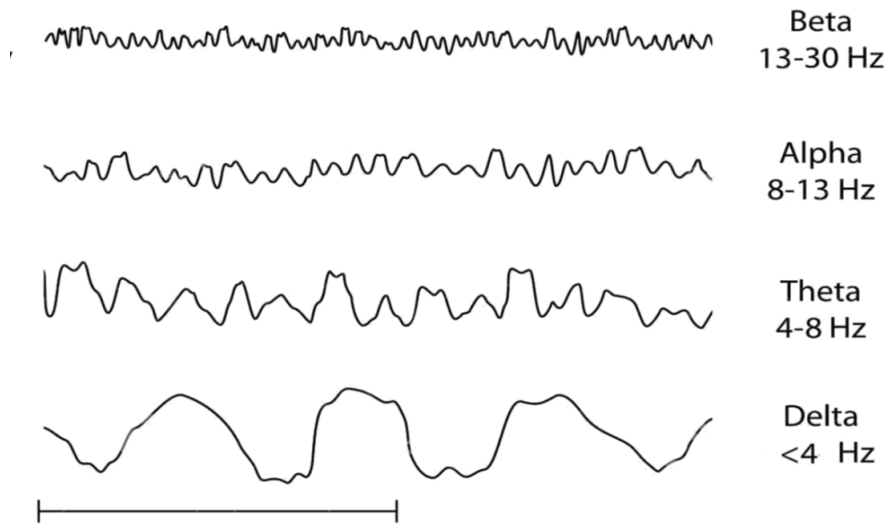


Figura 1.3: Bande di frequenza del segnale EEG: ritmi. Fonte: Mondini, 2015.

Passando al segnale EEG, esso è descritto dall'ampiezza e dalla frequenza delle oscillazioni misurate. L'ampiezza può essere anche di una decina di mV sulla superficie cerebrale, mentre sullo scalpo sono nell'ordine del centinaio di μV . Per quanto riguarda la frequenza invece il *range* entro cui stanno le oscillazioni è compreso tra gli 0.5 e 100 Hz (Rieger, 2006). In questo *range* è possibile individuare diverse onde cerebrali: i ritmi EEG.

Essi sono cinque e, a seconda della frequenza, come è visibile in Figura 1.3, si dividono in:

- **onde delta**, δ : caratterizzate da una frequenza che va da 0,5 a 3,9 Hz e da un'ampiezza di solito inferiore ai 100 μV (Rieger, 2006). Esse caratterizzano gli stadi di sonno profondo, mentre è considerata patologica un'eventuale attività in banda δ nello stato di veglia;
- **onde theta**, θ : il ritmo delle onde θ è dominante nel neonato ed è presente in molte patologie cerebrali dell'adulto, negli stati di tensione emotiva, o frustrazione come ad esempio quando è stato rimosso all'improvviso uno stimolo piacevole (Rieger, 2006) e nell'ipnosi. Si distingue in θ *lento* (4-6 Hz) e θ *rapido* (6-7.5 Hz), con una tensione media di 75 μV ;

- **onde alpha**, α : costituisce la frequenza di base presente in un EEG ed è distinto in α *lento* (8-9 Hz), α *intermedio* (9-11.5 Hz) ed α *rapido* (11.5-13 Hz), con un'ampiezza inferiore ai 10 μV (Rieger, 2006). Le onde α sono caratteristiche delle condizioni di veglia ma a riposo mentale. L'ampiezza di questo tipo di onde, nelle aree occipitali, aumenta molto quando si chiudono gli occhi mentre diminuisce alla loro riapertura. Esse infine non sono presenti nel sonno, fatta eccezione per lo stadio R.E.M., e quando un soggetto è sottoposto ad un'attività cerebrale che richiede concentrazione, si cominciano a registrare ritmi a frequenza maggiore (Rieger, 2006).

Nella stessa banda di frequenze si trovano anche le **onde μ** , dette anche ritmi sensomotori, molto interessanti perché fortemente correlate sia al movimento che alla sua semplice osservazione ed alla sua immaginazione (McFarland et al., 2000, Pfurtscheller et al., 2006): esse sono infatti localizzate nella corteccia motoria. Le onde μ sono utilizzate in alcune *Brain-Computer Interfaces* che sfruttano la desincronizzazione dell'attivazione di neuroni, che avviene quando una persona vede altri soggetti muoversi o semplicemente immagina un movimento.

Ciò risulta utile alla comprensione di quanto la persona sta immaginando perché quando un gruppo di neuroni è a riposo, essi tendono ad attivarsi in modo sincrono, mentre quando il soggetto viene spinto ad immaginare un movimento ne risulta una desincronizzazione di attivazione, che può essere rilevata ed analizzata da un computer;

- **onde beta**, β : esse vengono distinte in β *lento* (13.5-18 Hz) e β *rapido* (18.5-30 Hz), e presentano una tensione elettrica inferiore ai 20 μV (Rieger, 2006). Le onde β sono dominanti in un soggetto ad occhi aperti e impegnato in un'attività cerebrale qualsiasi, quasi continue negli stati di allerta (fase di *arousal*), ma anche durante il sonno onirico nella fase R.E.M. e si registrano soprattutto dalle aree cerebrali frontali. Le onde β sono anche associate all'attività motoria, sia reale che immaginaria (McFarland et al., 2000);

- **onde gamma**, γ : sono caratterizzate da frequenze maggiori di 30 Hz e ampiezza minore di 10 μV e denotano stati di profonda concentrazione (Nicolas-Alonso e Gomez-Gil, 2012). Esse sono più facilmente influenzate da artefatti muscolari.

1.2 Sistemi di *Brain-Computer Interface*

Wolpaw et. al (2002) descrive una *Brain-Computer Interface* (BCI) come un "sistema di comunicazione in cui messaggi o comandi che un individuo invia al mondo esterno non passano attraverso le normali vie di uscita del cervello, come i nervi ed i muscoli periferici". Quindi, una BCI fornisce un canale di comunicazione verso l'esterno alternativo, per i soggetti che sono impossibilitati ad usare le normali vie di comunicazione del sistema nervoso, operando una traduzione dei segnali cerebrali in istruzioni intellegibili da computer. Ciò significa che le *Brain-Computer Interfaces* dipendono dall'interazione del cervello dell'utente, che produce i segnali misurati dalle BCIs, con le BCIs stesse, che traducono questi segnali in comandi specifici.

Come qualsiasi sistema di comunicazione o controllo, una BCI necessita di un input, come l'attività elettrofisiologica dell'utente, di un output, vale a dire i comandi del dispositivo, componenti che traducono gli input in output ed un protocollo che determina l'inizio, l'*offset* ed i tempi di funzionamento (Wolpaw et al., 2002). Quindi, un sistema BCI è costituito principalmente dalle seguenti parti:

- **acquisizione del segnale:** questo modulo permette la registrazione dell'attività cerebrale del soggetto. I metodi di acquisizione possono essere vari, e possono essere classificati in base ai segnali che registrano, che possono essere *elettrofisiologici* ed *emodinamici*.

I metodi che acquisiscono segnali *elettrofisiologici* si concentrano sulle variazioni di campo elettromagnetico che sono diretta conseguenza dell'attività neuronale. Essi comprendono l'elettroencefalografia (EEG) e la magnetoencefalografia (MEG) tra le metodologie non invasive, l'e-

lettrocorticografia (ECoG) e la registrazione intracorticale tra quelle invasive.

Invece, i metodi che sfruttano i segnali *emodinamici* comprendono, ad esempio, tecniche di neuroimaging come la risonanza magnetica funzionale (fMRI) e la spettroscopia del vicino infrarosso (NIRS) (Purkayastha et al., 2014, Nicolas-Alonso e Gomez-Gil, 2012). Queste metodologie misurano la risposta emodinamica cerebrale, riuscendo a cogliere così le aree con maggiore attivazione neuronale.

Inoltre, i metodi di acquisizione possono presentare un'ulteriore classificazione, che si basa sul tipo di *input* che generano la risposta fisiologica del soggetto: si possono avere *input evocati* ed *input spontanei*.

Gli *input evocati*, ad esempio segnali elettroencefalografici provocati da lettere lampeggianti, derivano dalla stimolazione sensoriale stereotipata fornita dalla BCI, gli *input spontanei* invece, come ad esempio i ritmi EEG rilevati sulla corteccia sensomotrice, non dipendono dalla loro generazione da tale stimolazione (Wolpaw et al., 2002).

Qualsiasi sia il metodo di rilevazione scelto, una volta ottenuto il segnale cerebrale, esso viene amplificato e digitalizzato;

- **elaborazione del segnale:** questo modulo si occupa della traduzione dei segnali elettrofisiologici registrati in istruzioni per le periferiche che fanno parte della BCI. Esso è composto dalle seguenti parti: *preprocessing del segnale*, in cui per esempio sono applicati filtri spaziali e sono attuate analisi spettrali del segnale in modo da migliorare il rapporto segnale/rumore, *estrazione di features* dal segnale preprocessato e *traduzione in comandi* delle *features* appena estratte (Wolpaw et al., 2002), utili per il comando della BCI;
- **feedback:** rappresenta la conseguenza osservabile dell'attività cerebrale del soggetto. Oltre ad essere il prodotto previsto dalla BCI, questo risultato è il *feedback* che il cervello utilizza per mantenere e migliorare l'accuratezza e la velocità della comunicazione (Wolpaw et al., 2002).

1.2.1 *Brain-Computer Interfaces* basate su segnali EEG

Le *Brain-Computer Interfaces* solitamente possono essere divise in due gruppi principali: le BCI *dependenti* e quelle *indipendenti*.

Nel primo gruppo si trovano le BCI che sfruttano i *Visual Evoked Potentials* (VEP), che dipendono dal controllo muscolare della direzione dello sguardo (Wolpaw et al., 2002), mentre nel secondo gruppo, quello delle BCI *indipendenti*, ne troviamo una più ampia varietà, che sono i sistemi BCI che usano i potenziali corticali lenti, i potenziali evocati P300, ritmi μ e β ed infine i potenziali d'azione neuronale corticale. Queste ultime non necessitano della mediazione delle normali vie d'uscita del sistema nervoso centrale, come i muscoli degli occhi per volgere lo sguardo in una particolare direzione (Wolpaw et al., 2002).

In Figura 1.4 è possibile vedere le diverse rilevazioni di alcuni metodi di BCI. Dato che il sistema di *Brain-Computer Interface* considerato in questo lavoro è basato sul potenziale evocato P300, di seguito sarà descritta nel dettaglio solamente questa varietà di BCI *indipendente*.

I potenziali evocati sono visibili come variazioni del segnale EEG in risposta a particolari stimoli (Major e Conrad, 2014) ed i potenziali evento-relati sono anch'essi piccoli cambiamenti rilevati sul segnale EEG, ma oltre ad essere provocati da eventi esterni, possono essere causati anche da eventi interni, come un pensiero (Handy, 2005).

In particolare, il potenziale evento-relato P300 si presenta come una piccola variazione rilevata sul segnale EEG registrato, tipicamente evocato in corrispondenza della corteccia parietale, a seguito di stimoli uditivi, visuali o somatosensoriali, che provocano un picco positivo dopo 300 ms, quando questi sono intervallati a stimoli più frequenti o di routine (Wolpaw et al., 2002). Il paradigma *oddball* che solitamente viene utilizzato nelle BCI basate sul potenziale evento-relato P300, è stato usato per la prima volta da Farwell e Donchin (1988), in cui all'utente era presentata una matrice 6×6 di lettere, numeri e/o altri simboli.

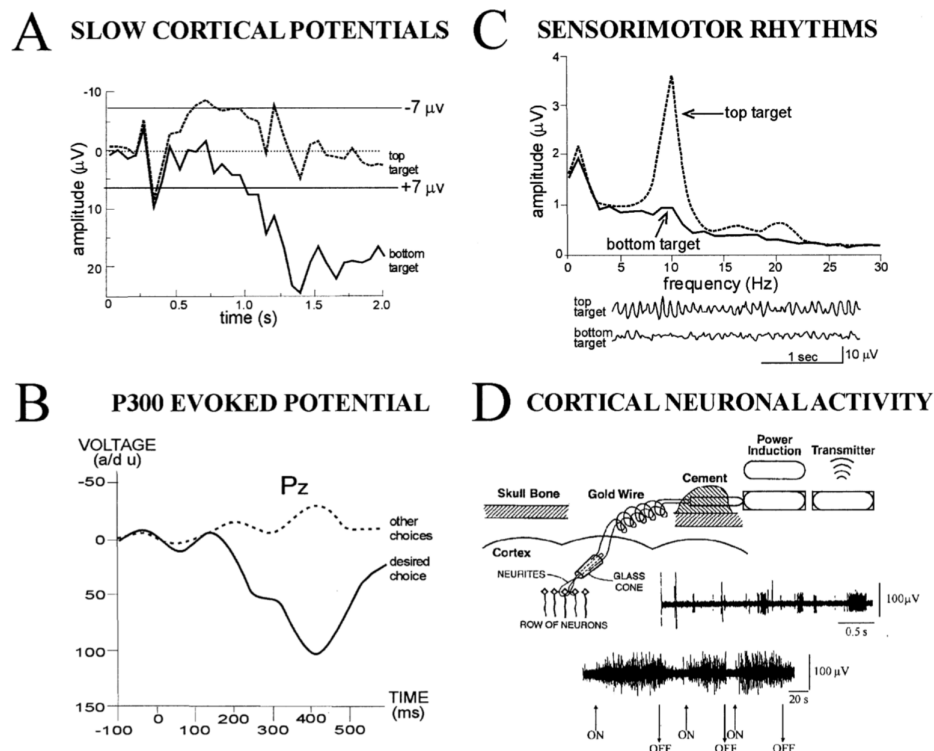


Figura 1.4: Tipi di *Brain-Computer Interfaces*. I pannelli A,B e C presentano metodi non invasivi, mentre il pannello D mostra un metodo di rilevazione dei segnali invasivo. (A) SCP BCI: l'utente impara a controllare gli SCP per muovere un cursore verso un simbolo *target* situato in basso sullo schermo di un pc (SCP positivo) o in alto (SCP negativo). (B) P300 BCI: una matrice di simboli è mostrata all'utente su uno schermo di pc. Solo il simbolo desiderato dall'utente suscita un'ampia onda P300 (potenziale positivo che avviene circa 300 ms dopo il flash). (C) BCI basata sui ritmi sensorimotori: il soggetto tramite onde μ o β controlla un cursore verso un simbolo *target* in alto o in basso sullo schermo di un pc. Spettri di frequenza e tracce EEG. (D) BCI basate su potenziali neuronali corticali: gli elettrodi sono impiantati nella corteccia motoria e rilevano potenziali di azione di singoli neuroni corticali. Fonte: Wolpaw et al., 2002.

Le righe e le colonne lampeggiavano una alla volta ogni 125 ms e l'utente doveva prestare attenzione contando quante volte veniva illuminata la riga o colonna contenente il simbolo scelto (Wolpaw, 2002).

Dato che il lampeggio della riga o colonna scelta rappresentava un evento infrequente, stimolava il potenziale P300, che risulta prominente solo nelle risposte suscitate dalla scelta desiderata, rendendo possibile la determinazio-

ne del simbolo scelto dall'utente.

Una *Brain-Computer Interface* basata sul potenziale evento-relato P300 ha un evidente vantaggio: essa non richiede necessariamente una fase iniziale di *training* da parte dell'utente. Più lo stimolo è infrequente, più la risposta sarà chiara e quindi l'individuazione del simbolo scelto dall'utente sarà accurata. Tuttavia, se il soggetto si abitua allo stimolo la *performance* della BCI potrebbe peggiorare (Ravden e Polich, 1999, Wolpaw et al., 2002).

Nel corso del tempo, si è cercato di migliorare l'accuratezza delle *Brain-Computer Interfaces* basate sul potenziale evento-relato P300 proponendo diverse strategie (Nicolas-Alonso e Gomez-Gil, 2012), tra cui per esempio la variazione della dimensione e del colore dei simboli nella matrice di selezione (Takano et al., 2009).

Per concludere, da un punto di vista operativo e concentrandosi sulla fase di elaborazione dei dati, il problema si riduce ad un'applicazione di tipo statistico, infatti il fine ultimo è distinguere i segnali elettroencefalografici che sottendono un potenziale P300 relativo alla visione di un "evento *target*" da quelli che non lo comprendono: di conseguenza il problema che si fronteggia è una classificazione.

Per questo motivo, in seguito, saranno provate diverse tipologie di classificatori in modo da comprendere quali diano le migliori *performance* nel contesto delle *Brain-Computer Interfaces* basate sul potenziale evento-relato P300.

Capitolo 2

Geometria Riemanniana

Nell'ambito delle applicazioni della *Brain-Computer Interface* soprattutto quelle più recenti, viene fatto uso di tecniche che lavorano su uno spazio diverso da quello euclideo: esse si basano sulla geometria differenziale, utilizzando, nello specifico, metodologie appartenenti al ramo della geometria Riemanniana.

Infatti, per rendere l'analisi dei dati EEG più immediata, vale a dire, senza dover filtrare spazialmente i segnali rilevati, è possibile utilizzare direttamente le matrici di covarianza spaziale dei segnali stimate usando le *matrici di covarianza campionaria* (SCM - *Sample Covariance Matrix*) $\mathbf{P}_i \in \mathbb{R}^{n \times n}$, dove n indica il numero di canali usati per la rilevazione dei segnali EEG.

Ponendo $\mathbf{x}_t \in \mathbb{R}^n$ vettore del segnale EEG in un certo punto temporale t e $\mathbf{X}_i = [\mathbf{x}_{t+T_i} \dots \mathbf{x}_{t+T_i+T_s-1}] \in \mathbb{R}^{n \times T_s}$ matrice contenente segmenti di breve durata temporale del segnale EEG registrato (i cosiddetti *trials*), esse possono essere calcolate come indicato in seguito:

$$\mathbf{P}_i = \frac{1}{T_s - 1} \mathbf{X}_i \mathbf{X}_i^T \quad (2.1)$$

Dato il numero delle osservazioni T_s molto più grande del numero degli elettrodi n , si ha che le SCM sono uno stimatore non distorto delle matrici di covarianza $\Sigma = E\{(\mathbf{x}_t - E\{\mathbf{x}_t\})(\mathbf{x}_t - E\{\mathbf{x}_t\})^T\}$ (Barachant et al., 2012).

2.1 Concetti e strumenti

Le matrici di covarianza campionaria appena citate appartengono ad uno spazio riconducibile ad una varietà Riemanniana, cioè uno spazio curvo di dimensione arbitraria, dotato di una propria metrica¹ e per questo è possibile risolvere il nostro problema di classificazione direttamente con un approccio basato sulla geometria Riemanniana. Per avere una visione più chiara, di seguito verranno introdotti i concetti principali, seguiti da alcune tecniche utili alla classificazione.

Indicando con $S(n) = \{\mathbf{S} \in M(n), \mathbf{S}^T = \mathbf{S}\}$ lo spazio di tutte le matrici quadrate $n \times n$ appartenenti allo spazio delle matrici reali quadrate $M(n)$, è possibile denotare con $P(n) = \{\mathbf{P} \in S(n), \mathbf{u}^T \mathbf{P} \mathbf{u} > 0, \forall \mathbf{u} \in \mathbb{R}\}$ l'insieme delle matrici $n \times n$ simmetriche e definite positive (SPD), sempre diagonalizzabili con valori singolari reali. Le matrici di covarianza campionaria appartengono a questo spazio.

Le matrici SPD hanno diverse proprietà, tra le quali (Barachant et al., 2012):

- $\forall \mathbf{P} \in P(n), \mathbf{P}^{-1} \in P(n)$
- $\forall (\mathbf{P}_1, \mathbf{P}_2) \in P(n)^2, \mathbf{P}_1 \mathbf{P}_2 \in P(n)$
- $\forall \mathbf{P} \in P(n), \log(\mathbf{P}) \in S(n)$
- $\forall \mathbf{S} \in S(n), \exp(\mathbf{S}) \in P(n)$

dove $\log(\bullet)$ e $\exp(\bullet)$ indicano rispettivamente il logaritmo e la forma esponenziale di una matrice, ottenute usando la scomposizione a valori singolari della stessa, date rispettivamente da:

$$\log(\mathbf{P}) = \mathbf{U} \text{diag}(\log(\sigma_1), \dots, \log(\sigma_n)) \mathbf{U}^T$$

e da:

$$\exp(\mathbf{P}) = \mathbf{U} \text{diag}(\exp(\sigma_1), \dots, \exp(\sigma_n)) \mathbf{U}^T$$

¹La metrica Riemanniana, cioè una famiglia di prodotti interni che associa uno scalare reale strettamente positivo ad una coppia di vettori non nulli e tangenti ad un punto della varietà.

Inoltre da notare è il fatto che $P(n)$ forma una varietà differenziabile riemanniana di dimensioni $n^* = n(n+1)/2$ (Förstner e Moonen, 1999). Anche lo spazio tangente ad una matrice $\mathbf{P}$ ha le stesse dimensioni, ed è uno spazio euclideo.

2.1.1 Distanza Riemanniana

Sullo spazio curvo della varietà è possibile anche avere una misura della distanza tra due punti ad essa appartenenti, chiamata *distanza geodetica*.

Chiamando $\Gamma(t) : [0, 1] \rightarrow P(n)$ ogni percorso differenziabile da $\Gamma(0) = \mathbf{P}_1$ a $\Gamma(1) = \mathbf{P}_2$, con $\mathbf{P}_1, \mathbf{P}_2 \in P(n)$ si ha che la lunghezza di $\Gamma(t)$ è data da:

$$L(\Gamma(t)) = \int_0^1 \|\dot{\Gamma}(t)\|_{\Gamma(t)} dt \quad (2.2)$$

dove $\dot{\Gamma}(t) = \frac{d\Gamma}{dt}$ e $\|\bullet\|_{\Gamma(t)}$ indica la norma indotta dal prodotto interno proprio dello spazio tangente a $\Gamma(t)$, così definita: $\langle \mathbf{S}, \mathbf{S} \rangle_{\Gamma(t)} = \|\mathbf{S}\|_{\Gamma(t)}^2 = \text{tr}(\mathbf{S}\Gamma(t)^{-1}\mathbf{S}\Gamma(t)^{-1})$ (*metrica naturale* nella varietà delle matrici SPD)².

La curva, tra tutte quelle che connettono due punti sulla varietà, con minore lunghezza è detta *geodetica*, ed è data dalla seguente formula (Said et al., 2015):

$$\gamma(t) = \mathbf{P}_1^{\frac{1}{2}} (\mathbf{P}_1^{-\frac{1}{2}} \mathbf{P}_2 \mathbf{P}_1^{-\frac{1}{2}})^t \mathbf{P}_1^{\frac{1}{2}} \quad t \in [0, 1] \quad (2.3)$$

La distanza *geodetica* Riemanniana, chiamata distanza di Rao, indotta dalla metrica naturale, è quindi data dalla sua lunghezza (Moakher, 2005):

$$\delta_R(\mathbf{P}_1, \mathbf{P}_2) = \|\log(\mathbf{P}_1^{-1} \mathbf{P}_2)\|_F = \left[\sum_{i=1}^n \log^2 \lambda_i \right]^{\frac{1}{2}} \quad (2.4)$$

con $\lambda_i, i = 1, \dots, n$ autovalori di $\mathbf{P}_1^{-1} \mathbf{P}_2$.

Essa, alternativamente, può essere calcolata in questo modo:

$$\delta_R(\mathbf{P}_1, \mathbf{P}_2) = \text{tr}(\log(\mathbf{P}_1^{-\frac{1}{2}} \mathbf{P}_2 \mathbf{P}_1^{-\frac{1}{2}})) \quad (2.5)$$

²Se si considera $\Gamma(t) = \mathbf{I}$, allora la norma si semplifica in quella di Frobenius $\langle \mathbf{S}, \mathbf{S} \rangle_{\Gamma(t)=\mathbf{I}} = \|\mathbf{S}\|_F^2 = \text{tr}(\mathbf{S}\mathbf{S}^T)$.

La distanza geodetica Riemanniana ha delle proprietà importanti:

- $\delta_R(\mathbf{P}_2, \mathbf{P}_1) = \delta_R(\mathbf{P}_1, \mathbf{P}_2)$;
- $\delta_R(\mathbf{P}_1^{-1}, \mathbf{P}_2^{-1}) = \delta_R(\mathbf{P}_1, \mathbf{P}_2)$;
- $\delta_R(\mathbf{W}^T \mathbf{P}_1 \mathbf{W}, \mathbf{W}^T \mathbf{P}_2 \mathbf{W}) = \delta_R(\mathbf{P}_1, \mathbf{P}_2), \forall \mathbf{W} \in Gl(n)$, con $Gl(n)$ insieme di tutte le matrici $n \times n$ invertibili in $M(n)$ (*invarianza di proiezione*).

2.1.2 Proiezioni nello spazio tangente

Per passare dalla varietà allo spazio tangente è possibile usare la *mappatura logaritmica*:

$$\log_{\mathbf{P}}(\mathbf{P}_i) = \mathbf{S}_i = \mathbf{P}^{\frac{1}{2}} \log(\mathbf{P}^{-\frac{1}{2}} \mathbf{P}_i \mathbf{P}^{-\frac{1}{2}}) \mathbf{P}^{\frac{1}{2}} \quad (2.6)$$

la cui operazione inversa è la *mappatura esponenziale*:

$$\exp_{\mathbf{P}}(\mathbf{S}_i) = \mathbf{P}_i = \mathbf{P}^{\frac{1}{2}} \exp(\mathbf{P}^{-\frac{1}{2}} \mathbf{S}_i \mathbf{P}^{-\frac{1}{2}}) \mathbf{P}^{\frac{1}{2}} \quad (2.7)$$

Una rappresentazione della procedura di proiezione nello spazio tangente e dell'operazione inversa si può trovare in Figura 2.1

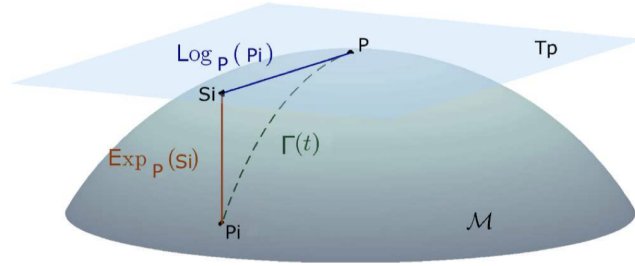


Figura 2.1: Spazio tangente al punto $\mathbf{P}$, con $\mathbf{S}_i$ vettore tangente e $\Gamma_i(t)$ distanza geodetica tra $\mathbf{P}$ e $\mathbf{P}_i$. Fonte: Barachant et al., 2012.

Da notare è il fatto che ogni vettore $\mathbf{S}_i$ appartenente allo spazio tangente calcolato su un punto $\mathbf{P}$ della varietà, può essere visto come la derivata a $t = 0$ della geodetica $\Gamma_i(t)$ tra $\mathbf{P}$ e la mappatura esponenziale $\mathbf{P}_i = \exp_{\mathbf{P}}(\mathbf{S}_i)$.

2.1.3 Media delle matrici SPD

Possiamo considerare due tipologie di media nell'ambito delle matrici SPD: la *media euclidea* e quella *Riemanniana*.

- La media euclidea, anche detta *aritmetica*, è calcolata come segue:

$$\mathcal{U}(\mathbf{P}_1, \dots, \mathbf{P}_I) = \arg \min_{\mathbf{P} \in P(n)} \sum_{i=1}^I \delta_E^2(\mathbf{P}, \mathbf{P}_i) = \frac{1}{I} \sum_{i=1}^I \mathbf{P}_i \quad (2.8)$$

dove $\delta_E^2(\mathbf{P}_1, \mathbf{P}_2) = \|\mathbf{P}_1 - \mathbf{P}_2\|_F$ rappresenta la distanza euclidea sulla varietà;

- la media Riemanniana invece usa la distanza geodetica, ed è così determinata:

$$\mathcal{G}(\mathbf{P}_1, \dots, \mathbf{P}_I) = \arg \min_{\mathbf{P} \in P(n)} \sum_{i=1}^I \delta_R^2(\mathbf{P}, \mathbf{P}_i) \quad (2.9)$$

Essa viene anche definita *media geometrica*.

Purtroppo non esiste una forma chiusa per calcolarla, di conseguenza vengono impiegati algoritmi iterativi che proiettano le matrici di covarianza nello spazio tangente nel quale stimano la media aritmetica, che in seguito viene proiettata nuovamente sulla varietà (Barachant et al., 2013). Nello specifico, uno degli algoritmi usati per il suo calcolo è il seguente (Moakher, 2005):

Algoritmo 1 Media geometrica di I matrici SPD

Input: Ω insieme di I matrici SPD $\mathbf{P}_i \in P(n)$ ed $\epsilon > 0$.

Output: $\mathcal{G}$ media geometrica stimata in $P(n)$ dell'insieme delle matrici in Ω .

- 1: Inizializzare $\mathcal{G}^{(1)} = \frac{1}{I} \sum_{i=1}^I \mathbf{P}_i$
 - 2: **repeat**
 - 3: $\bar{\mathbf{S}} = \frac{1}{I} \sum_{i=1}^I \log_{\mathcal{G}^{(t)}}(\mathbf{P}_i)$ Media aritmetica nello spazio tangente
 - 4: $\mathcal{G}^{(t+1)} = \exp_{\mathcal{G}^{(t)}}(\bar{\mathbf{S}})$
 - 5: **until** $\|\bar{\mathbf{S}}\|_F < \epsilon$
 - 6: **return** $\mathcal{G}^{(t+1)}$
-

2.2 Distribuzione di Riemann-Gauss

Per catturare le caratteristiche da un punto di vista statistico dei dati che giacciono su spazi riconducibili a varietà riemanniane, Said et al. (2015) hanno proposto una generalizzazione della distribuzione Normale, in modo da estendere le sue proprietà e da permettere l'applicazione delle consuete procedure inferenziali in uno spazio diverso da quello euclideo.

La distribuzione di probabilità proposta è chiamata *distribuzione di Riemann-Gauss*, $G(\bar{\mathbf{Y}}, \sigma)$, è definita in $P(n)$ e dipende da due parametri: $\bar{\mathbf{Y}} \in P(n)$, il baricentro Riemanniano, e $\sigma > 0$, il parametro di dispersione in termini di distanza quadratica da $\bar{\mathbf{Y}}$.

La sua funzione di densità di probabilità è la seguente:

$$p(\mathbf{Y}|\bar{\mathbf{Y}}, \sigma) = \frac{1}{\zeta(\bar{\mathbf{Y}}, \sigma)} \exp \left[-\frac{\delta_R^2(\mathbf{Y}, \bar{\mathbf{Y}})}{2\sigma^2} \right] \quad (2.10)$$

dove $\delta_R(\mathbf{Y}, \bar{\mathbf{Y}})$ è la distanza di Rao specificata in equazione (2.5) e $\zeta(\bar{\mathbf{Y}}, \sigma)$ rappresenta la costante di normalizzazione.

Per la corretta definizione della distribuzione, è necessario fornire l'esatta espressione della costante $\zeta(\bar{\mathbf{Y}}, \sigma)$: essa infatti ha lo scopo di normalizzare l'integrale della funzione di densità di probabilità in modo che, se integrata sull'intero spazio campionario, restituisca un risultato unitario.

Quindi, è possibile definire $\zeta(\bar{\mathbf{Y}}, \sigma)$ nel modo seguente:

$$\zeta(\bar{\mathbf{Y}}, \sigma) = \int_{P(n)} \exp \left[-\frac{\delta_R^2(\mathbf{Y}, \bar{\mathbf{Y}})}{2\sigma^2} \right] dv(\mathbf{Y}) \quad (2.11)$$

con $dv(\mathbf{Y}) = \det(\mathbf{Y})^{-\frac{m+1}{2}} \prod_{i \leq j} dY_{ij}$, indicante la misura di volume associata alla metrica di Rao-Fisher.

Per riuscire ad ottenere un'espressione analitica dell'integrale in equazione (2.11), risulta necessario esprimere la seguente proposizione:

Proposizione 1. *Per ogni $\bar{\mathbf{Y}} \in P(n)$ e $\sigma > 0$,*

$$\zeta(\bar{\mathbf{Y}}, \sigma) = \zeta(\mathbf{I}, \sigma)$$

dove $\mathbf{I} \in P(n)$ è una matrice identità di dimensione $n \times n$.

Ponendo $\zeta(\sigma) = \zeta(\mathbf{I}, \sigma)$, allora si ha che:

$$\zeta(\sigma) = (n!2^n)^{-1} \times \omega_n \times 8^{\frac{n(n-1)}{4}} \int_{\mathbb{R}^n} \exp \left[\frac{-(r_1^2 + \dots + r_n^2)}{2\sigma^2} \right] \prod_{i < j} \sinh \left(\frac{|r_i - r_j|}{2} \right) \prod_{i=1}^n dr_i \quad (2.12)$$

dove ω_n è data da:

$$\omega_n = \frac{2^n \pi^{n^2/2}}{\Gamma_n(n/2)}$$

in cui Γ_n indica la funzione gamma multivariata.

Per la dimostrazione completa si veda Said et al. (2015).

Dalla Proposizione 1 è possibile infine notare che la costante di normalizzazione non dipende da $\bar{\mathbf{Y}}$, ma solo da σ .

Per quanto riguarda la stima dei parametri, si osserva che la distanza di Rao gioca un ruolo essenziale nei problemi d'inferenza statistica tramite questa distribuzione. Di seguito sono riportate delle informazioni sulla stima dei parametri

La proposizione seguente permette di capire il processo di calcolo delle stime di massima verosimiglianza di $\bar{\mathbf{Y}}$ e σ (Said et al., 2015).

Proposizione 2. *Siano $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N$ osservazioni indipendenti dalla distribuzione di Riemann-Gauss $G(\bar{\mathbf{Y}}, \sigma)$.*

La stima di massima verosimiglianza di $\bar{\mathbf{Y}}$ è data dal baricentro Riemanniano empirico (anche chiamato media geometrica o media Riemanniana) $\hat{\mathbf{Y}}_N$ di $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N$.

Esso è la matrice $\hat{\mathbf{Y}}_N \in P(n)$ che minimizza $\mathcal{E}_N : P(n) \rightarrow \mathbb{R}_+$, chiamata funzione di varianza empirica:

$$\mathcal{E}_N(\mathbf{Y}) = \frac{1}{N} \sum_{n=1}^N \delta_R^2(\mathbf{Y}, \mathbf{Y}_n)$$

vale a dire $\hat{\mathbf{Y}}_N = \arg \min_{\mathbf{Y}} \mathcal{E}_N(\mathbf{Y})$.

Inoltre, la stima di massima verosimiglianza del parametro σ è la soluzione, indicata con $\hat{\sigma}_N$, della seguente equazione:

$$\sigma^3 \times \frac{d}{d\sigma} \log \zeta(\sigma) = \mathcal{E}_N(\hat{\mathbf{Y}}_N) \quad (2.13)$$

Entrambi i parametri esistono unici per ogni realizzazione del campione $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N$: il fatto che la varietà Riemanniana considerata sia uno spazio con curvatura negativa assicura che la stima di massima verosimiglianza esista e sia unica.

La stima dei due parametri è provata dal fatto che, usando l'indipendenza di $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N$, è possibile scrivere la funzione di log-verosimiglianza in questo modo:

$$l(\hat{\mathbf{Y}}_N, \sigma) = -N \log \zeta(\sigma) - \frac{1}{2\sigma^2} \sum_{n=1}^N \delta_R^2(\hat{\mathbf{Y}}_N, \mathbf{Y}_n) \quad (2.14)$$

Dato che il primo termine a destra non dipende da $\bar{\mathbf{Y}}$, la stima di massima verosimiglianza del baricentro riemanniano si trova massimizzando il secondo termine, $-\frac{1}{2\sigma^2} \sum_{n=1}^N \delta_R^2(\hat{\mathbf{Y}}, \mathbf{Y}_n)$, quindi minimizzando la somma delle distanze al quadrato.

La quantità che lo minimizza è appunto $\hat{\mathbf{Y}}_N$, che, come attesta la proposizione che segue, esiste ed è unica (Said et al., 2015).

Proposizione 3. *Se π è una distribuzione di probabilità su $P(n)$, allora π ha un unico baricentro riemanniano $\bar{\mathbf{Y}}_\pi$. Inoltre, $\bar{\mathbf{Y}}_\pi$ è l'unico punto stazionario della funzione di varianza*

$$\mathcal{E}_\pi(\mathbf{Y}) = \int_{P(n)} \delta_R^2(\mathbf{Y}, \mathbf{Z}) d\pi(\mathbf{Z})$$

con $\mathbf{P}, \mathbf{Z} \in P(n)$.

Per quanto riguarda la stima di massima verosimiglianza di σ , è possibile riparametrizzare la funzione di log-verosimiglianza ponendo $\eta = -\frac{1}{2\sigma^2}$

$$\frac{1}{N}l(\hat{\mathbf{Y}}_N, \sigma) = \eta \frac{1}{N} \sum_{n=1}^N \delta_R^2(\hat{\mathbf{Y}}_N, \mathbf{Y}_n) - \psi(\eta) \quad (2.15)$$

dove $\psi(\eta) = \log \zeta(\sigma)$, in modo da ottenere:

$$\hat{\eta}_N = \arg \max_{\eta} \left\{ \eta \frac{1}{N} \sum_{n=1}^N \delta_R^2(\hat{\mathbf{Y}}_N, \mathbf{Y}_n) - \psi(\eta) \right\} \quad (2.16)$$

Ovviamente $\psi'(\eta)$ è una funzione strettamente crescente, data la natura strettamente convessa di $\psi(\eta)$: $\hat{\eta}_N$ quindi può essere trovato come soluzione di

$$\psi'(\eta) = \mathcal{E}_N(\hat{\mathbf{Y}}_N) \quad (2.17)$$

che è equivalente all'espressione (2.13).

Infine, $\psi'(\eta)$ essa tende a zero quando $\eta \rightarrow -\infty$ e tende ad infinito quando $\eta \rightarrow 0$. Tutto ciò implica che l'equazione $\psi'(\eta) = c$ ha un'unica soluzione $\forall c > 0$ (Said et al., 2015).

Ottenuto $\hat{\eta}_N$, è possibile ottenere facilmente la stima di σ grazie alla proprietà di equivarianza della stima di massima verosimiglianza, e quindi sostituirla nell'espressione (2.12) in modo da ottenere la costante di normalizzazione $\zeta(\hat{\sigma}_N)$.

Capitolo 3

Metodi di classificazione

Nel seguente capitolo verranno descritte da un punto di vista teorico le metodologie di classificazione utilizzate per le analisi presenti in questa tesi. Esso si divide in due principali sezioni: la prima in cui vengono descritti i metodi di *statistical learning* utilizzati, e la seconda in cui vengono presentati gli algoritmi basati sulla geometria Riemanniana e quindi sui concetti presentati nel precedente capitolo.

3.1 Metodi di *statistical learning*

In questa sezione vengono presentati alcuni metodi di *statistical learning* che saranno applicati nel Capitolo 5. Sono stati considerati quattro metodi: le *Support Vector Machines* e metodologie che utilizzano combinazioni di classificatori, le *Model-Based Oblique Random Forests*, il *Bagging* ed infine l'*eXtreme Gradient Boosting*, data la loro buona adattabilità a svariate situazioni.

3.1.1 *Support Vector Machines*

Le *Support Vector Machines* (SVM) compongono una classe di modelli supervisionati per lo *statistical learning*, particolarmente utilizzata quando si dispone di insiemi di dati complessi.

Cortes e Vapnik ne hanno proposto una prima versione dedicata a problemi di classificazione, chiamandola *support-vector network* (Cortes e Vapnik, 1995). Si assuma che l'insieme di stima sia composto da N coppie $(x_1, y_1), \dots, (x_N, y_N)$, con $x_i \in \mathbb{R}^p$ e $y_i \in \{-1, 1\}$. La ragione geometrica sottostante lo sviluppo delle SVM per problemi di classificazione prevede la presenza di un iperpiano così definito:

$$\{x : f(x) = x^T \beta + \beta_0 = 0\} \quad (3.1)$$

dove β è un vettore di norma unitaria, $\|\beta\| = 1$, che discrimina nel modo migliore le osservazioni presenti, cioè che riesce a separare nei due ipersemitrassi di $\mathbb{R}^p$ le osservazioni x_i presenti. In sostanza, quindi si vuole cercare l'iperpiano che crea il più grande margine tra i punti delle due diverse classi (Figura 3.1, pannello di sinistra).

Il problema di ottimizzazione che si va ad affrontare è quindi il seguente:

$$\max_{\beta, \beta_0, \|\beta\|=1} M \quad \text{s.v. } y_i(x_i^T \beta + \beta_0) \geq M \quad i = 1, \dots, N \quad (3.2)$$

che può essere riformulato nella maniera seguente, eliminando il vincolo di norma unitaria sul vettore dei coefficienti β :

$$\min_{\beta, \beta_0} \|\beta\| \quad \text{s.v. } y_i(x_i^T \beta + \beta_0) \geq 1 \quad i = 1, \dots, N, \quad M = \frac{1}{\|\beta\|} \quad (3.3)$$

Il problema è così trasformato in un'ottimizzazione convessa (Hastie, Tibshirani, Friedman 2009). Supponendo ora che le due classi si sovrappongano nello spazio $\mathbb{R}^p$ considerato, un modo per risolvere il problema è massimizzare M , ma questo permetterebbe ad alcuni punti di essere presenti nell'ipersottospazio sbagliato. Si definiscono quindi le variabili ausiliarie $\xi = (\xi_1, \dots, \xi_N)$ modificando il vincolo indicato in (3.2):

$$y_i(x_i^T \beta + \beta_0) \geq M - \xi_i \quad (3.4)$$

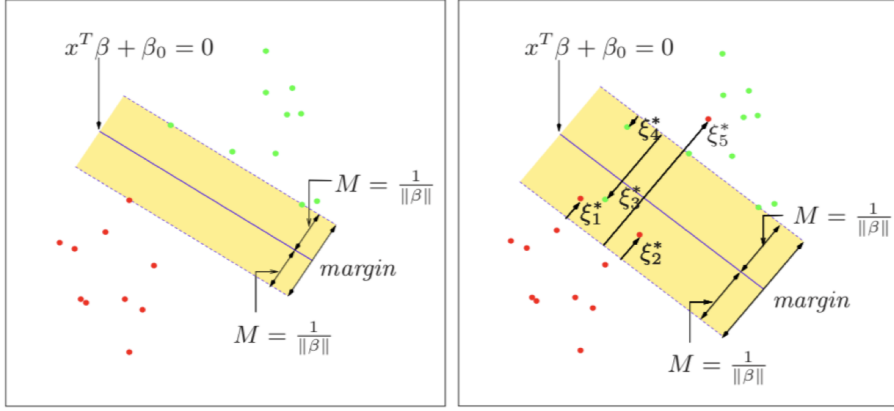


Figura 3.1: Esempio di classi linearmente separabili (pannello di sinistra) e classi sovrapposte (pannello di destra). Fonte: Hastie, Tibshirani, Friedman, 2009.

o, in alternativa:

$$y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i) \quad \forall i, \xi_i \geq 0, \sum_{i=1}^N \xi_i \leq c \quad (3.5)$$

La scelta tra (3.4) e (3.5) porta a differenti soluzioni: se la prima sembra più naturale in quanto misura l'effettiva distanza dal margine, la seconda misura la distanza relativa, che cambia al variare dell'ampiezza del margine M . La seconda equazione è la scelta "standard" nell'ambito delle *Support Vector Machines* in quanto porta ad un problema di ottimizzazione convessa, mentre la (3.4) risulta in un'ottimizzazione non convessa. Si preferisce quindi utilizzare la (3.5).

Il valore ξ_i presente nel vincolo è proporzionale alla misura in cui l'osservazione x_i si trova nel lato opposto dell'estremo del margine relativo alla propria classe di appartenenza (Figura 3.1, pannello di destra): infatti, imponendo che $\sum_{i=1}^N \xi_i \leq c$, con c valore fissato, stiamo imponendo un limite superiore alla proporzione totale di osservazioni che giacciono nel lato opposto definito dal margine.

La non corretta classificazione avviene quando $\xi_i > 1$, di conseguenza il vincolo impone in maniera implicita che al massimo possano essere classificate

erroneamente c osservazioni dell'insieme di stima (Hastie, Tibshirani, Friedman, 2009).

Il vincolo di norma unitaria del vettore β può essere eliminato scrivendo

$$\frac{1}{\|\beta\|} y_i(x_i^T \beta + \beta_0) \geq M$$

cioè come

$$y_i(x_i^T \beta + \beta_0) \geq M \|\beta\|$$

che implica la modifica di β_0 . Dato che la moltiplicazione di β e β_0 per una costante positiva non altera i vincoli, è possibile porre la condizione $\|\beta\| = \frac{1}{M}$, riscrivendo la (3.3) in questo modo:

$$\min_{\beta, \beta_0} \|\beta\| \quad \text{s.v.} \quad \begin{aligned} y_i(x_i^T \beta + \beta_0) &\geq 1 - \xi_i \quad \forall i \\ \xi_i &\geq 0, \sum \xi_i \leq c \end{aligned} \quad i = 1, \dots, N \quad (3.6)$$

L'equazione appena indicata, come specificato precedentemente, costituisce un problema di ottimizzazione convessa. Computazionalmente parlando, è conveniente riscriverla nel modo seguente

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 + c \sum_{i=1}^N \xi_i \quad \text{s.v.} \quad \begin{aligned} y_i(x_i^T \beta + \beta_0) &\geq 1 - \xi_i \quad \forall i, \\ \xi_i &\geq 0 \end{aligned} \quad (3.7)$$

in cui c rappresenta il parametro di costo di violazione delle barriere. Nel caso di due classi linearmente separabili si ha $c = \infty$.

Si può dimostrare che la soluzione al problema riportato in (3.7) è del tipo

$$\hat{\beta} = \sum_{i=1}^N \hat{\alpha}_i y_i x_i \quad (3.8)$$

dove solo alcuni $\hat{\alpha}_i$ sono non nulli; la soluzione è quindi espressa solo da alcune delle osservazioni, quelle che rispettano la condizione $y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$, dette *support vectors* (Azzalini, Scarpa, 2012).

Può costituire un vantaggio prendere in considerazione trasformazioni delle

variabili esplicative, come ad esempio $h(x) = (h_1(x), \dots, h_q(x))^T$, ($x \in \mathbb{R}^p$) in cui p può essere minore, uguale o maggiore p .

Inserendo l'espressione (3.8) nell'equazione dell'iperpiano separatore $f(x) = h(x)^T \beta + \beta_0$, è possibile esprimere la soluzione come:

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^N \hat{\alpha}_i y_i \langle h(x), h(x_i) \rangle \quad (3.9)$$

dove $\langle h(x), h(x_i) \rangle = h(x)^T h(x_i)$ indica un prodotto interno. La metodologia risultante prende il nome di *support vector machines*, o SVM.

Quindi, la regola decisionale secondo il quale la generica osservazione $\tilde{x}$ verrà classificata è la seguente:

$$G(\tilde{x}) = \text{sign}(\hat{f}(\tilde{x})) = \text{sign}(\hat{\beta}_0 + \sum_{i=1}^N \hat{\alpha}_i y_i \langle h(\tilde{x}), h(x_i) \rangle) \quad (3.10)$$

Come è possibile notare, le osservazioni entrano nelle formule solo attraverso i prodotti interni, di conseguenza non è necessario specificare la trasformazione $h(x)$ ma è sufficiente solo la definizione della funzione *kernel*

$$K(x, x') = \langle h(x), h(x') \rangle$$

che calcola i prodotti interni nello spazio trasformato.

Le tre funzioni nucleo più utilizzate sono riportate in Tabella 3.1:

Nucleo	$K(x, x')$
polinomiale	$(1 + \langle x, x' \rangle)^d$
base radiale	$\exp(-d \ x - x'\ ^2)$
sigmoidale	$\tanh(d_1 \langle x, x' \rangle + d_2)$

Tabella 3.1: Tipologia di funzioni *kernel* usate nelle SVM.

dove d, d_1 e d_2 sono quantità specificate a priori.

3.1.2 *Model-based Oblique Random Forest*

Le *Random Forest* al giorno d'oggi sono molto popolari in problemi di classificazione e regressione in cui le osservazioni giacciono su spazi ad elevata dimensionalità, come nell'analisi dei *micro-array* o il riconoscimento di oggetti nei problemi di *computer vision*, grazie anche alla loro capacità di discriminare i sottoinsiemi di variabili importanti per il particolare problema preso in considerazione.

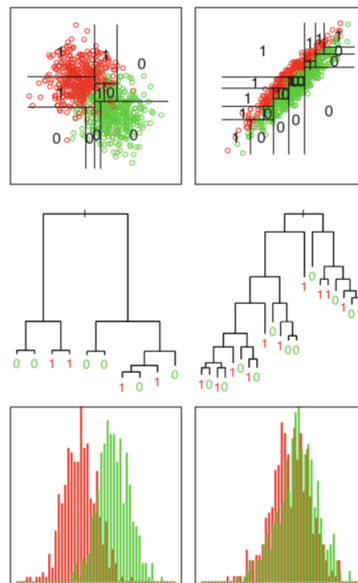


Figura 3.2: Esempio di separazione di dati correlati tramite alberi decisionali. Cambiamento dell'albero di decisione a seconda della correlazione delle classi. Fonte: Menze et al., 2011.

In questa sezione sono considerate delle foreste casuali composte da alberi "obliqui" regolarizzati. I recenti risultati sulla potatura dell'*ensemble* hanno mostrato un vantaggio nella scelta dei classificatori base che sono adattati in modo ottimale ai dati (Menze et al., 2011, Martínez-Muñoz et al., 2008). Quindi, Menze et al. (2011) hanno proposto di concentrarsi sugli alberi caratterizzati da un ottimale partizionamento ricorsivo, usando un modello discriminativo "forte" in ciascun nodo per ottenere regole di decisione

complessivamente migliori. Questo approccio è anche correlato alle *rotation forests* (Rodriguez et al., 2006) in cui si cercano direzioni di divisione oblique dai componenti principali dei sottogruppi di caratteristiche dei dati di addestramento, migliorando, secondo quanto riferito, i risultati in modo significativo nelle attività di classificazione selezionate (Menze et al., 2011). Nelle *oblique random forest*, invece di utilizzare l'approccio seguito da Rodriguez et al. (2006), vengono usati modelli regolarizzati per definire le direzioni di divisione ottimali negli *split*, permettendo l'integrazione delle buone proprietà delle foresta casuali originali di Breiman (2005) per classificare i dati ad alta dimensione con pochi campioni, un dominio in cui la foresta casuale con alberi ortogonali si comporta in modo eccezionalmente buono.

Per ricapitolare, le *oblique random forest* cercano un iperpiano obliquo (Figura 3.2) in ogni nodo interno in base ad una combinazione lineare delle variabili di un sottoinsieme scelto in modo casuale (Menze et al., 2011).

Costruzione dell'*oblique random forest*

Anche l'*oblique random forest* (oRF) è generata come la random forest tradizionale, cambia solamente il modo in cui sono trovate le direzioni degli *split* ottimali per ogni nodo dell'albero.

Nelle oRF gli *split* binari sono trovati mediante modelli multivariati: per un campione $\mathbf{x} = [x_1, \dots, x_{m_{try}}]^T$ in uno spazio m_{try} -dimensionale, la decisione f al nodo m può essere trovata nel seguente modo:

$$f_m(\mathbf{x}) : \beta_m^T > c_m \quad (3.11)$$

in cui i coefficienti β_m definiscono la proiezione per lo *split* (il sottospazio lineare ottimo) e c_m rappresenta la soglia. La maggiore difficoltà risiede nel trovare il vettore dei coefficienti β_m : esso può essere identificato in diversi modi, come la regressione logistica o l'analisi discriminante lineare (sfruttando l'informazione delle etichette delle classi), o cercando un ottimo tra l'informazione apportata dalle classi e lo spazio campionario, mediante regressione lineare con metodo di penalizzazione ridge o *support vector machines* con kernel lineare, per esempio.

Dato che metodi di regressione penalizzata forniscono buoni risultati in dati ad elevate dimensioni, Menze et al. (2011) hanno scelto di utilizzare come modello per la scelta dei coefficienti β_m una regressione lineare con penalizzazione ridge:

$$\beta_{ridge}(\lambda) \sim \arg \min_{\beta} \sum_{i=1}^N \left(y_i - \sum_{j=1}^P x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^P \beta_j^2 \quad (3.12)$$

in cui λ rappresenta il parametro di regolarizzazione, che permette al classificatore di adattarsi alla direzione ottimale data da β_{ridge} : con $\lambda = 0$, β_{ridge} si dirige verso la massima correlazione con le etichette delle classi, come l'analisi discriminante lineare (LDA), mentre con $\lambda \gg 1$ punta verso la massimizzazione della variabilità dei dati, come con l'analisi delle componenti principali (PCA), come è possibile notare in Figura 3.3.

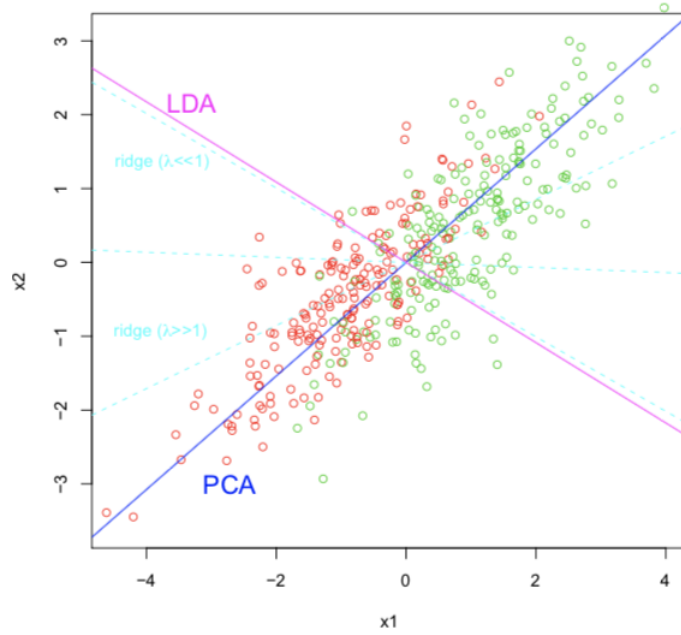


Figura 3.3: Effetto della regolarizzazione sulle direzioni degli split. Fonte: Menze et al., 2011.

Ad ogni nodo m il parametro λ viene adattato nei campioni *out-of-bag* disponibili; in seguito viene identificato il punto ottimale di split c_m usando l'indice di eterogeneità di Gini:

$$I(s_m) = \frac{1}{2} \times (1 - \sum_{k=0,1} P_k^2(s_m)) \quad (3.13)$$

con $s_m = X_m \beta_m$.

Il valore che massimizza la decrescita in eterogeneità nell'indice indicato in (3.13) è c_m , cioè il minimo in $I(s_m < c_m) + I(s_m > c_m)$: una volta trovato il valore vengono separate le osservazioni, e per ogni sottoinsieme, il procedimento viene infine iterato ricorsivamente finché le classi sono separate.

In sostanza, la generazione dell'oRF è così scandita:

1. sono creati tramite campionamento con reinserimento n_{tree} campioni di osservazioni a partire dai dati originali. In ogni campione viene fatto crescere un albero;
2. per ogni nodo dell'albero vengono campionate m_{try} variabili esplicative in modo casuale e senza reinserimento;
3. ad ogni split le variabili presenti sono riportate a media zero e varianza unitaria, in modo da aumentare la stabilità del modello;
4. viene scelto tramite *grid-search*¹ il parametro λ che è ottimizzato usando le osservazioni *out-of-bag* del nodo corrente. In seguito le osservazioni sono proiettate nell'iperpiano dato dai parametri β_m determinati dalla regressione lineare penalizzata adattata ad ogni nodo;
5. vengono cercate le soglie ottimali per la separazione dei dati utilizzando l'indice d'impurità di Gini e una volta trovate, i dati vengono separati;
6. per ognuno degli n_{tree} alberi, vengono ripetuti iterativamente gli step 2-5 finché non sono più disponibili osservazioni *out-of-bag* per adattare ulteriormente λ .

¹ $\lambda = 10^i$, con $i = \{-5, -4, \dots, 4, 5\}$.

Per concludere, è stato dimostrato che le oRF danno risultati molto validi in un'ampia gamma di insiemi di dati, dalle serie storiche, ai dati di *micro-array* e fMRI, comportandosi meglio delle tradizionali random forest e di molti altri classificatori alternativi, dando il meglio nella separazione di distribuzioni ad elevate dimensioni, anche quando le variabili esplicative presentano molta correlazione (Menze et al., 2011).

3.1.3 *Bagging*

Solitamente il *bootstrap* è utilizzato per valutare l'accuratezza della stima di parametri o di previsioni, ma esso può essere impiegato anche per migliorare le stime o le previsioni stesse. In seguito è possibile vedere la procedura da seguire.

Si consideri un classificatore $C(x)$, un insieme di stima $\mathbf{Z} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ e, per semplicità, un problema di classificazione a due classi, $K = 2$, con etichette pari a $y_i \in \{0, 1\}$.

Campionando con reinserimento n unità dell'insieme di stima originale e ripetendo il procedimento B volte, si possono formare B campioni *bootstrap* di n unità ciascuno. Il classificatore $C(x)$ sarà quindi applicato ad ognuno dei B campioni, ottenendo così $C_B^*(x)$, classificatori differenti.

Viene così introdotto un nuovo classificatore che costituisce, per ogni x , un indicatore medio dei risultati ottenuti da ciascuno dei B classificatori su quella stessa x . Solitamente si utilizza la media aritmetica

$$C_{bag} = \frac{1}{B} \sum_{b=1}^B C_b^*(x)$$

e si classifica l'unità con variabili esplicative x nella classe associata al valore 1 se $C_{bag}(x) > \frac{1}{2}$ e a quella indicata dal valore zero altrimenti. Quindi, in pratica l'osservazione viene assegnata alla classe per cui si ha il *voto di maggioranza*. Il modello finale, che si ottiene al termine della procedura viene chiamato *bootstrap aggregating*, meglio conosciuto come *bagging* (Azzalini, Scarpa, 2012).

Data la sua costruzione, l'errore di classificazione del modello finale risulta mediamente più basso rispetto a quello di ciascuno dei B classificatori.

Esiste una variante nel calcolo del voto di maggioranza per la classificazione delle osservazioni: essa consiste nel calcolare la media delle probabilità di appartenenza a ciascuna classe di un'unità con variabili esplicative x

$$\hat{p}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{p}_b^*(x)$$

e di usare questa nuova $\hat{p}_{bag}(x)$ come indicatore di appartenenza alla classe. Le procedure di tipo *bagging* in genere migliorano molto la classificazione quando il classificatore $C(x)$ scelto è molto instabile, come l'albero di classificazione o le reti neurali, anche se la combinazione dei risultati tramite la media aritmetica porta alla perdita di ogni semplice struttura dei modelli originari, rendendo più difficile l'interpretazione dei risultati.

Nel *bagging*, infine, è possibile utilizzare la tecnica dell'*out-of-bag* per stimare l'errore di classificazione: dato che per ogni campione *bootstrap* una parte dei dati dell'insieme di stima originale viene esclusa, essa può essere utilizzata come insieme di verifica. Ciò permette di adattare il modello senza ricorrere ad un insieme di verifica o alla convalida incrociata.

3.1.4 *Boosting*

La logica attuata dal *bagging* è ripresa anche dalla tecnica di combinazione di classificatori chiamata *boosting*, solo che, a differenza della prima, essa seleziona le unità da inserire nel campione in base ad una diversa probabilità di entrata. In particolare, si assegna un peso maggiore alle osservazioni che precedentemente sono state classificate erroneamente, cercando così di migliorare la prestazione del nuovo modello.

Il *boosting* quindi è una procedura iterativa, in cui si parte da un classificatore *debole* in cui ogni osservazione ha lo stesso peso; ad ogni iterazione, in base al risultato della classificazione precedente, vengono aggiornati, in funzione del tasso di errore globale ottenuto su un insieme di verifica, i pesi

di entrata assegnati alle osservazioni, ed alla fine si ottiene un nuovo classificatore attraverso voto di maggioranza tra tutti i modelli adattati nel corso delle iterazioni (Azzalini e Scarpa, 2012).

Ci sono diversi algoritmi disponibili per l'applicazione del *boosting*, tra i quali il più diffuso è chiamato *AdaBoost*, introdotto da Freund e Schapire nel 1996, descritto nell'Algoritmo 2:

Algoritmo 2 AdaBoost

Input: un insieme di osservazioni x_i $i = 1, \dots, n$ di K classi differenti e le corrispondenti etichette $y_i = 1, \dots, k$.

Output: il nuovo classificatore $C(x)$.

- 1: Inizializzare i pesi per ciascuna osservazione $w_i = 1/n$, $i = 1, \dots, N$
- 2: **for** $b = 1$ to B **do**
 - (a) adattare un modello di classificazione $C_b(x)$ all'insieme di stima usando i pesi w_i
 - (b) porre

$$\text{err}_b \leftarrow \frac{\sum_{i=1}^N w_i I(y_i \neq C_b(x_i))}{\sum_{i=1}^N w_i}, \quad \alpha_b \leftarrow \log \frac{1 - \text{err}_b}{\text{err}_b}$$

- (c) assegnare i nuovi pesi

$$w_i \leftarrow w_i \exp\{\alpha_b I(y_i \neq C_b(x_i))\}, \quad i = 1, \dots, N$$

end for

- 3: si ottiene il nuovo classificatore

$$C(x) = \begin{cases} 1 & \text{se } \frac{\sum_{b=1}^B \alpha_b C_b(x)}{\sum_{b=1}^B \alpha_b} > \frac{1}{2} \\ 0 & \text{altrimenti} \end{cases}$$

All'aumentare delle iterazioni diminuisce l'importanza della scelta del classificatore utilizzato, perché ci si concentra soprattutto sulle unità classificate peggio; si cerca infatti di utilizzare classificatori semplici, come alberi fatti crescere al massimo di due livelli, senza potatura.

Procedure di tipo *boosting* hanno mostrato una grande capacità di costruire

classificatori accurati in un'ampia gamma di situazioni (Hastie, Tibshirani, Friedman, 2009).

Nel corso degli anni *AdaBoost* è stato migliorato da diversi autori, come per esempio da Breiman (1997), ottenendo il più efficiente *gradient boosting*, che continua tutt'oggi ad essere oggetto di ricerca nell'ambito del *statistical learning*; esso può essere interpretato come un algoritmo di ottimizzazione su una funzione di costo adeguata.

Basandosi su quest'idea Friedman et al. (2001) hanno generalizzato l'algoritmo *Adaboost* per farlo adattare ad una grande varietà di funzioni di costo, seguito dalla creazione di una prospettiva più generale, in cui il *boosting* viene visto come un algoritmo iterativo di discesa del gradiente funzionale (Mason et al., 1999). Il metodo di discesa del gradiente rende possibile infatti la minimizzazione di funzioni complesse in maniera efficiente, per questo motivo è molto utilizzato nelle applicazioni del *data mining*.

Esso, brevemente, consiste nelle seguenti operazioni (Friedman, 2001):

1. scelta della funzione di perdita e di un punto di partenza x_k ;
2. calcolo del valore della funzione di perdita e del suo gradiente in x_k : l'opposto del gradiente valutato nel punto determina la direzione in cui la funzione di perdita scende più rapidamente verso il minimo;
3. aggiornamento di x_k in maniera iterativa: $x_{k+1} = x_k + \alpha \rho_k$, dove α_k rappresenta il parametro che controlla la velocità di discesa del punto di minimo lungo la direzione individuata al passo precedente, indicata da ρ_k ;
4. dopo i iterazioni dei punti 2 e 3, si trova il punto di minimo della funzione di perdita, dato da x_{k+i} .

Sempre in questa direzione, uno dei più recenti lavori è quello di Chen e Guestrin (2016) in cui è stato sviluppato l'algoritmo *XGBoost* o *eXtreme Gradient Boosting*, un'implementazione più veloce da un punto di vista computazionale rispetto ad altre implementazioni del *gradient boosting*, resa possibile grazie al calcolo parallelo e ad una migliore gestione della memoria del calcolatore.

3.2 Metodi basati sulla geometria Riemanniana

La classificazione di dati appartenenti ad uno spazio diverso da quello euclideo, in particolare appartenenti a varietà Riemanniane, rappresenta una delle principali sfide di molte applicazioni ai giorni nostri, come ad esempio la visione computazionale e l'*imaging* medico.

In questa sezione verranno presentate metodologie di classificazione supervisionata basate sulla geometria Riemanniana, in modo da sfruttare appieno lo spazio di appartenenza dei dati.

Per prima cosa sarà introdotta una modifica delle *Support Vector Machines*, in modo da adattarele all'uso delle matrici di covarianza, in seguito verrà specificato un semplice algoritmo basato sul confronto di distanze ed infine sarà descritto un approccio basato sulla verosimiglianza derivante dalla distribuzione di Riemann-Gauss vista nel Paragrafo 2.2.

3.2.1 *Support Vector Machines* con kernel Riemanniano

Le *support vector machines* sono molto usate nelle applicazioni di Brain-Computer Interface data la loro buona performance in presenza di dati ad elevate dimensioni e le loro ottime proprietà di generalizzazione.

Barachant et al. (2013) ha proposto l'introduzione di un kernel Riemanniano, in modo da fornire uno strumento più efficace nella classificazioni di segnali di elettroencefalografia mediante matrici di covarianza.

Considerando la geometria Riemanniana dello spazio delle matrici di covarianza $P(n)$, ad ogni punto $\mathbf{C}$ appartenente alla varietà $\mathcal{M}$ (in questo caso una matrice di covarianza), può essere definito nello spazio tangente $\mathcal{T}_{\mathbf{C}}\mathcal{M}$ associato al punto, un prodotto scalare. Lo spazio $\mathcal{T}_{\mathbf{C}}\mathcal{M}$ è euclideo e localmente omomorfo alla varietà e le distanze Riemanniane calcolate nella varietà possono essere ben approssimate da distanze euclidee calcolate nello spazio tangente.

Siano $\mathbf{S}_1$ e $\mathbf{S}_2$ due vettori tangenti². Il prodotto scalare nello spazio tangente a $\mathbf{C}$ può essere così definito:

$$\langle \mathbf{S}_1, \mathbf{S}_2 \rangle_{\mathbf{C}} = \text{tr}(\mathbf{S}_1 \mathbf{C}^{-1} \mathbf{S}_2 \mathbf{C}^{-1}) \quad (3.14)$$

Prima di presentare la formulazione del kernel Riemanniano, è necessario definire l'operatore di vettorizzazione vect applicato per usare le matrici di covarianza come input nel classificatore.

Data la simmetria delle matrici di covarianza, viene considerata questo tipo di vettorizzazione, che sovrappone, con una ponderazione appropriata, la parte triangolare superiore di $\mathbf{C} \in P(n)$ in un vettore di dimensioni $(n+1)n/2 \times 1$:

$$\text{vect}(\mathbf{C}) = [C_{1,1}; \sqrt{2}C_{1,2}; C_{2,2}; \sqrt{2}C_{1,3}; \sqrt{2}C_{2,3}; C_{3,3}; \dots; C_{n,n}] \quad (3.15)$$

Senza perdita di generalità, gli elementi che non si trovano nella diagonale principale di $\mathbf{C}$ sono moltiplicati per un coefficiente pari a $\sqrt{2}$, in modo da conservare uguaglianza tra norme $\|\mathbf{C}\|_F = \|\text{vect}(\mathbf{C})\|_2$ (Barachant et al., 2013).

Utilizzando come riferimento la matrice $\mathbf{C}_{ref}$, viene proposto quindi il kernel Riemanniano K_R :

$$\begin{aligned} K_R(\text{vect}(\mathbf{C}_i), \text{vect}(\mathbf{C}_j); \mathbf{C}_{ref}) &= \langle \phi(\mathbf{C}_i), \phi(\mathbf{C}_j) \rangle_{\mathbf{C}_{ref}} \\ &= \text{tr}[\log_{\mathbf{C}_{ref}}(\mathbf{C}_i) \mathbf{C}_{ref}^{-1} \log_{\mathbf{C}_{ref}}(\mathbf{C}_j) \mathbf{C}_{ref}^{-1}] \\ &= \text{tr}[\log(\mathbf{C}_{ref}^{-\frac{1}{2}} \mathbf{C}_i \mathbf{C}_{ref}^{-\frac{1}{2}}) \log(\mathbf{C}_{ref}^{-\frac{1}{2}} \mathbf{C}_j \mathbf{C}_{ref}^{-\frac{1}{2}})] \end{aligned} \quad (3.16)$$

in cui $\phi(\bullet) = \log_{\mathbf{C}_{ref}}(\bullet)$ è una funzione di mappatura esplicita delle matrici di covarianza, cioè la mappatura logaritmica vista in equazione (2.6), $\log(\bullet)$ rappresenta il logaritmo di una matrice e vect è l'operatore di vettorizzazione presentato in (3.15).

²Un vettore tangente è un elemento dello spazio tangente.

Il kernel appena descritto in (3.16) può essere riformulato nel seguente modo:

$$\begin{aligned}
K_R(\text{vect}(\mathbf{C}_i), \text{vect}(\mathbf{C}_j); \mathbf{C}_{ref}) &= \text{tr}[\mathbf{C}_{ref}^{-\frac{1}{2}} \log_{\mathbf{C}_{ref}}(\mathbf{C}_i) \mathbf{C}_{ref}^{-\frac{1}{2}} \mathbf{C}_{ref}^{-\frac{1}{2}} \log_{\mathbf{C}_{ref}}(\mathbf{C}_j) \mathbf{C}_{ref}^{-\frac{1}{2}}] \\
&= \langle \tilde{\mathbf{S}}_i, \tilde{\mathbf{S}}_j \rangle_F \\
&= \text{vect}(\tilde{\mathbf{S}}_i)^T \text{vect}(\tilde{\mathbf{S}}_j)
\end{aligned} \tag{3.17}$$

in cui è stata introdotta la matrice simmetrica

$$\tilde{\mathbf{S}}_i = \mathbf{C}_{ref}^{-\frac{1}{2}} \log_{\mathbf{C}_{ref}}(\mathbf{C}_i) \mathbf{C}_{ref}^{-\frac{1}{2}} = \log(\mathbf{C}_{ref}^{-\frac{1}{2}} \mathbf{C}_i \mathbf{C}_{ref}^{-\frac{1}{2}}) \tag{3.18}$$

Da qui è possibile notare che, trasformando per prima cosa le matrici di covarianza $\{\mathbf{C}_p\}_{p=1}^P$ in matrici $\tilde{\mathbf{S}}_p$ usando l'equazione (3.18) e applicando una classificazione di tipo lineare usando la loro forma vettorizzata, tramite vect in (3.15), è possibile avere anche un'implementazione senza l'uso di kernel..

Per concludere, dato che la trasformazione $\phi(\bullet)$ indicata precedentemente è nota, si hanno due implementazioni equivalenti delle *support vector machines* con kernel Riemanniano (Barachant et al., 2013):

1. vettorizzare tramite vect le matrici di covarianza ed utilizzare una SVM con kernel Riemanniano come definito in (3.16);
2. trasformare le matrici di covarianza con la formula indicata in (3.18), vettorizzarle e usare una SVM con kernel lineare per la classificazione.

Entrambe forniscono gli stessi risultati, la scelta si basa solamente sulla volontà specifica dello sperimentatore.

3.2.2 *Minimum Distance to Riemannian Mean*

La diretta classificazione delle matrici di covarianza è stata provata molto utile in ogni applicazione basata su segnali elettroencefalografici in cui si tiene conto soprattutto dei *pattern* spaziali dei segnali (Barachant et al., 2013). La metrica Riemanniana infatti permette di sfruttare direttamente l'informazione spaziale contenuta nelle matrici di covarianza.

Uno dei più semplici algoritmi supervisionati basati sul confronto delle distanze è il *Minimum Distance to Riemannian Mean*, chiamato brevemente MDRM: esso è molto usato grazie alla sua efficienza e potenza nello sviluppo di sistemi di *Brain-Computer Interface* (Barachant e Congedo, 2014).

Dato un insieme di stima composto da *trial* EEG, $\mathbf{X}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_i)$, e dalle corrispondenti etichette $y_i = 1, \dots, N_c$, il *training* del classificatore consiste nella stima della matrice di covarianza media delle N_c classi:

$$\bar{\mathbf{C}}_J = \mathcal{G}(\mathbf{C}_i | y_i = J) \quad (3.19)$$

dove $J = 1, \dots, N_c$ rappresenta l'etichetta della classe e $\mathcal{G}(\bullet)$ è la media geometrica definita in (2.9).

La classificazione di un nuovo *trial* $\tilde{\mathbf{X}} \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{C}})$ di cui non si conosce la classe di appartenenza y , è raggiunta tramite l'osservazione della distanza minima tra le varie classi:

$$\hat{y} = \arg \min_J \delta_R(\bar{\mathbf{C}}_J, \tilde{\mathbf{C}}) \quad (3.20)$$

L'algoritmo completo è descritto qui in seguito:

Algoritmo 3 Minimum Distance to Riemannian Mean

Input: un insieme di *trial* $\mathbf{X}_i$ di N_c classi differenti e le corrispondenti etichette $y_i = 1, \dots, N_c$.

Input: $\tilde{\mathbf{X}}$ un *trial* EEG di cui non si conosce la classe.

Output: la classe stimata $\hat{y}$ di $\tilde{\mathbf{X}}$.

- 1: Calcolare le matrici di covarianza campionaria (SCM) dei *trial* $\mathbf{X}_i$ per ottenere $\mathbf{C}_i$
 - 2: Calcolare la SCM di $\tilde{\mathbf{X}}$ per ottenere $\tilde{\mathbf{C}}$
 - 3: **for** $J = 1$ to N_c **do**
 - $\bar{\mathbf{C}}_J = \mathcal{G}(\mathbf{C}_i | y_i = J)$, media geometrica Riemanniana (2.9)
 - end for**
 - 4: $\hat{y} = \arg \min_J \delta_R(\bar{\mathbf{C}}_J, \tilde{\mathbf{X}})$, distanza Riemanniana (2.4).
 - 5: **return** $\hat{y}$
-

3.2.3 Rapporto di verosimiglianza

L'ultimo metodo di classificazione che utilizza concetti della geometria Riemanniana considerato nella presente tesi, è basato sul rapporto di verosimiglianza costruito partendo dalla distribuzione di Riemann-Gauss presentata nel Paragrafo 2.2.

In questo metodo, oltre alle informazioni date dal baricentro delle classi, dato dalla media geometrica delle matrici di covarianza campionaria, si aggiunge un altro elemento: la stima della dispersione delle stesse; infatti, come è possibile notare nella formula in (2.10), la funzione di probabilità di Riemann-Gauss dipende da due parametri, $\bar{\mathbf{Y}}$, il baricentro, e σ , il parametro di dispersione in termini di distanza quadratica da $\bar{\mathbf{Y}}$.

Le informazioni campionarie provenienti dall'insieme di stima vengono quindi impiegate per la loro stima, in ciascuna delle classi, per poi calcolare il loro rapporto di verosimiglianza con il fine di classificare la matrice di covarianza campionaria nella classe da cui è più probabile che essa provenga.

Il criterio di decisione impiegato nella classificazione si basa sulla quantificazione del log-rapporto di verosimiglianza

$$LR \propto l(\tilde{\mathbf{C}}|\hat{\mathbf{Y}}_I, \hat{\sigma}_I) - l(\tilde{\mathbf{C}}|\hat{\mathbf{Y}}_J, \hat{\sigma}_J) \quad (3.21)$$

dove $\tilde{\mathbf{C}}$ è l'osservazione da classificare, $(\hat{\mathbf{Y}}_I, \hat{\sigma}_I)$ e $(\hat{\mathbf{Y}}_J, \hat{\sigma}_J)$ sono le stime dei parametri della distribuzione per la classe I e J .

Se $LR > 0$ allora l'osservazione è assegnata alla classe I , mentre se il rapporto risulta inferiore a zero la classe assegnata è quella indicata da J .

La regola decisionale appena presentata include anche la classificazione tramite l'algoritmo *Minimum Distance to Riemannian Mean*, infatti, se il parametro σ risulta identico nelle diverse classi, cioè se la dispersione in esse risulta identica, la scelta della classe per l'osservazione da classificare $\tilde{\mathbf{C}}$ ritorna ad essere un confronto delle distanze misurate tra essa e i baricentri delle diverse classi.

Capitolo 4

Brain Invaders: colpire gli alieni col pensiero

I dati che saranno analizzati nel Capitolo 5 provengono da un esperimento condotto al GIPSA-lab (Università di Grenoble Alpes, CNRS, Grenoble-INP) nel 2013 da E. Vaineau, A. Barachant, M. Congedo e A. Andreev, mirato a testare un videogioco *open-source* basato puramente su tecnologie BCI, in particolare sul potenziale evento-relato P300, *Brain Invaders*: il nome non è nuovo, infatti richiama molto quello del famoso *Space Invaders*, videogioco su cui si basa.

In seguito verrà descritto il suo funzionamento, in modo da avere una visione più chiara dei dati utilizzati, preceduto da un'introduzione ai metodi con cui vengono costruite le BCIs basate sul potenziale evento-relato P300.

4.1 Costruzione di BCI *P300-based*

Le *Brain-Computer Interfaces* (BCIs) basate sul potenziale evento-relato P300 permettono all'utente di interagire con il computer, come per esempio per selezionare simboli, senza l'ausilio di periferiche, sfruttando il paradigma *oddball* (Wolpaw et al., 2002) descritto nel Capitolo 1.

Infatti, facendo lampeggiare in modo esaustivo i simboli, uno alla volta o in gruppi, è possibile stimare la probabilità che ciascuno di essi sia quello se-

lezionato dall'utente; la probabilità è ottenuta valutando il potenziale P300 suscitato da ciascuno dei simboli una volta che ha lampeggiato.

Risulta comunque necessario ripetere più volte il set di lampeggi per ottenere stime dei potenziali evocati affidabili (Congedo et al. 2011).

Nel paradigma originale i simboli lampeggiano per righe e colonne, causando errori di rilevazione dovuti dal fenomeno della "adiacenza-distrazione" (Jin et al., 2011, Townsend et al., 2010): simboli *non-target* in righe o colonne adiacenti a quello *target* attirano l'attenzione dell'utente quando lampeggiano, producendo un'onda P300 che rende più difficile il rilevamento del vero target P300.

Per cercare di ridurre questo problema, una soluzione è far lampeggiare i simboli in gruppi casuali: ponendo (r, c) , $r = 1, \dots, R$ e $c = 1, \dots, C$ la coppia indicante la posizione del simbolo, dove r specifica la riga e c la colonna di appartenenza, è sufficiente creare una tabella di indici supplementare le cui entrate $(x[r], y[c])$, con $x = 1, \dots, R$ e $y = 1, \dots, C$, sono indici ordinali mescolati ad ogni ripetizione. I simboli quindi vengono fatti lampeggiare secondo questi nuovi indici, facendo attenuare così l'effetto "adiacenza-distrazione" ed ottenendo un modello di lampeggiamento totalmente imprevedibile, che dovrebbe contribuire a sostenere l'attenzione del giocatore.

In aggiunta, il lampeggiamento di gruppi casuali consente il posizionamento arbitrario dei simboli sullo schermo (non è più necessario disporre i simboli su una griglia), il che espande notevolmente l'usabilità del paradigma P300 (Congedo et al., 2011).

Inoltre, solitamente l'intervallo di stimolo (la durata del lampeggio) e l'intervallo tra gli stimoli (il tempo tra due lampeggi) sono mantenuti costanti, ma il lampeggiamento ad intervalli regolari è fastidioso e stancante perché la corteccia visiva è spinta ad oscillare alla frequenza di lampeggiamento, che di solito è molto lontana dalla naturale oscillazione talamo-corticale di questa regione, che è nell'intervallo α (8-12 Hz) ed inoltre, esso rende prevedibile e noioso lo schema di lampeggiamento (Congedo et al., 2011).

Per eliminare queste problematiche, è possibile utilizzare un intervallo tra gli

stimoli casuale, generandolo da una distribuzione esponenziale di parametro λ : essa è adatta alla situazione perché è solitamente usata per modellare l'attesa all'accadimento di un evento (Congedo et al., 2011).

Infine, è stato riscontrato che l'aumento delle dimensioni dei simboli lampeggiati migliora l'ampiezza del potenziale evocato P300, influenzando in maniera positiva la bontà della classificazione (Gibert et al., 2008).

4.2 Brain Invaders

Il videogioco presenta due modalità di gioco: una modalità operativa caratterizzata da una fase di stima ed una di verifica del classificatore (modalità classica) ed una modalità operativa senza calibrazione, entrambe caratterizzate da una fase di *training* e da una fase *online*, sempre passate in quest'ordine (Vaineau et al., 2019). L'unica differenza tra le due modalità sta nel fatto che in quella senza calibrazione non è utilizzato l'insieme di stima e i segnali vengono classificati direttamente tramite un'inizializzazione del classificatore basata su dati già disponibili. In qualsiasi caso, i soggetti sottoposti al gioco sono completamente ignari rispetto alla modalità di funzionamento, dato che le due modalità appaiono identiche.

Brain Invaders funziona a turni: in ogni *round* viene visualizzata una griglia di alieni di dimensioni 6×6 su sfondo nero. Gli alieni sono tutti visualizzati in grigio, ad eccezione del bersaglio (*target*) che viene visualizzato in rosso.

Il lampeggiamento di gruppo si ottiene visualizzando i simboli *non-target* con un grigio più chiaro e il simbolo *target* con il colore complementare al rosso, cioè il ciano: viene scelto il colore complementare in quanto crea il massimo contrasto (Congedo et al., 2011). Inoltre, per ottenere migliori risultati, quando lampeggia il simbolo *target* esso è ingrandito del 30%.

In Figura 4.1 sono presenti due schermate d'esempio: nel pannello di sinistra è possibile notare un gruppo casuale che lampeggia, in cui è compreso l'alieno *target* ingrandito e colorato di ciano, mentre a destra è visibile la distruzione di un alieno *non-target* durante il terzo *round*.

Nella fase di stima il numero di lampeggi è stato fissato (80 lampeggi *target* e 400 lampeggi *non-target*), mentre nella fase di verifica (fase *online*) il numero di lampeggi *target* e *non-target* è ancora in rapporto 1/5, tuttavia il loro numero è variabile perché *Brain Invaders* lavora con un numero fisso di livelli di gioco, ed il numero di ripetizioni necessarie per distruggere il *target* (e quindi passare al livello successivo) dipende dalle prestazioni dell'utente.

Come nel videogioco *Space Invaders*, gli alieni si spostano continuamente da sinistra a destra dello schermo e dall'alto verso il basso di una riga quando entrano in contatto con uno dei bordi dello schermo.

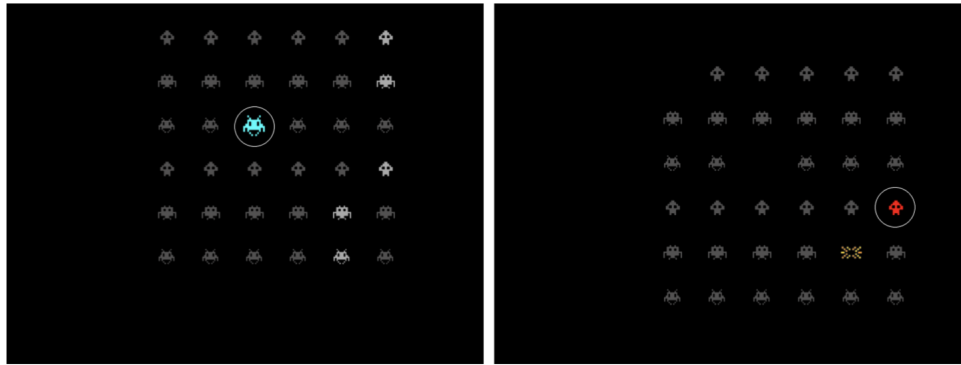


Figura 4.1: *Brain Invaders*: esempi di schermate di gioco. A sinistra è in risalto l'alieno *target* illuminato ed ingrandito del 30% nella prima ripetizione di lampeggiamenti, a destra è mostrata la distruzione di un alieno *non-target* nella terza ripetizione. Fonte: Congedo et al., 2011.

Ad ogni ripetizione il sistema, tramite un algoritmo di classificazione, discerne i simboli *non-target* da quello *target* e distrugge l'alieno bersaglio (Figura 4.1, a destra): se l'alieno distrutto è quello corretto, il *round* termina, altrimenti esso viene eliminato comunque ed inizia un'altra ripetizione di lampeggi. Il processo continua fino alla distruzione dell'alieno *target* o fino alla distruzione di 14 alieni *non-target*, dopodiché inizia un altro *round* (Vaineau et al., 2019, Congedo et al., 2011).

Alla fine di ogni *round* è prevista una pausa di 30 secondi, che permette al giocatore un breve rilassamento; dopo di essa il nuovo *round* ha inizio.

Un insieme di sei *round* è denominato blocco: una sessione di gioco è formata da più blocchi.

Il videogioco possiede anche un sistema di punteggi: tra due *round* consecutivi vengono visualizzati sullo schermo i punti ottenuti nell'ultimo *round* ed il punteggio cumulativo ottenuto fino a quel momento.

Il calcolo del punteggio si basa sulle ripetizioni: i punti ottenuti ad ogni *round* infatti sono inversamente proporzionali al numero di ripetizioni necessarie per distruggere il bersaglio.

Per quanto riguarda altre caratteristiche tecniche, il tempo di lampeggiamento è fisso, pari a 60 ms, mentre il tempo d'attesa tra due lampeggi, come già menzionato precedentemente, è estratto casualmente da una distribuzione esponenziale di media 1 ($\lambda = 1$), moltiplicato per 100 per ottenere un intervallo *inter-stimulus* medio di 100 ms e limitato nell'intervallo [20,500] ms (Congedo et al., 2011).

In conclusione, il funzionamento di *Brain Invaders* si fonda su tre principali fasi controllate da tre moduli di software: l'acquisizione dei dati, l'elaborazione degli stessi ed il *rendering* (Congedo et al., 2011).

Acquisizione dei dati. Essa è controllata dal server di acquisizione Open-ViBE (Renard et al., 2010), che si occupa di estrarre i dati dall'elettroencefalografo, di correggerli per eventuali errori di amplificazione e di inviarli alla piattaforma Open-ViBE per l'analisi.

Elaborazione dei dati. Essa viene attuata *online* dalla piattaforma Open-ViBE. Ad ogni ripetizione il sistema applica l'algoritmo di classificazione *Minimum Distance to Riemannian Mean* (Vaineau et al., 2019) per discriminare l'alieno bersaglio da quelli che il giocatore non vuole colpire e, una volta identificato l'alieno *target*, la sua posizione viene inviata all'applicazione di *rendering*.

Rendering. L'interfaccia visiva del gioco è implementata in linguaggio C++ usando il motore Ogre3D (www.ogre3d.org).

Capitolo 5

Analisi dei dati

Nel presente capitolo saranno presentati i risultati ottenuti mediante le tecniche di classificazione descritte nel Capitolo 3 applicate all'insieme dei dati proveniente da sessioni di gioco di *Brain Invaders*, descritto in dettaglio nel capitolo precedente.

Prima di approfondirli verrà descritto il paradigma sperimentale adottato, le tecniche di *preprocessing* utilizzate e verrà condotta una breve analisi esplorativa.

5.1 Presentazione del dataset

5.1.1 Paradigma sperimentale

All'esperimento accennato all'inizio del Capitolo 4 hanno partecipato 24 soggetti in totale, di cui 12 donne, con un'età media (sd) pari a 25.96 (4.46): 7 soggetti, tra cui quattro donne, hanno preso parte ad otto sessioni di gioco, svolte in giorni diversi, mentre i restanti soggetti (soggetti 8-24) hanno giocato soltanto ad una sessione. Ciascuna sessione era caratterizzata da due parti: una caratterizzata dalla modalità operativa non adattiva (modalità classica) ed una da quella senza calibrazione, come descritte nel Capitolo 4; l'ordine delle modalità è stato randomizzato per ogni sessione, in modo da non avere una distorsione "psicologica" da parte del soggetto. Nel presente Capitolo si è fatto uso solamente dell'insieme di dati derivante dalla moda-

lità classica, in cui insieme di stima e di verifica sono già suddivisi: questa divisione è stata quindi usata per dividere l'insieme di stima dei classificatori qui utilizzati da quello di verifica.

5.1.2 Acquisizione dei dati

I dati sono stati acquisiti con un amplificatore g.USBamp (g.tec Schiedlberg, Austria), con queste caratteristiche:

- elettrodi: 16 elettrodi di cloruro d'argento / argento posizionati su FP1, FP2, F5, AFz, F6, T7, Cz, T8, P7, P3, Pz, P4, P8, O1, Oz, O2 secondo il sistema internazionale 10/20;
- riferimento: lobo dell'orecchio sinistro;
- ground: nella posizione dell'elettrodo Fz.

Nessun filtro digitale è stato applicato ai dati e la frequenza di campionamento è di 512 campioni al secondo. In Figura 5.1 è possibile avere una visione d'insieme del posizionamento degli elettrodi sullo scalpo.

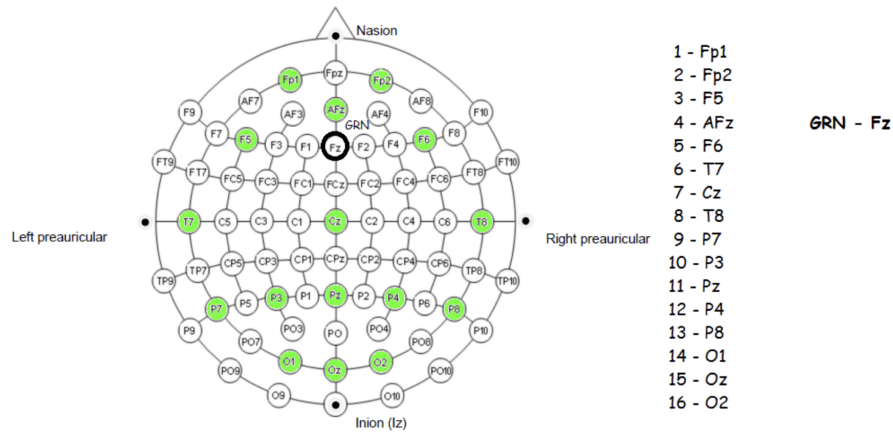


Figura 5.1: Posizionamento degli elettrodi sullo scalpo per l'acquisizione del segnale EEG durante le sessioni di gioco di *Brain Invaders*. In verde gli elettrodi posizionali, cerchiato di nero l'elettrodo di riferimento. Fonte: Vaineau et al., 2019.

5.1.3 Importazione e preprocessing dei dati

Il formato originale dei dati (`.gdf`) non consente l'importazione diretta nell'ambiente statistico R utilizzato poi per tutte le analisi, di conseguenza è necessaria l'importazione mediante il linguaggio `Python`, in modo da convertire i file in formato `.mat`, facilmente leggibile in R tramite il pacchetto `R.matlab`. Lo script `python` è disponibile in <https://github.com/plcrodrigues/py.BI.EEG.2013-GIPSA/tree/master/braininvaders2013>, mentre lo script usato per l'importazione in R si trova in Appendice A.

Per quanto riguarda il *preprocessing* dei dati, in modo da considerare solo le frequenze di interesse, è stato applicato un filtro passa-banda ai segnali EEG (8-35 Hz) per tener conto delle onde μ (8-14 Hz) e β (14-30 Hz).

Solitamente, dopo aver applicato il filtro appena accennato, è comune procedere con un filtro spaziale per ridurre la dimensionalità dei dati ed essere in grado di discriminare ulteriormente i segnali che riflettono l'evento di interesse da quelli di nessun interesse: Barachant et al. (2013) ha proposto l'applicazione dei metodi basati sulla geometria Riemanniana visti nel Capitolo 3 per consentire di bypassare l'applicazione dei filtri spaziali, basando la classificazione dei segnali solo sulle matrici di covarianza dei segnali rilevati. Per questo motivo i segnali EEG sono stati ridotti in frammenti chiamati *trials* (480 per ogni sessione) e con essi sono state calcolate, mediante la formula (2.1), le matrici di covarianza $\mathbf{P}_i \in \mathbb{R}^{16 \times 16}, i = 1, \dots, 24$ necessarie per la classificazione.

5.1.4 Divisione in due insiemi di dati distinti

Data la diversa numerosità di sessioni a cui sono stati sottoposti i soggetti, l'insieme di dati originale è stato diviso in due dataset distinti:

- il primo costituito dai soggetti da 1 a 7;
- il secondo con i rimanenti soggetti, da 8 a 24.

In questo modo all'interno di ogni dataset i soggetti hanno lo stesso numero di sessioni: nel primo 8 sessioni per soggetto, nel secondo solamente una.

Quindi, in seguito le analisi verranno svolte separatamente, con l'indicazione del dataset di riferimento.

5.2 Creazione di *super trial*

Prima di condurre le analisi esplorative ed illustrare i risultati ottenuti è necessaria una breve introduzione al *super trial*.

Dato che ***Brain Invaders*** è un *P300-based* video game, per ottenere migliori risultati nella classificazione può essere necessario introdurre nel calcolo delle matrici di covarianza campionaria anche informazioni di tipo temporale oltre che di tipo spaziale.

Ciò può essere raggiunto mediante la costruzione di *super trial* $\tilde{\mathbf{X}}_i \in \mathbb{R}^{2C \times N}$

$$\tilde{\mathbf{X}}_i = \begin{bmatrix} \mathbf{P1} \\ \mathbf{X}_i \end{bmatrix}$$

ottenuti concatenando la risposta ERP attesa $\mathbf{P1} = \frac{1}{|I|} \sum_{i \in I} \mathbf{X}_i \in \mathbb{R}^{C \times N}$ dove C indica il numero di canali utilizzati, N il numero di punti temporali e I è l'insieme degli indici dei *trial* che appartengono alla classe d'interesse, calcolata per esempio mediando le risposte della classe d'interesse di vari *trial*, ad ogni *trial* disponibile $\mathbf{X}_i$.

La definizione del *super trial* permette una migliore applicazione di classificatori basati sulla geometria Riemanniana in problemi in cui è necessario distinguere la presenza di potenziali evento-relati, come il P300, dato che si prende come riferimento l'informazione propria di ciascuna classe (in questo caso la media ERP), sovrapponendola ad ogni *trial* registrato.

Quindi le matrici di covarianza campionaria, che poi verranno utilizzate per la classificazione, sono costruite a partire da questi *super trial* e possono essere scomposte in tre blocchi differenti: Σ_{P1} che rappresenta la matrice di covarianza di $\mathbf{P1}$, Σ_i che è la matrice di covarianza di $\mathbf{X}_i$ e $\mathbf{C}_{P1, X_i}$ che è la

matrice di cross-covarianza tra $\mathbf{P1}$ e $\mathbf{X}_i$:

$$\tilde{\Sigma}_i = \begin{bmatrix} \Sigma_{P1} & \mathbf{C}_{P1,X_i}^T \\ \mathbf{C}_{P1,X_i} & \Sigma_i \end{bmatrix}$$

La giustificazione dell'uso del *super trial* deriva dal fatto che l'informazione contenuta in Σ_i non è sufficiente per raggiungere un'alta accuratezza di classificazione: la maggior parte delle informazioni utili è contenuta in $\mathbf{C}_{P1,X_i}$, infatti se il *trial* contiene un ERP in fase a Σ_{P1} , la cross-covarianza sarà alta per i canali corrispondenti, negli altri casi sarà molto vicina a zero (Barachant e Congedo, 2014).

In Figura 5.2 è possibile notare infatti i blocchi in cui la matrice di covarianza costruita con il *super trial* può essere suddivisa e come quelli di interesse siano differenti nei *trial* in cui si trova un evento *target* rispetto a quelle che sono generate da eventi *non-target*.

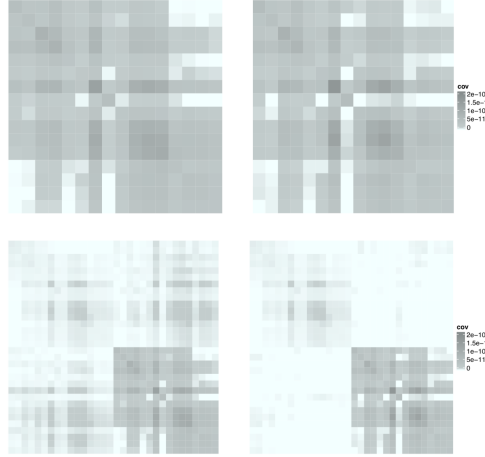


Figura 5.2: Esempio di matrici di covarianza, rappresentate con heatmap, costruite senza *super trial* (in alto) e con *super trial* (in basso). Nella parte sinistra matrici riferite ai *trials* degli eventi *target*, a destra degli eventi *non-target*.

In modo da verificare l'eventuale miglioramento della classificazione in ciascuno dei due dataset, i modelli presentati nel Capitolo 3 verranno applicati alle matrici di covarianza costruite con e senza i *super trial*.

5.3 Analisi esplorative

L'obiettivo di questa sezione è esplorare la distribuzione spaziale dei dati, cioè le matrici di covarianza, di alcuni soggetti che hanno partecipato all'esperimento descritto precedentemente, in modo da avere una rappresentazione grafica delle loro caratteristiche.

Di seguito è stata sfruttata la tecnica chiamata *t*-SNE che permette la visualizzazione di dati ad elevata dimensionalità tramite la loro proiezione in uno spazio di minori dimensioni: in questo caso è particolarmente utile dato che le matrici di covarianza considerate sono di dimensioni 16×16 o, in caso di costruzione da *super trial*, 32×32 .

5.3.1 *t*-SNE

Il *T-distributed Stochastic Neighbor Embedding*, brevemente *t*-SNE, è una tecnica di riduzione della dimensionalità, proposta da Maaten e Hinton (2008), che rende possibile la visualizzazione, in due o tre dimensioni, di dati che giacciono in uno spazio ad alta dimensionalità.

L'algoritmo fa in modo che oggetti vicini nello spazio originale risultino vicini nello spazio a dimensionalità ridotta, ed oggetti lontani risultino lontani in entrambi gli spazi, cercando di preservare la struttura locale.

In particolare l'algoritmo *t*-SNE comprende due passi principali. Per prima cosa, l'algoritmo costruisce una distribuzione di probabilità su coppie di oggetti ad alta dimensione in modo tale che ad ogni coppia di punti nello spazio originale ad alta dimensionalità venga associato un valore di probabilità elevato se i due punti sono simili, basso se sono dissimili. In secondo luogo, il *t*-SNE definisce una seconda distribuzione di probabilità simile, definita nello spazio a dimensionalità ridotta e minimizza la divergenza di Kullback-Leibler

$$D_{KL}(p|q) = \int_{-\infty}^{+\infty} p(x) \log \frac{p(x)}{q(x)} dx \quad (5.1)$$

tra le due distribuzioni rispetto alle posizioni dei punti nella mappa, riorga-

nizzando così i punti nello spazio a dimensione ridotta.

L'uso della divergenza di Kullback-Leibler come funzione obiettivo consente di avere penalità elevate se punti vicini nello spazio originale vengono considerati lontani nello spazio a dimensionalità ridotta, tendendo quindi a preservare la struttura locale della distribuzione dei punti. Il risultato è una mappa a bassa dimensionalità che riflette la similarità tra i punti nello spazio originario.

Infine, da notare che la distribuzione q che viene utilizzata per approssimare la distribuzione p che giace sullo spazio ad alta dimensionalità è una t di Student con un grado di libertà, in modo da modellare meglio la dissimilarità tra oggetti distanti grazie alle code pesanti che la caratterizzano.

Di seguito è stato applicato l'algoritmo t -SNE su matrici di distanze calcolate per due soggetti, a titolo esemplificativo, uno appartenente al primo dataset (Soggetto 1) ed uno al secondo (Soggetto 8). Per ognuno è stata creata una visualizzazione della similarità delle matrici sia bidimensionale che tridimensionale, distinguendo le matrici costruite con e senza i *super trial* descritti in precedenza.

La distanza tra due matrici di covarianza è stata ottenuta applicando la formula (2.4) ad ogni coppia, ottenendo per ognuno dei due soggetti considerati una **matrice di distanze** di dimensione $N \times N$, con N indicante la numerosità del campione preso in esame, così strutturata:

$$\Delta_N = \begin{bmatrix} 0 & \delta_R(\mathbf{P}_1, \mathbf{P}_2) & \cdots & \delta_R(\mathbf{P}_1, \mathbf{P}_N) \\ \delta_R(\mathbf{P}_2, \mathbf{P}_1) & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \delta_R(\mathbf{P}_N, \mathbf{P}_1) & \cdots & \cdots & 0 \end{bmatrix}$$

in cui nella diagonale abbiamo zero, dato che la distanza tra una matrice e se stessa è nulla, $\delta_R(\mathbf{P}_i, \mathbf{P}_i) = 0$, e $\delta_R(\mathbf{P}_i, \mathbf{P}_j) \geq 0$, $i \neq j$, $i, j = 1, \dots, N$ rappresenta la distanza nella metrica di Rao-Fisher.

La matrice è simmetrica, infatti $\delta_R(\mathbf{P}_i, \mathbf{P}_j) = \delta_R(\mathbf{P}_j, \mathbf{P}_i)$, $\forall i, j$.

Le rappresentazioni riportate di seguito sono basate su un'unica sessione per ogni soggetto, ottenendo così matrici di distanze di dimensioni pari a 480×480 .

Dataset 1: soggetto 1

Come è possibile notare nei pannelli di sinistra delle Figure 5.3 e 5.4, le matrici di covarianza calcolate senza l'ausilio del *super trial* relative alle due classi, *target* e *non-target*, sembrano non essere molto dissimili tra loro, infatti la distribuzione spaziale nel grafico a dispersione non sembra evidenziare gruppi ben definiti.

Per quanto riguarda invece le matrici di covarianza ottenute con i *super trial* (pannello di destra delle Figure 5.3 e 5.4) è possibile notare come le matrici associate agli eventi *target* abbiano una distribuzione spaziale più limitata rispetto alle altre ed alle stesse calcolate senza *super trial*, andando quasi a creare un gruppo.

Dataset 2: soggetto 8

Anche per il soggetto 8 è possibile dire quanto detto per il soggetto 1: nel caso di matrici di covarianza calcolate senza l'ausilio di *super trial* non sembra esserci tendenza alla creazione di gruppi, come si può vedere nel pannello di sinistra delle Figure 5.5 e 5.6.

A differenza del soggetto 1 però, le matrici calcolate con *super trial* sono più concentrate spazialmente, ma non sembrano formare cluster di particolare rilevanza.

In conclusione non sembra esserci una netta separazione delle due classi da un punto di vista visivo per il soggetto 8, mentre per il soggetto uno si nota maggiormente la similarità delle matrici di covarianza riguardanti gli eventi *target* create con *super trial*.

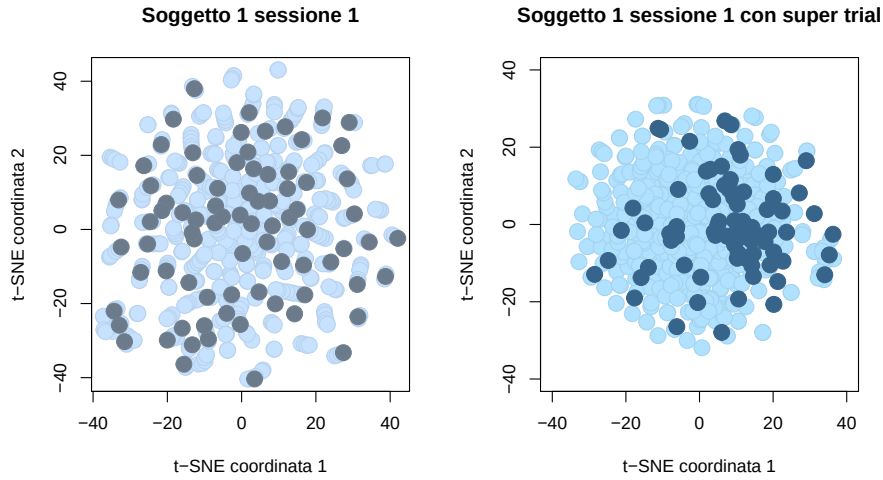


Figura 5.3: Rappresentazione bidimensionale della similarità tra le matrici di covarianza calcolate per la prima sessione del soggetto 1, ottenuta tramite l'algoritmo t-SNE. In azzurro chiaro le matrici dei *trial non-target*, in blu quelle riferite ai *trial target*. A sinistra matrici calcolate senza *super trial*, a destra con esso.

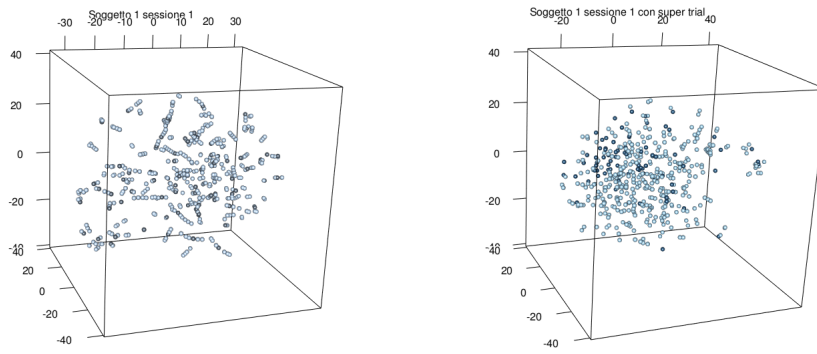


Figura 5.4: Rappresentazione tridimensionale della similarità tra le matrici di covarianza calcolate per la prima sessione del soggetto 1, ottenuta tramite l'algoritmo t-SNE. In azzurro chiaro le matrici dei *trial non-target*, in blu quelle riferite ai *trial target*. A sinistra matrici calcolate senza *super trial*, a destra con esso.

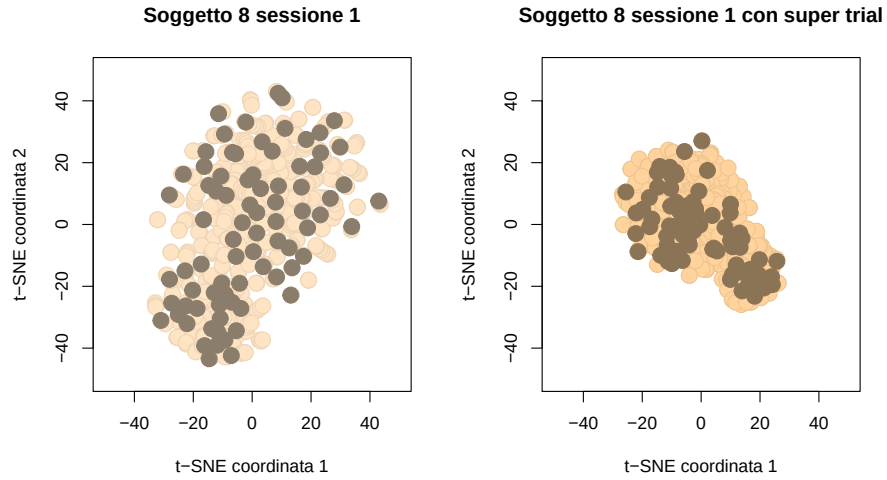


Figura 5.5: Rappresentazione bidimensionale della similarità tra le matrici di covarianza calcolate per la prima sessione del soggetto 8, ottenuta tramite l'algoritmo t-SNE. In giallo le matrici dei *trial non-target*, in marrone quelle riferite ai *trial target*. A sinistra matrici calcolate senza *super trial*, a destra con esso.

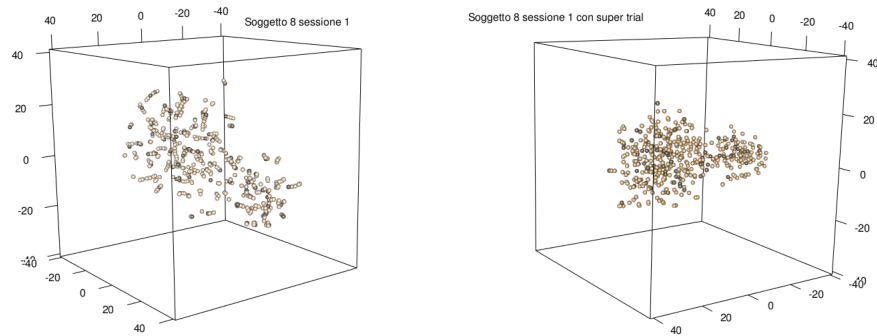


Figura 5.6: Rappresentazione tridimensionale della similarità tra le matrici di covarianza calcolate per la prima sessione del soggetto 8, ottenuta tramite l'algoritmo t-SNE. In giallo le matrici dei *trial non-target*, in marrone quelle riferite ai *trial target*. A sinistra matrici calcolate senza *super trial*, a destra con esso.

5.4 Risultati

In questa sezione saranno presentati i risultati delle analisi condotte, suddivisi per metodologie usate: per primi quelli derivanti dai metodi di *statistical learning*, seguiti da quelli basati sulla geometria Riemanniana, entrambi descritti nel Capitolo 3.

Ogni modello è stato utilizzato sia con matrici di covarianza create con l'ausilio dei *super trial*, che matrici di covarianza "semplici". Inoltre, per entrambi i dataset sono state svolte le analisi sia usando come insieme di stima le sessioni di *training* di tutti i soggetti, che quelle disponibili per ogni singolo soggetto, ottenendo così una generalizzazione del processo di stima nel primo caso, ed una stima personalizzata nel secondo.

In questo modo è possibile verificare la bontà di classificazione dei vari metodi al variare del numero delle sessioni presenti. Nel primo dataset infatti si hanno a disposizione otto sessioni per soggetto, nel secondo solamente una: ciò consente quindi di valutare l'accuratezza dei metodi utilizzati con solo una sessione, con otto sessioni e con il totale delle sessioni derivanti da tutti i soggetti appartenenti all'insieme di dati considerato.

La bontà delle metodologie utilizzate è espressa tramite i seguenti indici riassuntivi:

- **Accuratezza bilanciata** (in seguito indicata solo con Accuratezza): media delle proporzioni di unità correttamente classificate per ognuna delle due classi¹. Non è stata utilizzata la semplice somma delle unità correttamente classificate sul totale generale a causa dei dataset non bilanciati;
- **Sensibilità**: la proporzione delle osservazioni appartenenti alla classe *target* classificate in modo corretto;
- **Specificità**: la proporzione delle osservazioni appartenenti alla classe *non-target* classificate in modo corretto.

¹Il peso applicato alle unità correttamente classificate per ogni classe è il reciproco del corrispondente totale di classe osservato.

5.4.1 Metodi di *statistical learning*

Dato che le unità statistiche da classificare sono matrici di covarianza, per utilizzare i metodi di *statistical learning* è necessario vettorizzare le matrici: per fare ciò è stato utilizzato l'operatore `vect` presentato in (3.15).

Le funzioni create per la costruzione delle matrici di covarianza e per la loro vettorizzazione si trovano in Appendice A.

Infine, è utile indicare che per ogni modello considerato è stata fatta una *grid search* in modo da individuare la combinazione migliore dei parametri da utilizzare.

In seguito si trovano gli indici riassuntivi che rappresentano la bontà delle previsioni effettuate sul primo dataset (Tabelle 5.1 e 5.2) e sul secondo (Tabelle 5.3, 5.4, 5.5 e 5.6). In grassetto sono riportati i valori di accuratezza migliori in assoluto per ogni soggetto, mentre sono sottolineate le performance migliori nella divisione tra creazione delle matrici di covarianza mediante l'uso di *super trial* e non. Questa distinzione è resa evidente in modo da capire se l'utilizzo di questa metodologia per la costruzione delle matrici di covarianza è necessaria per un netto miglioramento delle previsioni.

Per quanto riguarda la classificazione nel primo dataset, dalla Tabella 5.1 si può evincere che il metodo migliore tra quelli considerati, nella prova con l'insieme di training generalizzato, è XGBoost sia quando applicato alle matrici di covarianza costruite con il *super trial*, che quando usato con quelle semplici: esso infatti ha raggiunto un'accuratezza bilanciata di circa 85%. Mentre, quando è stato utilizzato il *training set* specifico per ogni soggetto (Tabella 5.2), risultano migliori la Model-Based Oblique Random Forest (oRF) ed il bagging per l'applicazione dei metodi con l'utilizzo delle matrici di covarianza costruite senza *super trial*, per tornare a scegliere XGBoost con le matrici costruite con il *super trial*.

Si può notare inoltre che i metodi valutati in questa sezione raggiungono risultati migliori se applicati alle matrici di covarianza costruite con *super trial*:

per esempio, l'accuratezza bilanciata, ma anche la sensibilità e la specificità di XGBoost aumenta circa del 20% rispetto a quando è usato su matrici di covarianza costruite senza l'ausilio del *super trial*.

Per concludere, si può osservare che i metodi di *statistical learning* utilizzati con le matrici di covarianza costruite con i *super trial* forniscono sempre i migliori risultati, sia utilizzando l'insieme di training generalizzato che quello specifico per soggetto.

		Senza <i>super trial</i>				Con <i>super trial</i>			
Soggetto	Indice	SVM	oRF	Bagging	XGBoost	SVM	oRF	Bagging	XGBoost
1	Accuratezza	0.50	0.50	0.51	<u>0.52</u>	0.84	0.75	0.80	0.86
	Sensibilità	0.00	0.60	0.03	0.53	0.82	0.89	0.75	0.89
	Specificità	1.00	0.40	0.99	0.51	0.85	0.60	0.86	0.84
2	Accuratezza	0.49	0.53	0.51	<u>0.57</u>	0.86	0.73	0.84	0.88
	Sensibilità	0.01	0.60	0.07	0.54	0.91	0.95	0.86	0.92
	Specificità	0.98	0.46	0.95	0.60	0.82	0.51	0.82	0.83
3	Accuratezza	0.50	<u>0.55</u>	0.51	<u>0.55</u>	0.80	0.71	0.77	0.82
	Sensibilità	0.00	0.68	0.06	0.62	0.80	0.81	0.72	0.80
	Specificità	1.00	0.41	0.95	0.47	0.79	0.61	0.82	0.84
4	Accuratezza	0.50	<u>0.52</u>	0.50	0.50	0.86	0.74	0.84	0.87
	Sensibilità	0.00	0.53	0.02	0.47	0.90	0.82	0.84	0.93
	Specificità	1.00	0.51	0.98	0.54	0.82	0.65	0.84	0.82
5	Accuratezza	0.50	0.51	0.50	<u>0.53</u>	0.80	0.75	0.78	0.82
	Sensibilità	0.00	0.46	0.01	0.49	0.71	0.87	0.67	0.76
	Specificità	1.00	0.56	0.99	0.58	0.88	0.64	0.89	0.88
6	Accuratezza	0.50	<u>0.52</u>	0.50	0.51	0.81	0.72	0.69	0.85
	Sensibilità	0.00	0.59	0.01	0.41	0.75	0.86	0.39	0.86
	Specificità	1.00	0.45	0.99	0.62	0.87	0.57	0.98	0.84
7	Accuratezza	0.50	0.53	0.50	<u>0.54</u>	0.79	0.71	0.79	0.84
	Sensibilità	0.00	0.63	0.02	0.56	0.74	0.84	0.75	0.80
	Specificità	1.00	0.42	0.99	0.52	0.85	0.58	0.83	0.88
Indici medi	Accuratezza	0.50	0.52	0.50	<u>0.53</u>	0.82	0.73	0.79	0.85
	Sensibilità	0.00	0.58	0.03	0.52	0.80	0.86	0.71	0.85
	Specificità	1.00	0.46	0.98	0.55	0.84	0.60	0.86	0.85

Tabella 5.1: Dataset 1. Risultati ottenuti tramite classificazione con i metodi di *statistical learning*. Insieme di stima generalizzato.

Soggetto	Indice	Senza <i>super trial</i>				Con <i>super trial</i>			
		SVM	oRF	Bagging	XGBoost	SVM	oRF	Bagging	XGBoost
1	Accuratezza	<u>0.54</u>	<u>0.54</u>	0.53	0.53	0.77	0.76	0.80	0.84
	Sensibilità	0.35	0.61	0.38	0.43	0.64	0.87	0.76	0.80
	Specificità	0.73	0.47	0.69	0.64	0.89	0.65	0.84	0.87
2	Accuratezza	0.51	<u>0.57</u>	<u>0.57</u>	<u>0.57</u>	0.50	0.76	0.84	0.86
	Sensibilità	0.15	0.56	0.41	0.31	1.00	0.94	0.82	0.86
	Specificità	0.87	0.58	0.74	0.82	0.00	0.58	0.85	0.86
3	Accuratezza	0.47	<u>0.54</u>	<u>0.54</u>	0.51	0.50	0.71	0.81	0.86
	Sensibilità	0.07	0.53	0.49	0.40	1.00	0.83	0.88	0.87
	Specificità	0.87	0.54	0.59	0.62	0.00	0.60	0.75	0.84
4	Accuratezza	0.48	<u>0.53</u>	0.51	0.52	0.50	0.71	0.81	0.86
	Sensibilità	0.33	0.62	0.33	0.39	0.00	0.84	0.76	0.85
	Specificità	0.63	0.44	0.68	0.64	1.00	0.57	0.86	0.86
5	Accuratezza	0.50	<u>0.52</u>	<u>0.52</u>	<u>0.52</u>	0.82	0.73	0.78	0.82
	Sensibilità	0.01	0.57	0.38	0.38	0.89	0.83	0.64	0.74
	Specificità	0.99	0.46	0.67	0.66	0.76	0.62	0.91	0.89
6	Accuratezza	0.45	0.53	0.53	<u>0.54</u>	0.59	0.69	0.77	0.83
	Sensibilità	0.29	0.67	0.43	0.44	0.94	0.87	0.80	0.87
	Specificità	0.62	0.38	0.63	0.64	0.24	0.52	0.74	0.79
7	Accuratezza	0.44	<u>0.55</u>	<u>0.55</u>	0.53	0.50	0.70	0.81	0.83
	Sensibilità	0.51	0.67	0.50	0.47	0.00	0.88	0.86	0.89
	Specificità	0.37	0.42	0.60	0.59	1.00	0.52	0.76	0.76
Indici medi	Accuratezza	0.48	<u>0.54</u>	<u>0.54</u>	0.53	0.60	0.72	0.80	0.84
	Sensibilità	0.24	0.60	0.42	0.40	0.64	0.87	0.79	0.84
	Specificità	0.72	0.47	0.66	0.66	0.56	0.58	0.81	0.84

Tabella 5.2: Dataset 1. Risultati ottenuti tramite classificazione con i metodi di *statistical learning*. Insieme di stima specifico per ogni soggetto.

Osservando ora le Tabelle 5.4 e 5.6, è possibile notare che anche nel secondo dataset il miglior metodo in termini di accuratezza bilanciata è XGBoost applicato a matrici di covarianza costruite con *super trial*, sia utilizzando il *training set* generalizzato che specifico per ogni soggetto, con un'accuratezza media di circa 85%. Invece, se vengono utilizzate le matrici di covarianza campionarie semplici, il classificatore migliore tra quelli valutati in questa sezione si rivela l'oRF, affiancata dal bagging in casi di uso dell'insieme di training specifico per ogni soggetto, con un'accuratezza del 53%.

Soggetto	Indice	Senza <i>super trial</i>				Con <i>super trial</i>			
		SVM	oRF	Bagging	XGBoost	SVM	oRF	Bagging	XGBoost
8	Accuratezza	0.46	<u>0.59</u>	0.58	0.55	0.50	0.81	0.85	<u>0.94</u>
	Sensibilità	0.87	0.68	0.37	0.55	0.00	0.97	0.92	0.97
	Specificità	0.06	0.49	0.78	0.54	1.00	0.64	0.78	0.90
9	Accuratezza	0.43	<u>0.62</u>	0.60	0.59	0.81	0.80	0.81	<u>0.84</u>
	Sensibilità	0.78	0.66	0.46	0.56	0.76	0.92	0.86	0.84
	Specificità	0.09	0.57	0.75	0.62	0.86	0.68	0.76	0.85
10	Accuratezza	0.47	0.55	0.52	<u>0.57</u>	0.81	0.76	0.78	<u>0.85</u>
	Sensibilità	0.83	0.62	0.33	0.53	0.75	0.89	0.75	0.84
	Specificità	0.11	0.47	0.70	0.61	0.88	0.63	0.81	0.86
11	Accuratezza	0.45	<u>0.63</u>	0.58	0.54	0.70	0.72	0.77	<u>0.87</u>
	Sensibilità	0.80	0.73	0.45	0.45	0.59	0.84	0.75	0.88
	Specificità	0.10	0.53	0.72	0.62	0.82	0.60	0.78	0.87
12	Accuratezza	0.50	<u>0.54</u>	0.51	0.53	0.78	0.68	0.73	<u>0.82</u>
	Sensibilità	0.86	0.58	0.30	0.49	0.67	0.68	0.59	0.76
	Specificità	0.14	0.50	0.73	0.57	0.88	0.68	0.88	0.88
13	Accuratezza	0.47	0.55	0.48	<u>0.57</u>	<u>0.88</u>	0.80	0.83	<u>0.88</u>
	Sensibilità	0.85	0.52	0.17	0.48	0.90	0.85	0.81	0.90
	Specificità	0.07	0.58	0.80	0.66	0.87	0.74	0.86	0.85
14	Accuratezza	0.49	0.53	0.53	<u>0.54</u>	0.74	0.63	0.79	<u>0.83</u>
	Sensibilità	0.85	0.60	0.35	0.53	0.53	0.72	0.80	0.85
	Specificità	0.13	0.46	0.71	0.55	0.95	0.54	0.77	0.82
15	Accuratezza	0.49	<u>0.56</u>	0.51	0.49	0.75	0.71	0.75	<u>0.82</u>
	Sensibilità	0.86	0.55	0.29	0.45	0.61	0.82	0.66	0.75
	Specificità	0.12	0.57	0.74	0.54	0.90	0.60	0.85	0.89
16	Accuratezza	0.49	<u>0.57</u>	0.56	0.53	0.50	0.76	0.84	<u>0.92</u>
	Sensibilità	0.97	0.77	0.43	0.33	0.00	0.97	1.00	1.00
	Specificità	0.01	0.38	0.69	0.72	1.00	0.56	0.68	0.83

Tabella 5.3: Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi di *statistical learning*. Insieme di stima generalizzato.

Anche in questo dataset si può notare come la classificazione migliori applicando i metodi di *statistical learning* alle matrici di covarianza campionarie costruite con *super trial*: l'accuratezza infatti aumenta per tutti i metodi del 20-30%, con l'unica eccezione costituita dalla SVM (Tabella 5.6).

Soggetto	Indice	Senza <i>super trial</i>				Con <i>super trial</i>			
		SVM	oRF	Bagging	XGBoost	SVM	oRF	Bagging	XGBoost
17	Accuratezza	0.45	<u>0.54</u>	0.52	0.52	0.50	0.69	0.74	<u>0.81</u>
	Sensibilità	0.89	0.69	0.35	0.39	0.00	0.89	0.89	0.89
	Specificità	0.01	0.39	0.69	0.65	1.00	0.48	0.60	0.73
18	Accuratezza	<u>0.52</u>	0.46	0.47	0.50	0.84	0.75	0.83	<u>0.89</u>
	Sensibilità	1.00	0.53	0.20	0.40	0.93	0.90	0.90	0.97
	Specificità	0.04	0.39	0.75	0.59	0.75	0.59	0.76	0.81
19	Accuratezza	0.49	0.55	0.49	<u>0.56</u>	0.84	0.80	0.87	<u>0.92</u>
	Sensibilità	0.97	0.73	0.37	0.63	0.87	1.00	1.00	0.97
	Specificità	0.01	0.36	0.62	0.48	0.81	0.59	0.73	0.87
20	Accuratezza	0.49	<u>0.62</u>	0.48	0.43	<u>0.91</u>	0.77	0.90	<u>0.91</u>
	Sensibilità	0.97	0.80	0.28	0.23	0.97	0.88	0.97	0.95
	Specificità	0.01	0.45	0.69	0.65	0.84	0.66	0.83	0.87
21	Accuratezza	0.51	0.54	0.47	<u>0.57</u>	0.90	0.76	0.91	<u>0.92</u>
	Sensibilità	1.00	0.69	0.28	0.56	1.00	0.91	0.97	1.00
	Specificità	0.03	0.39	0.67	0.57	0.81	0.61	0.85	0.83
22	Accuratezza	0.48	0.45	0.50	<u>0.52</u>	<u>0.89</u>	0.83	0.88	<u>0.89</u>
	Sensibilità	0.91	0.50	0.23	0.50	0.97	0.97	0.97	0.97
	Specificità	0.05	0.41	0.77	0.54	0.81	0.70	0.78	0.80
23	Accuratezza	0.50	0.52	0.47	0.61	<u>0.91</u>	0.75	0.87	<u>0.91</u>
	Sensibilità	1.00	0.53	0.20	0.57	1.00	0.93	0.97	1.00
	Specificità	0.00	0.50	0.74	0.65	0.83	0.57	0.78	0.82
24	Accuratezza	0.41	0.50	<u>0.57</u>	0.56	0.50	0.67	0.74	<u>0.84</u>
	Sensibilità	0.44	0.57	0.41	0.43	0.00	0.94	0.98	0.93
	Specificità	0.39	0.43	0.73	0.69	1.00	0.40	0.49	0.76
Indici medi	Accuratezza	0.48	<u>0.55</u>	0.52	0.54	0.75	0.75	0.82	<u>0.87</u>
	Sensibilità	0.87	0.63	0.32	0.47	0.62	0.89	0.87	0.91
	Specificità	0.08	0.46	0.72	0.60	0.88	0.61	0.76	0.84

Tabella 5.4: Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi di *statistical learning*. Insieme di stima generalizzato.

Infine, si può constatare che, come per il dataset precedente, i classificatori che fanno uso delle matrici di covarianza costruite con *super trial* mostrano i risultati migliori, sia con l'insieme di training generalizzato, che con quello specifico per ogni soggetto.

Soggetto	Indice	Senza <i>super trial</i>				Con <i>super trial</i>			
		SVM	oRF	Bagging	XGBoost	SVM	oRF	Bagging	XGBoost
8	Accuratezza	0.49	0.50	0.55	<u>0.57</u>	0.50	0.83	0.83	<u>0.86</u>
	Sensibilità	0.00	1.00	0.47	0.40	1.00	0.95	0.87	0.84
	Specificità	0.97	0.00	0.62	0.75	0.00	0.72	0.80	0.87
9	Accuratezza	0.61	0.61	<u>0.63</u>	0.51	0.50	0.75	0.81	<u>0.83</u>
	Sensibilità	0.46	0.70	0.56	0.26	1.00	0.86	0.90	0.84
	Specificità	0.76	0.52	0.70	0.76	0.00	0.64	0.72	0.82
10	Accuratezza	0.49	<u>0.55</u>	0.50	0.52	0.50	0.71	0.76	<u>0.78</u>
	Sensibilità	0.03	0.72	0.47	0.53	1.00	0.80	0.80	0.70
	Specificità	0.94	0.38	0.54	0.50	0.00	0.62	0.72	0.86
11	Accuratezza	0.58	0.56	0.50	<u>0.60</u>	0.50	0.63	0.71	<u>0.75</u>
	Sensibilità	0.48	0.79	0.45	0.54	1.00	0.82	0.75	0.71
	Specificità	0.68	0.34	0.56	0.66	0.00	0.44	0.68	0.78
12	Accuratezza	0.48	<u>0.54</u>	0.53	0.52	0.50	0.63	0.71	<u>0.72</u>
	Sensibilità	0.64	0.59	0.36	0.27	1.00	0.65	0.66	0.67
	Specificità	0.33	0.49	0.70	0.77	0.00	0.60	0.76	0.77
13	Accuratezza	0.38	<u>0.57</u>	0.52	0.56	0.50	0.74	0.78	<u>0.83</u>
	Sensibilità	0.48	0.69	0.35	0.40	1.00	0.83	0.81	0.81
	Specificità	0.28	0.45	0.68	0.72	0.00	0.65	0.75	0.84
14	Accuratezza	0.48	<u>0.56</u>	<u>0.56</u>	0.54	0.50	0.61	0.68	<u>0.77</u>
	Sensibilità	0.12	0.58	0.47	0.37	1.00	0.73	0.78	0.78
	Specificità	0.85	0.53	0.65	0.71	0.00	0.49	0.57	0.76
15	Accuratezza	0.46	<u>0.53</u>	0.52	0.47	0.50	0.59	0.72	<u>0.77</u>
	Sensibilità	0.11	0.61	0.36	0.25	1.00	0.62	0.82	0.77
	Specificità	0.81	0.46	0.68	0.70	0.00	0.56	0.61	0.77
16	Accuratezza	0.46	0.48	<u>0.54</u>	0.45	0.50	0.79	0.81	<u>0.85</u>
	Sensibilità	0.07	0.73	0.47	0.20	1.00	0.97	0.97	0.97
	Specificità	0.86	0.23	0.62	0.70	0.00	0.62	0.65	0.73

Tabella 5.5: Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi di *statistical learning*. Insieme di stima specifico per ogni soggetto.

Soggetto	Indice	Senza <i>super trial</i>				Con <i>super trial</i>			
		SVM	oRF	Bagging	XGBoost	SVM	oRF	Bagging	XGBoost
17	Accuratezza	0.45	0.47	<u>0.58</u>	0.56	0.50	0.75	0.69	<u>0.76</u>
	Sensibilità	0.27	0.61	0.54	0.50	1.00	0.92	0.89	0.89
	Specificità	0.62	0.33	0.62	0.62	0.00	0.58	0.48	0.63
18	Accuratezza	<u>0.55</u>	0.41	0.47	0.40	0.50	0.70	0.79	<u>0.86</u>
	Sensibilità	0.50	0.53	0.30	0.17	1.00	0.87	1.00	0.97
	Specificità	0.60	0.28	0.63	0.63	0.00	0.54	0.58	0.75
19	Accuratezza	0.41	<u>0.59</u>	0.51	0.57	0.50	0.79	<u>0.86</u>	<u>0.86</u>
	Sensibilità	0.13	0.83	0.40	0.43	1.00	1.00	0.90	0.97
	Specificità	0.69	0.34	0.63	0.71	0.00	0.59	0.82	0.76
20	Accuratezza	<u>0.56</u>	0.53	0.51	0.51	0.50	0.74	0.84	<u>0.89</u>
	Sensibilità	0.62	0.68	0.68	0.50	1.00	0.90	0.97	0.95
	Specificità	0.51	0.38	0.35	0.53	0.00	0.58	0.70	0.82
21	Accuratezza	0.45	0.49	0.44	<u>0.52</u>	0.50	0.74	<u>0.87</u>	0.86
	Sensibilità	0.41	0.56	0.25	0.25	1.00	0.91	0.97	0.94
	Specificità	0.49	0.42	0.64	0.79	0.00	0.58	0.76	0.79
22	Accuratezza	0.42	0.49	<u>0.57</u>	0.54	0.50	0.79	0.83	<u>0.90</u>
	Sensibilità	0.41	0.85	0.59	0.35	1.00	0.94	0.88	0.97
	Specificità	0.43	0.14	0.55	0.72	0.00	0.63	0.78	0.83
23	Accuratezza	0.43	<u>0.58</u>	0.56	0.48	0.50	0.70	0.78	<u>0.83</u>
	Sensibilità	0.37	0.60	0.57	0.20	1.00	0.97	1.00	0.97
	Specificità	0.49	0.56	0.55	0.77	0.00	0.43	0.56	0.69
24	Accuratezza	0.51	0.52	<u>0.53</u>	0.52	0.50	0.68	<u>0.81</u>	<u>0.81</u>
	Sensibilità	0.28	0.44	0.26	0.15	1.00	0.72	0.89	0.81
	Specificità	0.74	0.59	0.81	0.88	0.00	0.64	0.72	0.81
Indici medi	Accuratezza	0.48	<u>0.53</u>	<u>0.53</u>	0.52	0.50	0.72	0.78	<u>0.82</u>
	Sensibilità	0.32	0.68	0.44	0.34	1.00	0.85	0.87	0.86
	Specificità	0.65	0.38	0.62	0.70	0.00	0.58	0.69	0.78

Tabella 5.6: Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi di *statistical learning*. Insieme di stima specifico per ogni soggetto.

5.4.2 Metodi basati sulla geometria Riemanniana

Per l'applicazione dei metodi di classificazione basati sulla geometria Riemanniana non è stato necessario vettorizzare le matrici di covarianza dato che si va a lavorare direttamente sulla varietà di cui fanno parte: in particolare, le matrici costruite senza l'uso di *super trial* giacciono in $\mathcal{P}_{16}$, data la presenza di 16 canali di rilevazione, mentre quelle costruite con il *super trial* appartengono a $\mathcal{P}_{32}$, varietà a 32 dimensioni. L'unico metodo che necessitava della vettorizzazione delle unità da classificare è la SVM con kernel Riemanniano, in cui le matrici vengono trasformate appositamente per sfruttare l'algoritmo delle SVM come spiegato nel Paragrafo 3.2.1. Nella sua applicazione è stata fatta una *grid search* per identificare il migliore parametro di costo da usare ed è stato necessario parallelizzare il suo calcolo per ridurre il tempo di *training*. L'algoritmo MDRM, il *Rapporto di Verosimiglianza* e tutte le funzioni necessarie per il loro funzionamento sono state implementate in R ed inserite nel pacchetto **RiemannR** scaricabile da <https://github.com/LauraMasiero/RiemannR>. Esse sono elencate in Appendice A.

Dalla Tabella 5.7 è possibile notare che, per quanto riguarda i modelli stimati con l'insieme di stima generalizzato, l'accuratezza bilanciata migliore appartiene sia al *Minimum Distance to Riemannian Mean* (MDRM) e al *Rapporto di Verosimiglianza* (RV), in cui essa raggiunge il 90%; se i modelli vengono stimati con l'insieme di stima specifico per ogni soggetto, MDRM supera di poco RV, mantenendo sempre un'accuratezza del 90%.

Come visto per i metodi di *statistical learning*, anche i modelli basati sulla geometria Riemanniana danno risultati migliori se applicati alle matrici di covarianza costruite con *super trial*, notando un miglioramento di circa il 40% nell'accuratezza della classificazione. Solo la performance della SVM con kernel Riemanniano non varia nei due casi (Tabelle 5.7 e 5.8).

Comunque, anche in questo caso, i classificatori dati da MDRM ed RV raggiungono i migliori risultati se applicati alle matrici di covarianza campionaria costruite con *super trial*, sia con l'utilizzo dell'insieme di stima generalizzato che con quello specifico per ogni soggetto.

Soggetto	Indice	Senza <i>super trial</i>			Con <i>super trial</i>		
		SVM-RK	MDRM	RV	SVM-RK	MDRM	RV
1	Accuratezza	0.49	<u>0.51</u>	0.50	0.61	0.91	<u>0.92</u>
	Sensibilità	0.35	0.29	1.00	0.78	0.87	0.90
	Specificità	0.63	0.72	0.00	0.44	0.94	0.94
2	Accuratezza	0.49	<u>0.59</u>	0.50	0.63	<u>0.93</u>	0.91
	Sensibilità	0.41	0.64	1.00	0.77	0.92	0.86
	Specificità	0.58	0.53	0.00	0.49	0.94	0.96
3	Accuratezza	0.50	<u>0.55</u>	0.50	0.61	0.82	<u>0.83</u>
	Sensibilità	0.43	0.60	1.00	0.78	0.92	0.93
	Specificità	0.57	0.50	0.00	0.44	0.72	0.73
4	Accuratezza	<u>0.52</u>	0.50	0.50	0.63	0.88	<u>0.89</u>
	Sensibilità	0.41	0.82	1.00	0.77	0.94	0.96
	Specificità	0.62	0.19	0.00	0.48	0.83	0.81
5	Accuratezza	0.49	<u>0.56</u>	0.50	0.61	<u>0.94</u>	0.93
	Sensibilità	0.36	0.88	1.00	0.76	0.98	0.99
	Specificità	0.63	0.23	0.00	0.47	0.90	0.87
6	Accuratezza	<u>0.50</u>	<u>0.50</u>	<u>0.50</u>	0.60	0.92	<u>0.93</u>
	Sensibilità	0.39	0.04	1.00	0.77	0.88	0.91
	Specificità	0.61	0.97	0.00	0.43	0.96	0.96
7	Accuratezza	<u>0.51</u>	<u>0.51</u>	0.50	0.60	<u>0.91</u>	<u>0.91</u>
	Sensibilità	0.42	0.06	1.00	0.70	0.85	0.84
	Specificità	0.60	0.97	0.00	0.51	0.97	0.97
Indici medi	Accuratezza	0.50	<u>0.53</u>	0.50	0.61	<u>0.90</u>	<u>0.90</u>
	Sensibilità	0.39	0.48	1.00	0.76	0.91	0.91
	Specificità	0.61	0.59	0.00	0.47	0.89	0.89

Tabella 5.7: Dataset 1. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima generalizzato.

Per finire, riguardo il dataset 2, è possibile osservare che la migliore *performance* in termini di accuratezza è data da MDRM: esso infatti si rivela il migliore classificatore sia nell'applicazione alle matrici di covarianza costruite con *super trial* che senza ed in entrambe le varianti dell'insieme di stima.

Soggetto	Indice	Senza <i>super trial</i>			Con <i>super trial</i>		
		SVM-RK	MDRM	RV	SVM-RK	MDRM	RV
1	Accuratezza	0.51	<u>0.52</u>	0.50	0.60	<u>0.94</u>	<u>0.94</u>
	Sensibilità	0.27	0.33	1.00	0.81	0.94	0.94
	Specificità	0.74	0.71	0.00	0.38	0.93	0.94
2	Accuratezza	0.49	<u>0.51</u>	0.50	0.59	<u>0.78</u>	0.73
	Sensibilità	0.63	0.10	1.00	0.57	0.61	0.50
	Specificità	0.36	0.91	0.00	0.61	0.95	0.96
3	Accuratezza	0.51	<u>0.55</u>	0.50	0.49	<u>0.78</u>	<u>0.78</u>
	Sensibilità	0.09	0.56	1.00	0.63	0.93	0.96
	Specificità	0.92	0.53	0.00	0.35	0.63	0.60
4	Accuratezza	0.48	<u>0.55</u>	0.50	0.51	<u>0.97</u>	<u>0.97</u>
	Sensibilità	0.81	0.34	1.00	0.54	0.96	0.96
	Specificità	0.14	0.76	0.00	0.48	0.97	0.97
5	Accuratezza	0.50	<u>0.56</u>	0.50	0.53	<u>0.94</u>	<u>0.94</u>
	Sensibilità	0.76	0.58	1.00	0.48	0.98	0.98
	Specificità	0.25	0.55	0.00	0.58	0.91	0.91
6	Accuratezza	0.51	<u>0.58</u>	0.50	0.50	<u>0.95</u>	0.94
	Sensibilità	0.15	0.48	1.00	0.80	0.96	0.92
	Specificità	0.86	0.69	0.00	0.19	0.93	0.96
7	Accuratezza	0.49	<u>0.57</u>	0.50	0.45	<u>0.95</u>	<u>0.95</u>
	Sensibilità	0.35	0.43	1.00	0.51	0.96	0.96
	Specificità	0.63	0.71	0.00	0.39	0.93	0.93
Indici medi	Accuratezza	0.50	<u>0.55</u>	0.50	0.52	<u>0.90</u>	0.89
	Sensibilità	0.44	0.40	1.00	0.62	0.91	0.89
	Specificità	0.56	0.69	0.00	0.43	0.89	0.90

Tabella 5.8: Dataset 1. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima specifico per ogni soggetto.

Anche in questo caso, come tutti quelli visti in precedenza, i risultati migliori si ottengono utilizzando le matrici di covarianza con *super trial*, che consentono, per esempio, di aumentare la capacità previsiva di circa il 40% per quanto riguarda il classificatore MDRM; solamente la *performance* della SVM con kernel Riemanniano rimane simile.

Soggetto	Indice	Senza <i>super trial</i>			Con <i>super trial</i>		
		SVM-RK	MDRM	RV	SVM-RK	MDRM	RV
8	Accuratezza	0.47	<u>0.53</u>	0.50	0.46	<u>0.97</u>	0.94
	Sensibilità	0.45	0.24	1.00	0.79	1.00	1.00
	Specificità	0.49	0.83	0.00	0.14	0.95	0.89
9	Accuratezza	0.45	<u>0.55</u>	0.50	0.50	<u>0.93</u>	0.89
	Sensibilità	0.44	0.30	1.00	0.88	0.96	0.98
	Specificità	0.46	0.80	0.00	0.12	0.90	0.81
10	Accuratezza	0.50	<u>0.55</u>	0.50	0.53	<u>0.87</u>	0.78
	Sensibilità	0.52	0.55	1.00	0.92	0.95	0.95
	Specificità	0.48	0.55	0.00	0.13	0.79	0.61
11	Accuratezza	0.51	<u>0.57</u>	0.50	0.45	<u>0.93</u>	0.88
	Sensibilità	0.52	0.48	1.00	0.73	0.95	0.96
	Specificità	0.49	0.66	0.00	0.16	0.92	0.80
12	Accuratezza	0.50	<u>0.53</u>	0.50	0.50	<u>0.84</u>	0.78
	Sensibilità	0.53	0.43	1.00	0.85	0.97	0.99
	Specificità	0.47	0.63	0.00	0.15	0.71	0.57
13	Accuratezza	0.51	<u>0.55</u>	0.50	0.49	<u>0.90</u>	0.86
	Sensibilità	0.50	0.38	1.00	0.85	0.92	0.96
	Specificità	0.52	0.72	0.00	0.13	0.89	0.77
14	Accuratezza	0.46	<u>0.54</u>	0.50	0.45	<u>0.90</u>	<u>0.90</u>
	Sensibilità	0.42	0.55	1.00	0.47	0.93	0.97
	Specificità	0.49	0.54	0.00	0.43	0.87	0.83
15	Accuratezza	0.49	<u>0.56</u>	0.50	0.52	<u>0.85</u>	<u>0.85</u>
	Sensibilità	0.48	0.66	1.00	0.73	0.88	0.96
	Specificità	0.51	0.47	0.00	0.31	0.83	0.74
16	Accuratezza	0.51	<u>0.56</u>	0.50	0.52	<u>0.99</u>	<u>0.99</u>
	Sensibilità	0.57	0.17	1.00	0.93	1.00	1.00
	Specificità	0.45	0.95	0.00	0.11	0.99	0.98

Tabella 5.9: Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima generalizzato.

Soggetto	Indice	Senza <i>super trial</i>			Con <i>super trial</i>		
		SVM-RK	MDRM	RV	SVM-RK	MDRM	RV
17	Accuratezza	0.40	<u>0.50</u>	<u>0.50</u>	0.34	<u>0.99</u>	<u>0.99</u>
	Sensibilità	0.31	0.00	1.00	0.58	1.00	1.00
	Specificità	0.50	0.99	0.00	0.11	0.98	0.98
18	Accuratezza	0.62	<u>0.53</u>	0.50	0.50	<u>0.98</u>	0.96
	Sensibilità	0.70	0.10	1.00	0.87	1.00	1.00
	Specificità	0.54	0.95	0.00	0.14	0.97	0.92
19	Accuratezza	0.49	0.47	<u>0.50</u>	0.46	<u>0.99</u>	0.98
	Sensibilità	0.47	0.27	1.00	0.87	1.00	1.00
	Specificità	0.51	0.68	0.00	0.05	0.98	0.96
20	Accuratezza	<u>0.51</u>	0.49	0.50	0.47	<u>0.99</u>	0.95
	Sensibilità	0.55	0.03	1.00	0.78	1.00	1.00
	Specificità	0.47	0.95	0.00	0.17	0.99	0.91
21	Accuratezza	0.45	<u>0.55</u>	0.50	0.53	<u>0.99</u>	0.95
	Sensibilità	0.41	0.44	1.00	0.94	1.00	1.00
	Specificità	0.50	0.67	0.00	0.13	0.98	0.90
22	Accuratezza	0.49	<u>0.51</u>	0.50	0.51	<u>0.96</u>	0.94
	Sensibilità	0.53	0.18	1.00	0.88	0.97	1.00
	Specificità	0.46	0.84	0.00	0.14	0.96	0.88
23	Accuratezza	0.49	<u>0.60</u>	0.50	0.48	<u>0.99</u>	0.96
	Sensibilità	0.50	0.60	1.00	0.87	1.00	1.00
	Specificità	0.47	0.59	0.00	0.10	0.99	0.91
24	Accuratezza	0.45	<u>0.62</u>	0.50	0.49	0.80	<u>0.83</u>
	Sensibilità	0.44	0.69	1.00	0.85	0.67	0.74
	Specificità	0.46	0.56	0.00	0.14	0.94	0.93
Indici medi	Accuratezza	0.49	<u>0.54</u>	0.50	0.48	<u>0.94</u>	0.91
	Sensibilità	0.49	0.36	1.00	0.81	0.95	0.97
	Specificità	0.49	0.73	0.00	0.16	0.92	0.85

Tabella 5.10: Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima generalizzato.

Soggetto	Indice	Senza <i>super trial</i>			Con <i>super trial</i>		
		SVM-RK	MDRM	RV	SVM-RK	MDRM	RV
8	Accuratezza	0.47	<u>0.56</u>	0.50	0.58	<u>0.97</u>	<u>0.97</u>
	Sensibilità	0.18	0.26	1.00	0.58	0.97	0.97
	Specificità	0.76	0.85	0.00	0.57	0.96	0.96
9	Accuratezza	0.49	<u>0.57</u>	0.50	0.43	<u>0.93</u>	0.88
	Sensibilità	0.32	0.32	1.00	0.24	0.94	0.98
	Specificità	0.66	0.82	0.00	0.62	0.91	0.78
10	Accuratezza	0.52	<u>0.55</u>	0.50	0.61	<u>0.81</u>	0.52
	Sensibilità	0.52	0.41	1.00	0.59	0.81	1.00
	Specificità	0.52	0.68	0.00	0.63	0.80	0.04
11	Accuratezza	0.55	<u>0.59</u>	0.50	0.58	<u>0.82</u>	0.50
	Sensibilità	0.79	0.71	1.00	0.86	0.95	1.00
	Specificità	0.32	0.46	0.00	0.30	0.69	0.00
12	Accuratezza	<u>0.53</u>	0.50	0.50	0.49	<u>0.71</u>	0.50
	Sensibilità	0.52	0.27	1.00	0.06	0.68	1.00
	Specificità	0.54	0.72	0.00	0.93	0.74	0.00
13	Accuratezza	0.50	<u>0.56</u>	0.50	0.56	<u>0.93</u>	0.65
	Sensibilità	0.46	0.48	0.00	0.54	0.92	1.00
	Specificità	0.55	0.63	1.00	0.57	0.93	0.30
14	Accuratezza	0.49	<u>0.50</u>	<u>0.50</u>	0.50	<u>0.86</u>	0.50
	Sensibilità	0.53	0.22	1.00	0.58	0.87	1.00
	Specificità	0.44	0.78	0.00	0.42	0.86	0.00
15	Accuratezza	0.53	<u>0.56</u>	0.50	0.46	<u>0.85</u>	0.67
	Sensibilità	0.73	0.38	1.00	0.23	0.96	1.00
	Specificità	0.33	0.74	0.00	0.69	0.73	0.34
16	Accuratezza	0.52	<u>0.58</u>	0.50	0.47	<u>0.99</u>	0.93
	Sensibilità	0.63	0.77	1.00	0.10	1.00	1.00
	Specificità	0.41	0.39	0.00	0.84	0.99	0.86

Tabella 5.11: Dataset 2, soggetti da 8 a 16. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima specifico per ogni soggetto.

Soggetto	Indice	Senza <i>super trial</i>			Con <i>super trial</i>		
		SVM-RK	MDRM	RV	SVM-RK	MDRM	RV
17	Accuratezza	<u>0.58</u>	0.54	0.50	0.60	<u>0.98</u>	0.86
	Sensibilità	0.39	0.65	1.00	0.31	1.00	1.00
	Specificità	0.77	0.43	0.00	0.90	0.97	0.73
18	Accuratezza	<u>0.52</u>	0.48	0.50	0.50	<u>0.98</u>	0.50
	Sensibilità	0.67	0.47	0.00	1.00	1.00	0.00
	Specificità	0.37	0.49	1.00	0.00	0.95	1.00
19	Accuratezza	0.50	<u>0.54</u>	0.50	0.63	<u>0.94</u>	0.50
	Sensibilità	1.00	0.90	0.00	0.43	1.00	0.00
	Specificità	0.00	0.19	1.00	0.82	0.89	1.00
20	Accuratezza	0.50	<u>0.52</u>	0.50	0.63	<u>0.98</u>	0.93
	Sensibilità	0.20	0.75	0.00	0.35	1.00	0.85
	Specificità	0.80	0.28	1.00	0.92	0.97	1.00
21	Accuratezza	<u>0.61</u>	0.57	0.50	0.42	<u>0.97</u>	0.59
	Sensibilità	0.50	0.56	1.00	0.22	0.94	1.00
	Specificità	0.72	0.58	0.00	0.62	1.00	0.18
22	Accuratezza	<u>0.62</u>	0.47	0.50	0.50	<u>0.97</u>	0.50
	Sensibilità	0.91	0.53	0.00	0.41	0.97	0.00
	Specificità	0.33	0.40	1.00	0.59	0.96	1.00
23	Accuratezza	0.49	<u>0.53</u>	0.50	0.55	<u>0.99</u>	0.50
	Sensibilità	0.80	0.73	0.00	0.30	1.00	0.00
	Specificità	0.19	0.33	1.00	0.80	0.98	1.00
24	Accuratezza	0.57	<u>0.60</u>	0.50	0.56	<u>0.71</u>	0.62
	Sensibilità	0.69	0.54	1.00	0.28	0.72	1.00
	Specificità	0.45	0.67	0.00	0.84	0.71	0.24
Indici medi	Accuratezza	0.53	<u>0.54</u>	0.50	0.53	<u>0.91</u>	0.65
	Sensibilità	0.58	0.53	0.65	0.42	0.93	0.75
	Specificità	0.48	0.56	0.35	0.65	0.88	0.55

Tabella 5.12: Dataset 2, soggetti da 17 a 24. Risultati ottenuti tramite classificazione con i metodi basati sulla geometria Riemanniana. Insieme di stima specifico per ogni soggetto.

5.4.3 Risultati salienti

Appare chiaro che le metodologie che apportano una migliore classificazione siano quelle basate sulla geometria Riemanniana, come il *Minimum Distance to Riemannian Mean* (MDRM) ed il *Rapporto di Verosimiglianza*. L'eccezione che si presenta è data dalla *Support Vector Machine* con kernel Riemanniano che fornisce risultati peggiori, in tutti i casi studiati, anche di alcuni metodi di *statistical learning*, come la *Model-Based Oblique Random Forest*, il *Bagging* e *XGBoost*.

Tra i metodi Riemanniani il più affidabile sembra MDRM: in alcuni casi infatti raggiunge un'accuratezza bilanciata ed una specificità del 99% ed una sensibilità di addirittura il 100%. Esso inoltre dà ottimi risultati in dataset numerosi, come quelli generalizzati, ma supera ogni aspettativa con pochi dati, come con l'insieme di training specifico per ogni soggetto nel dataset 2, in cui si ha solamente una sessione.

Un altro aspetto importante da considerare è l'uso del *super trial* per la costruzione delle matrici di covarianza campionarie. Dalle analisi svolte si può concludere che i metodi di classificazione che fanno uso delle matrici di covarianza costruite con il *super trial* hanno una migliore *performance* degli stessi che fanno uso delle matrici di covarianza campionaria "semplici".

Come si può vedere in Figura 5.7 si ha una migliore discriminazione con le matrici costruite con i *super trial*, specialmente con il dataset generalizzato (pannello in alto a destra), rispetto a quelle costruite senza, che come si vede dai grafici nella parte sinistra, sono più mescolate tra loro, rimanendo prevalentemente in un lato della frontiera decisionale.

Tutto ciò è molto evidente per il soggetto 8, e soprattutto nell'ultimo pannello in basso a destra, in cui il baricentro Riemanniano è stato stimato nel dataset specifico per soggetto e in cui la classificazione raggiunge i migliori risultati come si vede in Tabella 5.11.

Tutto ciò che si può vedere in Figura 5.7 rispecchia quanto visto nelle analisi esplorative nelle Figure 5.3 e 5.5, e nella loro versione in 3D nelle Figure 5.4, 5.6, in cui le matrici di covarianza create con *super trial*, specialmente quelle dell'evento *target*, creano *cluster*: in questo modo il baricentro Riemanniano è stimato in modo più preciso e consente una discriminazione migliore nell'insieme di verifica.

Infine, per quanto riguarda la scelta del tipo di *training set* da utilizzare tra una sessione per ogni soggetto (caso dataset 2, *training* singolo soggetto), otto sessioni per soggetto (caso dataset 1, *training* singolo soggetto) e il dataset generalizzato è possibile notare che:

- per i metodi di *statistical learning* non ci sono cambiamenti di rilievo nelle *performance* dalla loro applicazione all'insieme di training generalizzato, con otto sessioni o con solo una sessione per soggetto, specialmente per quanto riguarda l'applicazione alle matrici costruite con *super trial*. Da notare solamente che la SVM applicata alle matrici calcolate con *super trial* fornisce risultati migliori con il dataset generalizzato;
- per i classificatori basati sulla geometria Riemanniana si ha similarità dei risultati ottenuti in tutte le configurazioni dell'insieme di training; l'unica cosa che si nota è che la *performance* della SVM con kernel Riemanniano applicata alle matrici di covarianza costruite con *super trial* è migliore se applicata al *training set* generalizzato nel dataset 1, ottenendo un'accuratezza del 61% , mentre nel dataset 2 l'accuratezza si rivela leggermente più alta utilizzando l'insieme di training specifico per soggetto. La classificazione attuata dal metodo RV invece peggiora nel dataset appena accennato rispetto a quello generalizzato, suggerendo che per questo metodo una sola sessione non è sufficiente per stimare la differente variabilità delle classi.

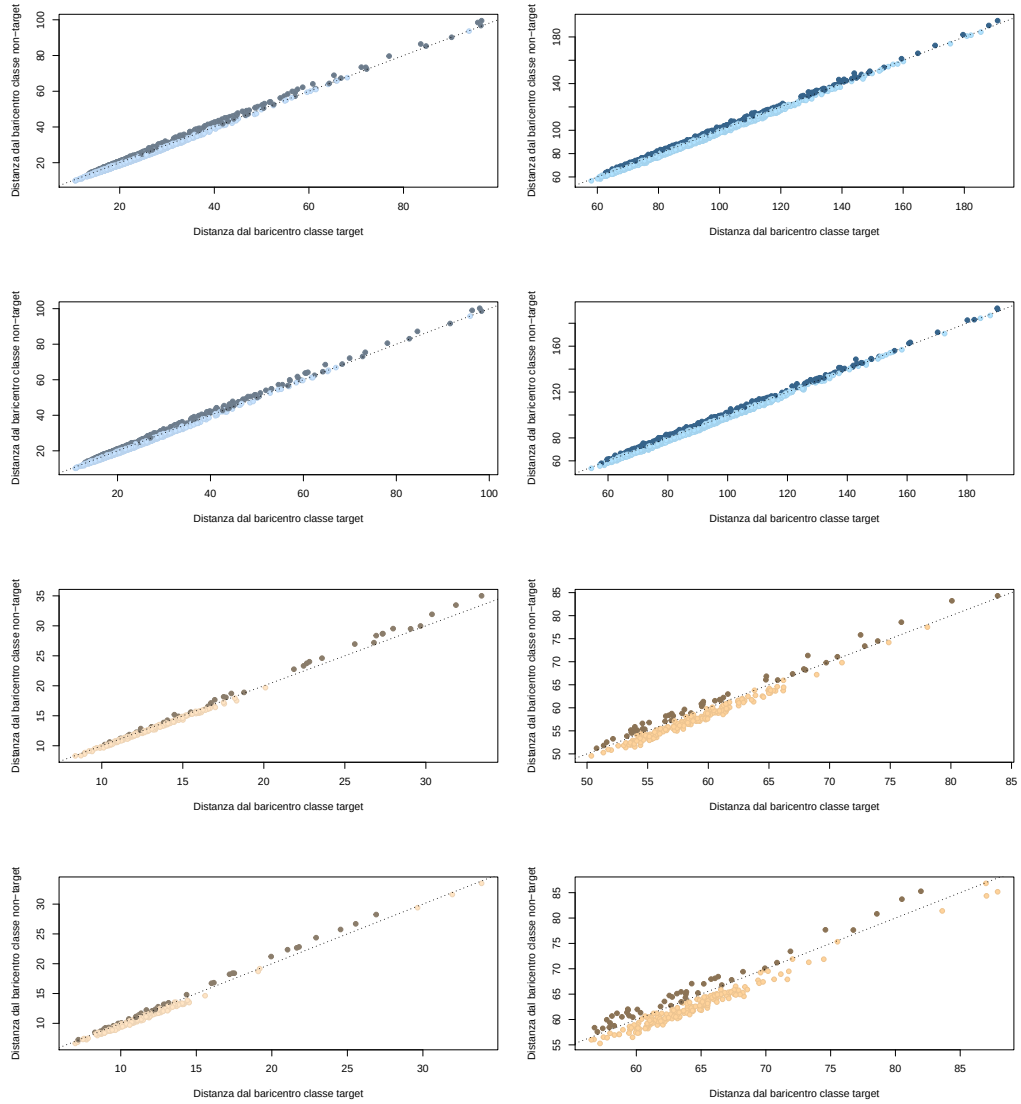


Figura 5.7: Diagramma di dispersione delle osservazioni dell'insieme di verifica: distanze dal baricentro di ciascuna classe (*target* e *non-target* partendo da matrici di covarianza costruite senza *super trial* (a sinistra) e con (a destra). In alto baricentri calcolati nell'insieme di stima generalizzato, in basso specifico per ogni soggetto. In colore chiaro la classe *non-target*, in colore scuro la classe *target*. Soggetto 1 in sfumature di blu, soggetto 8 in sfumature di ocra.

Conclusioni

Le analisi svolte in questa tesi hanno lo scopo di confrontare, con diverse sfumature, i metodi di *statistical learning* e quelli basati sulla geometria differenziale, in particolare Riemanniana, nella classificazione delle matrici di covarianza per il funzionamento del gioco ***Brain Invaders*** descritto nel Capitolo 4.

Per fare quanto appena detto si è cercato di seguire le idee e le proposte metodologiche sviluppate da Said et al. (2015), Congedo et al. (2012) e Barachant et al. (2013), in modo da comprendere ed implementare in R metodi per la classificazione più idonei per questo tipo di dati che giacciono su varietà di diverse dimensioni, a seconda del numero di canali utilizzati per la registrazione del segnale elettroencefalografico e cercare il confronto con metodi già molto utilizzati, come il *Bagging* e la *Support Vector Machine*, ma anche con modifiche di metodi già esistenti, come la *Model-Based Oblique Random Forest* e *eXtreme Gradient Boosting*.

In presenza di *Brain-Computer Interfaces* basate su potenziali evento-relati come ***Brain Invaders***, le conclusioni che si possono trarre alla fine di questo lavoro sono numerose e di diversa natura, per questo motivo si è cercato di riassumere le principali:

- l'algoritmo migliore si rivela essere il *Minimum Distance to Riemannian Mean*, MDRM. Esso è anche il più veloce da un punto di vista computazionale, aspetto molto importante in questo caso, dato che la classificazione è necessaria per lo svolgimento del gioco;

- alcune volte MDRM e il classificatore basato sul *rapporto di verosimiglianza*, RV, offrono gli stessi risultati: ciò avviene perché la variabilità stimata nelle due classi è uguale, riportandoci ad un confronto tra distanze, e quindi portandoci all'analisi MDRM. Nel secondo dataset con *training* specifico per ogni soggetto, è possibile inoltre notare che la classificazione basata su RV non dà risultati buoni quanto quelli forniti da MDRM: ciò accade perché è disponibile solamente una sessione per soggetto, che risulta insufficiente per avere una buona stima della variabilità intra classe;
- se si dispone già di sessioni di gioco pregresse per un soggetto, è possibile inizializzare l'algoritmo di classificazione, per esempio MDRM, con esse e non necessariamente con molti dati, anche provenienti da diversi soggetti, perché le *performance* di classificazione restano pressoché stabili;
- se un nuovo soggetto si presta all'uso del videogioco, l'inizializzazione del classificatore che porta maggior accuratezza finale è quella data da un dataset di training vario, vale a dire un dataset in cui sono raccolte poche sessioni di tanti soggetti diversi. Altrimenti, è possibile far registrare una sessione ed allenare il classificatore su di essa, infatti la bontà della *performance* in questo caso non diminuisce di molto;
- l'uso di *super trial* nella costruzione delle matrici di covarianza campionaria migliora in tutti i casi studiati la bontà della classificazione.

In conclusione, sarebbe interessante vedere se, considerando le matrici di covarianza costruite a partire dai *super trial*, isolando solo il blocco che apporta maggiori informazioni sia possibile migliorare ulteriormente la bontà di classificazione e rendere più veloce il classificatore, data la minore quantità di dati utilizzati.

Appendice A

Codice R

A.1 Importazione dei dati in formato .mat

```
1 library(R.matlab)

3 #####Insieme di stima#####

5 folder <- "~/Desktop/Tesi/Preprocessed_data/mat/Non-adaptive/Train"
  temp <- list.files(folder, pattern="*.mat")
7 non_adaptive_train <- lapply(temp, readMat)

9 #####Insieme di verifica#####

11 folder <- "~/Desktop/Tesi/Preprocessed_data/mat/Non-adaptive/Online"
  temp <- list.files(folder, pattern="*.mat")
13 non_adaptive_online <- lapply(temp, readMat)

15 #####Estrazione delle etichette di gruppo (target - non-target)#####

17 #####Insieme di stima #####

19 response <- lapply(non_adaptive_train, function (x) x$labels )
  response <- lapply(response, as.vector)
21
  #####Insieme di verifica #####
23
  responsev <- lapply(non_adaptive_online, function (x) x$labels )
25 responsev <- lapply(responsev, as.vector)
```

A.2 Funzioni create per il pacchetto RiemannR

```

1 #####SCM#####

3 SCM<- function(matrix){
  t <- ncol(matrix)
5   scm <- (t-1)^(-1)*matrix % * % t(matrix)
  return(scm)
7 }

9 #####SUPER TRIAL#####

11 SuperTrial <- function(list, n=16, length_epochs=dim(list[[i]]$epochs)[3],
  tidy=F, col=17){
  P <- length(list)
13  mean_epoch <- matrix(NA, n, length_epochs)
  super_trial <- list()
15  scm_st <- list()
  if(tidy==F){
17    for(i in 1:P){
      new_trials <- list()
19      indices<- which(list[[i]]$labels==1, arr.ind = T)
      epochs <- list[[i]]$epochs[indices[,2], -col ,]
21      N <- sum(list[[i]]$labels)
      for(c in 1:n){
23        for(j in 1:129){
          mean_epoch[c,j] <- sum(epochs[,c,j])/N
25        }
      }
27      for(t in 1:dim(list[[i]]$epochs)[1]){
        new_trials[[t]] <- rbind(mean_epoch,list[[i]]$epochs[t,-col,] )
29        super_trial[[i]]<- new_trials
      }
31    }
  }
33  else{
    for(i in 1:P){
35      new_trials <- list()
      indices<- which(list[[i]]$labels==1, arr.ind = T)
37      epochs <- list[[i]]$epochs[indices[,2], ,]
      N <- sum(list[[i]]$labels) #dovrebbe essere sempre 80
39      for(c in 1:n){
        for(j in 1:129){
41          mean_epoch[c,j] <- sum(epochs[,c,j])/N
        }
43      }
      for(t in 1:dim(list[[i]]$epochs)[1]){
45        new_trials[[t]] <- rbind(mean_epoch,list[[i]]$epochs[t,,] )

```

```

        super_trial[[i]]<- new_trials
47     }
    }
49 }
    return(list(mean_epoch=mean_epoch, super_trial=super_trial))
51 }

53
#####RIEMANNIAN DISTANCE#####
55
RaoDist <- function(matrix1, matrix2){
57   (sum(log(eigen(solve(matrix1) % * % matrix2)$values)^2))
  }
59

61 #####VECT#####

63 vect <- function(matrix){
  require(lavaan)
65   d <- diag(diag(matrix))
  m <- (matrix-diag(diag(matrix)))*sqrt(2)
67   lav_matrix_vechr(m+d)
  }
69

71 #####LOGARITHMIC MAPPING#####

73 log_map <- function(scm_list, C){
  require(powerplus)
75   require(expm)
  C_star <- Matpow(C, 0.5,1)
77   C_2star <- Matpow(C, -0.5,1)
  mapped_values <- list()
79   for(i in 1:length(scm_list))
    mapped_values[[i]] <- C_star % * % logm(C_2star % * % scm_list[[i]] % *
      % C_2star) % * % C_star
81   return(mapped_values)
  }
83

85 #####EXPONENTIAL MAPPING#####

87 exp_map <- function(mapped_values, C){
  require(powerplus)
89   require(expm)
  C_star <- Matpow(C, 0.5,1)
91   C_2star <- Matpow(C, -0.5,1)
  orig_space_values <- list()
93   for(i in 1:length(mapped_values))
    orig_space_values[[i]] <- C_star % * % expm(C_2star % * % mapped_values

```

```

[[i]] % * % C_2star) % * % C_star
95   return(orig_space_values)
    }
97

99 #####GEOMETRIC MEAN#####

101 geometric_mean <- function(scm_list, eps=1/3){
    P <- length(scm_list)
103   cat("Initializing...", "\n")
    geo_mean <- P^(-1)*Reduce("+", scm_list) #inizializzazione
105   cat("Estimating geometric mean...", "\n")
    repeat{
107     x <- geo_mean
        S_hat <- P^(-1)*Reduce("+", log_map(scm_list, x)) #media aritmetica nello
            spazio tangente
109     geo_mean <- exp_map(S_hat, x)
        if(norm(S_hat, type="F")<eps) break
111   }
    cat("Geometric mean estimated.", "\n")
113   return(geo_mean)
    }
115

117 #####MINIMUM DISTANCE TO RIEMANNIAN MEAN#####

119 MDRM <- function(scm_list, centroids_list){
    P <- length(scm_list)
121   classes <- rep(0,P)
    distances <- array(NA, dim=c(P,length(centroids_list))) #centroids_list=
        LENGTH 2
123   for(m in 1:P){
        for(c in 1:length(centroids_list))
125         distances[m,c] <- RaoDist(centroids_list[[c]], scm_list[[m]])
    }
127   classes<- apply(distances, 1, which.min)
    class <- rep(0, P)
129   for(i in 1:P){
        if(classes[i]==1) class[i] <- 1
131   }
    return(list(classes=class, distances=distances))
133 }

135

#####SCM TRANSFORMATION FOR LINEAR SVM APPLICATION#####
137

scm_transform <- function(scm_list, centroid){
139   require(powerplus)
    require(expm)
141   C_2star <- Matpow(centroid, -0.5,1)

```

```

transformed_matrices <- list()
143 cat("Transforming matrices...", "\n")
for(i in 1:length(scm_list))
145   transformed_matrices[[i]] <- logm(C_2star % * % scm_list[[i]] % * % C_2
      star)
cat("Vectorizing matrices...", "\n")
147 vect_matrices <- lapply(transformed_matrices, vect)
vect_matrices <- lapply(vect_matrices, as.vector)
149 final_matrix<- matrix(NA, length(vect_matrices), length(vect_matrices
  [[1]]))
cat("Constructing final matrix...", "\n")
151 for(i in 1:length(vect_matrices))
  final_matrix[i,] <- unlist(vect_matrices[[i]])[1:length(vect_matrices[[i]
    ])]
153 cat("Done.", "\n")
return(final_matrix)
155 }

157
#####EMPIRICAL VARIANCE FUNCTION#####
159
emp_var_function <- function(mu, scm_list) {
161   P <- length(scm_list)
  P^(-1)*Reduce("+",lapply(scm_list, function(x) RaoDist(x, mu)))
163 }

165 eta <- function(sigma) -1/(2*sigma^2)

167
#####NORMALIZATION CONSTANT ESTIMATION####
169
zeta_est <- function(eta,n_channels,R){
171   require(mvtnorm)
  mu <- rep(0,n_channels)
173   Sigma <- diag(-1/eta,n_channels)
  set.seed(123)
175   val <- rmvnorm(R,mu,Sigma)
  temp <- rep(1,R)
177   for (i in 1:(n_channels-1))
    for (j in (i+1):n_channels)
179     temp <- temp*(sinh(abs(val[,i]-val[,j])/2))
  omegap <- (((pi^((n_channels^2)/2))*8^((n_channels*(n_channels-1))/4)))/(
    factorial(n_channels)*lmvgamma(n_channels/2,n_channels))
181   res <- omegap*mean(temp)
  res <- res*(2*pi)^(n_channels/2)*(det(Sigma))^(1/2)
183   return(res)
}
185
187

```

```

#####RIEMANN-GAUSS DISTRIBUTION#####
189
RG_distr <- function(mu,sigma, p, scm_matrix) zeta_est(eta(sigma),p,10000)
  ^(-1)*exp(-RaoDist(scm_matrix,mu)/(2*sigma^2))
191

193 LR_classification <- function(mu, sigma, scm_list){
  cat("Constructing lik_events...", "\n")
195   lik_events <- lapply(scm_list, function(x) RG_distr(mu[[1]],sigma[1],16, x
    ))
  cat("Constructing lik_0...", "\n")
197   lik_0 <- lapply(scm_list, function(x) RG_distr(mu[[2]],sigma[2],16, x))
  lik_ratios <- list()
199   classes <- rep(0,length(scm_list))
  for(i in 1:length(scm_list)){
201     lik_ratios[[i]] <- log(lik_events[[i]])-log(lik_0[[i]])
     if(lik_ratios[[i]]>=0) classes[i] <- 1
203     #cat("Iteration: ", i, "\t")
  }
205   return(list(lik_ratios=lik_ratios, classes=classes))
  }
207

209 #####CREATION OF SCMS#####

211 createAllSCM <- function(list, n=8, st=T, tidy=F, col=17) {
  tmp <- list()
213   final_list <- list()
  if(tidy==F){
215     if(st==T){
       for(j in 1:n){
217         tmp <- list()
         for(i in 1:length(list$super_trial[[j]])){
219           tmp[[i]] <- SCM(list$super_trial[[j]][[i]])
           final_list[[j]] <- tmp
221         }
       }
223     }
     else{
225       for(j in 1:n){
         tmp <- list()
227         for(i in 1:dim(list[[j]]$epochs)[1]){
           tmp[[i]] <- SCM(list[[j]]$epochs[i,-col,])
229           final_list[[j]] <- tmp
         }
231       }
     }
233   }
  else{
235     if(st==T){

```

```

    for(j in 1:n){
237       tmp <- list()
        for(i in 1:length(list$super_trial[[j]])){
239           tmp[[i]] <- SCM(list$super_trial[[j]][[i]])
           final_list[[j]] <- tmp
241       }
    }
243 }
    else{
245     for(j in 1:n){
        tmp <- list()
247         for(i in 1:dim(list[[j]]$epochs)[1]){
            tmp[[i]] <- SCM(list[[j]]$epochs[i,,])
249             final_list[[j]] <- tmp
        }
251     }
    }
253 }
    return(final_list)
255 }

257 #####GROUPS EXTRACTION#####

259 extractGroups <- function(list, dataset, events=1){
    group_list <- list()
261     if(events==1){
        for(t in 1:length(list)) {
263             pos <- which(dataset[[t]]$labels==1, arr.ind=T)[,2]
            for(j in 1:length(list[[t]])) group_list[[t]] <- list[[t]][pos]
265         }
    }
267     else {
        for(t in 1:length(list)) {
269             pos <- which(dataset[[t]]$labels==0, arr.ind=T)[,2]
            for(j in 1:length(list[[t]])) group_list[[t]] <- list[[t]][pos]
271         }
    }
273     return(group_list)
}

275 #####SCM VECTORIZATION AND MISC#####
277
    VectorizeSCM <- function(scm_list){
279     require(purrr)
        final_list <- lapply(flatten(scm_list), vect)
281     final_list <- lapply(final_list, as.vector)
        return(final_list)
283 }

```

285

```

    vectToMatrix <- function(scm_vect_list){
287   mat <- matrix(NA, length(scm_vect_list), length(scm_vect_list[[1]]))
      for(i in 1:length(scm_vect_list)) mat[i,] <- unlist(scm_vect_list[[i]])
289   return(mat)
    }
291

293 extractBlock <- function(scm_st_list){
      dimension <- unlist(lapply(scm_st_list, function(x) dim(x)[1]))
295   dim_block <- unlist(lapply(dimension, function(x) x/2))
      blocks <- list()
297   for(d in 1:length(dim_block)){
      new_mat <- matrix(NA, dim_block[d], dim_block[d])
299     for(i in 1:dim_block[d]){
      for(j in dim_block[d]:dimension[d]){
301       new_mat[j-dim_block[d],i] <- scm_st_list[[d]][j,i]
      }
303     }
      blocks[[d]] <- new_mat
305   }
      return(blocks)
307 }

```

Bibliografia

- Azzalini, Adelchi e Bruno Scarpa (2012). *Data Analysis and Data Mining: An Introduction*. Oxford University Press, USA. ISBN: 0199767106.
- Barachant, Alexandre, Stéphane Bonnet, Marco Congedo e Christian Jutten (2012). “Multiclass brain-computer interface classification by Riemannian geometry”. In: *IEEE Transactions on Biomedical Engineering*. ISSN: 00189294. DOI: 10.1109/TBME.2011.2172210.
- (2013). “Classification of covariance matrices using a Riemannian-based kernel for BCI applications”. In: *Neurocomputing*. ISSN: 09252312. DOI: 10.1016/j.neucom.2012.12.039.
- Barachant, Alexandre e Marco Congedo (2014). “A Plug&Play P300 BCI Using Information Geometry”. In: URL: <http://arxiv.org/abs/1409.0107>.
- Berger, Hans (1929). “Über das Elekrenkephalogram des Menschens”. In: *Archiv für Psychiatrie und Nervenkrankheiten* 87, pp. 570–570.
- Breiman, Leo (1997). *Arcing the edge*. Rapp. tecn. Statistics Department, University of California.
- (2001). “Random Forests”. In: *Machine Learning* 45.1, pp. 5–32.
- Chen, Tianqi e Carlos Guestrin (ago. 2016). “XGBoost: A scalable tree boosting system”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. 13-17-August-2016. Association for Computing Machinery, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785.
- Congedo, Marco, Matthieu Goyat, Nicolas Tarrin, Gelu Ionescu, Léo Varnet, Bertrand Rivet, Ronald Phlypo, Nisrine Jrad, Michael Acquadro e Christian Jutten (2011). “Brain Invaders: a prototype of an open-source

- P300- based video game working with the OpenViBE platform”. In: *5th International Brain-Computer Interface Conference*.
- Cortes, Corinna e Vladimir Vapnik (1995). “Support-Vector Networks”. In: *Machine Learning* 20.3, pp. 273–297. ISSN: 15730565. DOI: 10.1023/A:1022627411411.
- Farwell, L. A. e E. Donchin (1988). “Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials”. In: *Electroencephalography and Clinical Neurophysiology* 70.6, pp. 510–523. ISSN: 00134694. DOI: 10.1016/0013-4694(88)90149-6.
- Fonseca, C, J P Silva Cunha, R E Martins, V M Ferreira, J P Marques de Sá, M A Barbosa e A Martins da Silva (gen. 2007). “A novel dry active electrode for EEG recording.” In: *IEEE transactions on bio-medical engineering* 54.1, pp. 162–5. ISSN: 0018-9294. DOI: 10.1109/TBME.2006.884649. URL: <http://www.ncbi.nlm.nih.gov/pubmed/17260869>.
- Förstner, Wolfgang, Wolfgang Förstner e Boudewijn Moonen (1999). “A Metric for Covariance Matrices”. In: URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.733>.
- Freund, Yoav, Yoav Freund e Robert E. Schapire (1996). “Experiments with a new boosting algorithm”. In: *Thirteenth International Conference on Machine Learning*, pp. 148–156. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.380.9055>.
- Friedman, Jerome H (2001). “Greedy Function Approximation : A Gradient Boosting Machine”. In: *Annals of statistics*, pp. 1189–1232.
- Gibert, G, V Attina, J Mattout, E Maby e O Bertrand (2008). *Size enhancement coupled with intensification of symbols improves P300 Speller accuracy*. Rapp. tecn. Inserm, Lyon.
- Handy, Todd C. (2005). *Event-related potentials : a methods handbook*. MIT Press. ISBN: 9780262083331.
- Hastie, Trevor, Robert Tibshirani e Jerome Friedman (2009). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer.
- He, Bin (2005). *Neural engineering*. Kluwer Academic/Plenum. ISBN: 9780306486104.
- Hill, N. Jeremy, Thomas Navin Lal, Michael Schröder, Thilo Hinterberger, Guido Widman, Christian E. Elger, Bernhard Schölkopf e Niels Birbau-

- mer (2006). “Classifying Event-Related Desynchronization in EEG, ECoG and MEG Signals”. In: pp. 404–413. DOI: 10.1007/11861898_41. URL: http://link.springer.com/10.1007/11861898%7B%5C_%7D41.
- Jin, Jing, Brendan Z. Allison, Eric W. Sellers, Clemens Brunner, Petar Horki, Xingyu Wang e Christa Neuper (feb. 2011). “Optimized stimulus presentation patterns for an event-related potential EEG-based brain-computer interface”. In: *Medical and Biological Engineering and Computing* 49.2, pp. 181–191. ISSN: 01400118. DOI: 10.1007/s11517-010-0689-8.
- Klem, George H, Hans Otto Lüders, HH Jasper e C Elger (n.d.). *The ten-twenty electrode system of the International Federation*. Rapp. tecn.
- Lotte, F., M. Congedo, A. Lécuyer, F. Lamarche e B. Arnaldi (giu. 2007). *A review of classification algorithms for EEG-based brain-computer interfaces*. DOI: 10.1088/1741-2560/4/2/R01.
- Luck, Steven J. (Steven John) (2014). *An introduction to the event-related potential technique*. ISBN: 9780262525855.
- Major, Tyler C. e James M. Conrad (nov. 2014). “A survey of brain computer interfaces and their applications”. In: *Conference Proceedings - IEEE SOUTHEASTCON*. Institute of Electrical e Electronics Engineers Inc. ISBN: 9781479965854. DOI: 10.1109/SECON.2014.6950751.
- Martínez-Muñoz, Gonzalo, Daniel Hernández-Lobato, Alberto Suárez, G Martínez-Muñoz, D Hernández-Lobato e A Suárez (2008). “An Analysis of Ensemble Pruning Techniques Based on Ordered Aggregation”. In: DOI: 10.1109/TPAMI.2008.78. URL: <http://dx.doi.org/10.1109/TPAMI.2008.78>.
- Mason, Llew, Jonathan Baxter, Peter Bartlett e Marcus Frean (1999). *Boosting Algorithms as Gradient Descent in Function Space*. Rapp. tecn.
- McFarland, Dennis J., Laurie A. Miner, Theresa M. Vaughan e Jonathan R. Wolpaw (2000). “Mu and beta rhythm topographies during motor imagery and actual movements”. In: *Brain Topography* 12.3, pp. 177–186. ISSN: 08960267. DOI: 10.1023/A:1023437823106.
- Menze, Bjoern H., B. Michael Kelm, Daniel N. Splitthoff, Ullrich Koethe e Fred A. Hamprecht (2011). “On oblique random forests”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial In-*

- telligence and Lecture Notes in Bioinformatics*). Vol. 6912 LNAI. PART 2, pp. 453–469. ISBN: 9783642237829. DOI: 10.1007/978-3-642-23783-6_29.
- Meulman, Nienke, Bregtje Seton, Laurie Stowe e Monika Schmid (lug. 2015). “EEG and Event-Related Brain Potentials”. In: pp. 81–107. ISBN: 978-3-319-11528-3. DOI: 10.1007/978-3-319-11529-0.
- Moakher, Maher (2005). “A differential geometric approach to the geometric mean of symmetric positive-definite matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 26.3, pp. 735–747. ISSN: 08954798. DOI: 10.1137/S0895479803436937.
- Mondini, Valeria (2015). “Sviluppo di un sistema bci eeg-based per il controllo del movimento di un modello di arto superiore”. URL: <http://amslaurea.unibo.it/8461/>.
- Nicolas-Alonso, Luis Fernando e Jaime Gomez-Gil (2012). “Brain computer interfaces, a review.” In: *Sensors (Basel, Switzerland)* 12.2, pp. 1211–79. ISSN: 1424-8220. DOI: 10.3390/s120201211. URL: <http://www.ncbi.nlm.nih.gov/pubmed/22438708>.
- Nuwer, M R, G Comi, R Emerson, A Fuglsang-Frederiksen, J M Guérit, H Hinrichs, A Ikeda, F J Luccas e P Rappelsburger (mar. 1998). “IFCN standards for digital recording of clinical EEG. International Federation of Clinical Neurophysiology.” In: *Electroencephalography and clinical neurophysiology* 106.3, pp. 259–61. ISSN: 0013-4694. DOI: 10.1016/s0013-4694(97)00106-5. URL: <http://www.ncbi.nlm.nih.gov/pubmed/9743285>.
- Oostenveld, Robert e Peter Praamstra (2001). “The five percent electrode system for high-resolution EEG and ERP measurements”. In: *Clinical Neurophysiology* 112.4, pp. 713–719. ISSN: 13882457. DOI: 10.1016/S1388-2457(00)00527-7.
- Pfurtscheller, G., C. Brunner, A. Schlögl e F. H. Lopes da Silva (mag. 2006). “Mu rhythm (de)synchronization and EEG single-trial classification of different motor imagery tasks”. In: *NeuroImage* 31.1, pp. 153–159. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2005.12.003.
- Ravden, D e J Polich (ott. 1999). “On P300 measurement stability: habituation, intra-trial block variation, and ultradian rhythms.” In: *Biological*

- psychology* 51.1, pp. 59–76. ISSN: 0301-0511. URL: <http://www.ncbi.nlm.nih.gov/pubmed/10579421>.
- Renard, Yann, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand e Anatole Lécuyer (feb. 2010). “OpenViBE: An open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments”. In: *Presence: Teleoperators and Virtual Environments* 19.1, pp. 35–53. ISSN: 10547460. DOI: 10.1162/pres.19.1.35.
- Rieger, Josef (2006). *EEG signal analysis Study (Preparation for Doctoral Thesis)*. Rapp. tecn.
- Rodríguez, Juan J., Ludmila I. Kuncheva e Carlos J. Alonso (2006). “Rotation forest: A New classifier ensemble method”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10, pp. 1619–1630. ISSN: 01628828. DOI: 10.1109/TPAMI.2006.211.
- Said, Salem, Lionel Bombrun, Yannick Berthoumieu e Jonathan Manton (lug. 2015). “Riemannian Gaussian Distributions on the Space of Symmetric Positive Definite Matrices”. In: arXiv: 1507.01760. URL: <http://arxiv.org/abs/1507.01760>.
- Sen Purkayastha, Satyajit, V K Jain e H K Sardana (2014). “Topical Review: A Review of Various Techniques Used for Measuring Brain Activity in Brain Computer Interfaces”. In: *Advance in Electronic and Electric Engineering* 4.5, pp. 513–522.
- Sinclair, Christopher M., Mason C. Gasper e Andrew S. Blum (2007). “Basic electronics in clinical neurophysiology” in *The Clinical Neurophysiology Primer*. Springer, pp. 3–18.
- Taheri, B A, R T Knight e R L Smith (mag. 1994). “A dry electrode for EEG recording.” In: *Electroencephalography and clinical neurophysiology* 90.5, pp. 376–83. ISSN: 0013-4694. DOI: 10.1016/0013-4694(94)90053-1. URL: <http://www.ncbi.nlm.nih.gov/pubmed/7514984>.
- Takano, Kouji, Tomoaki Komatsu, Naoki Hata, Yasoichi Nakajima e Kenji Kansaku (ago. 2009). “Visual stimuli for the P300 brain-computer interface: a comparison of white/gray and green/blue flicker matrices.” In: *Clinical neurophysiology : official journal of the International Federa-*

- tion of Clinical Neurophysiology* 120.8, pp. 1562–6. ISSN: 1872-8952. DOI: 10.1016/j.clinph.2009.06.002. URL: <http://www.ncbi.nlm.nih.gov/pubmed/19560965>.
- Townsend, G., B. K. LaPallo, C. B. Boulay, D. J. Krusienski, G. E. Frye, C. K. Hauser, N. E. Schwartz, T. M. Vaughan, J. R. Wolpaw e E. W. Sellers (lug. 2010). “A novel P300-based brain-computer interface stimulus presentation paradigm: Moving beyond rows and columns”. In: *Clinical Neurophysiology* 121.7, pp. 1109–1120. ISSN: 13882457. DOI: 10.1016/j.clinph.2010.01.030.
- UK, Brain Products (2019). <http://www.brainproducts.uk/solutions/products/>. [Accesso fatto il 20 agosto 2019].
- Vaineau, Erwan, Alexandre Barachant, Anton Andreev, Pedro C. Rodrigues, Grégoire Cattan e Marco Congedo (apr. 2019). “Brain Invaders Adaptive versus Non-Adaptive P300 Brain-Computer Interface dataset”. In: arXiv: 1904.09111. URL: <http://arxiv.org/abs/1904.09111>.
- Van Der Maaten, Laurens e Geoffrey Hinton (2008). “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 1, pp. 1–48.
- Wikipedia (2019). *Electroencephalography* — *Wikipedia, L’enciclopedia libera*. <https://en.wikipedia.org/w/index.php?title=Electroencephalography&oldid=913194495>. [Accesso fatto il 20 agosto 2019].
- Wolpaw, Jonathan R, Niels Birbaumer, Dennis J Mcfarland, Gert Pfurtscheller e Theresa M Vaughan (2002). “Brain-computer interfaces for communication and control”. In: *Clinical Neurophysiology* 113, pp. 767–791.

Ringraziamenti

Finalmente il giorno è arrivato: scrivere queste frasi di ringraziamento rappresenta la conclusione di un percorso durato cinque anni. È stato un periodo di profondo apprendimento, non solo a livello scientifico, ma anche personale. Per questo vorrei ringraziare tutte le persone che mi hanno sostenuto durante questo periodo.

Prima di tutto, vorrei ringraziare la mia famiglia perché mi è sempre stata accanto e non mi ha mai fatto mancare il suo sostegno e aiuto nel corso di tutti questi anni. Senza di loro non sarei mai diventata ciò che sono e non avrei avuto la possibilità di inseguire i miei sogni. Un grazie speciale va a te, mamma, che mi sei sempre stata vicina, aiutandomi e confortandomi nei momenti più difficili, spingendomi a dare il meglio di me stessa.

Un sincero grazie va al Prof. Bruno Scarpa per i validi consigli che mi ha dato, la costanza con la quale mi ha sostenuto e guidato attraverso le difficoltà, e per aver creduto sempre nelle mie capacità.

Un grande ringraziamento al Prof. Livio Finos per aver stimolato il mio interesse per l'argomento qui discusso e per avermi dimostrato che anche dalle avversità possiamo trovare la motivazione per ripartire.

Tutti i miei amici hanno avuto un peso determinante nel conseguimento di questo traguardo. Grazie per aver condiviso con me in questi cinque anni le esperienze più importanti, vi voglio bene.

In particolare grazie Sara, per le interminabili, innumerevoli e divertenti pause che hanno costellato questi ultimi due anni: sono state indispensabili per portare a termine questo percorso. Grazie anche per avermi ascoltato nei momenti difficili e per avermi sopportato quando non ero sicura di niente.

Un grazie speciale va a Massimiliano, che ha sempre trovato il modo di tirarmi su il morale quando ero triste. Grazie per avermi portato svariati dolcetti dopo le lezioni di Statistica Computazionale, per le gite al Pam per comprare i gelati e per i momenti felici e spensierati in Asid.

Infine grazie Gianmarco, ricorderò sempre le domeniche in "aula studio" per ripassare Data Mining, la tua avversione per la biblioteca, la tua vivacità ed il tuo entusiasmo nel fare codice. Mi hai aiutata quando credevo di non farcela ed eri disponibile a fare una chiacchierata o a mangiare in Forcellini. Grazie mille di tutto. Statistica e tutti i miei colleghi-amici saranno custoditi per sempre nel mio cuore.