Introduzione

L'Interfaccia Neurale, meglio conosciuta con il termine Brain Computer Interface (BCI), è *"quell' insieme di dispositivi volti a dotare il cervello di un canale di output motorio artificiale, in funzione sostitutiva o aggiuntiva ai canali fisiologici"*. Il principio base di questa tecnologia è la possibilità di creare un collegamento, attraverso l'analisi del segnale EEG del paziente, diretto tra il cervello, o una parte del sistema nervoso centrale e un dispositivo elettronico che è in grado di interpretare tale segnale come l'intenzione del soggetto ed eseguirla.

In generale possiamo vedere un generico sistema di BCI come composto di tre unità:

1. L'unità di acquisizione del segnale
2. L'unità di elaborazione che traduce il segnale in istruzioni per la macchina
3. Unità di feedback che da un riscontro al soggetto del suo stato mentale.

Un'importante considerazione va fatta sulle modalità di acquisizione del segnale: le BCI sono principalmente rivolte a quei soggetti con gravi deficit motori che hanno bisogno di una assistenza continua per l'esecuzione di movimenti basilari, come per esempio il camminare, che si traduce nel poter direzionare una sedia rotelle senza il diretto intervento del soggetto, ma grazie ad un comportamento intelligente di essa, derivato da un'interfaccia BCI. Per questa ragione l'acquisizione del segnale deve essere non invasiva, portabile, sicura e soprattutto di facile utilizzo da parte del paziente; come l'elettroencefalografia.

Dall'EEG possono essere estratti diversi segnali di controllo; in questa sede ci occuperemo di una branchia dei cosiddetti SEP, i Potenziali Sensoriali Evocati, in particolare, dei VEP, Visual Evoked Potential, che sono quei potenziali rilevabili nella corteccia visiva dopo l'incorrenza di uno stimolo visivo. Di questi ci concentriamo sugli SSVEP, Steady State Visual Evoked Potential, generati in risposta a treni di stimoli ripetuti, che insorgono nella zona occipitale del cervello quando un pattern visivo viene ripetuto a una frequenza costante compresa tra i 5 e i 60-70 Hz. La particolarità degli SSVEP, infatti, è di possedere uno spettro con picchi di potenza costanti nel tempo corrispondenti alla frequenza di stimolo e alle sue armoniche.

Il metodo più utilizzato nell'ambito BCI SSVEP-Based consiste nel selezionare tramite il movimento degli occhi stimoli luminosi presentati al soggetto a diversa frequenza. Il comando impartito viene quindi interpretato e tradotto dalla BCI analizzando le caratteristiche del segnale legate alle frequenze di stimolazione, in modo tale da riconoscere tra tutti, quale il paziente stesse guardando. I comandi impartibili possono essere tanti quanti gli stimoli disponibili, con l'importante aggiunta di un comando nullo, dove l'utente non fornisce alcuna disposizione al sistema.

Molti sono i vantaggi legati all'utilizzo di questa tipologia di BCI come:

- Minimo addestramento da parte nel paziente nell'imparare ad utilizzare il dispositivo associato
- Facile set-up
- Il segnale SSVEP è facilmente rilevabile grazie ad un SNR notevolmente più elevato rispetto ad altri potenziali evocati ed alla sua natura frequenzialmente stabile
- Permette di impartire una grande quantità di comandi differenti (fino a 48).

Ma allo stesso tempo possiamo notare delle limitazioni, infatti: il paziente deve porre la sua attenzione sullo schermo che presenta gli stimoli visivi, il che potrebbe affaticarne la visione, compromettendo le future osservazioni e l'utilizzo del mezzo.

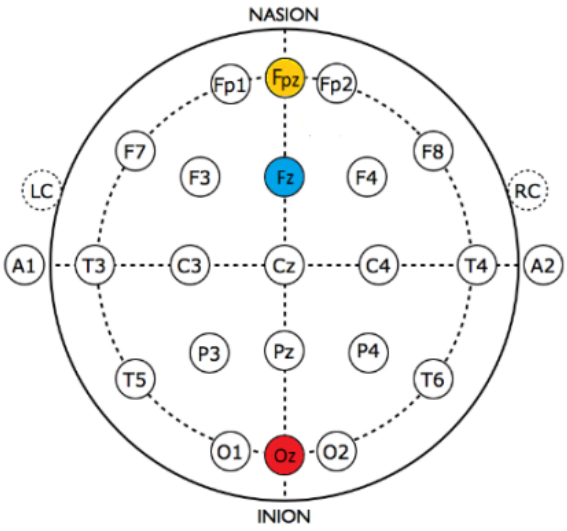
Il segnale così registrato risultata generalmente avere SNR molto basso, che rende davvero difficile estrarre il segnale di controllo di interesse, e dimensionalità molto elevata. Per questi motivi, sono necessari trattamenti del segnale come filtraggio in frequenza e un'analisi spettrale classica (FFT), che nonostante non siano di solito sufficienti, sono ancora molto utilizzati in questo tipo tecnologie.

Le feature estratte, principalmente con un'analisi di frequenza, perché ricordiamo che le SSVEP producono picchi in corrispondenza delle frequenze di stimolo e delle loro armoniche, vengono utilizzate per la classificazione del segnale. Gli algoritmi principalmente impiegati sono quelli di classificazione come clustering, SVM ed in alcuni casi reti neurali.

IL DATASET

La raccolta di segnali significativi per questo tipi di applicazione è preziosissima, in quanto la procedura di acquisizione necessita di mezzi appositi, la cui portabilità e popolarità sta crescendo negli ultimi anni. Per questo progetto di studio degli algoritmi di classificazione si fa uso del data AVI SSVEP, disponibile all'indirizzo https://www.setzner.com/avi-ssvep-dataset/#comment-14989.

I segnali sono stati raccolti nella seguente modalità: gli elettrodi sono stati posizionati seguendo lo standard 10-20, per cui l'elettrodo di segnale alla posizione Oz, in corrispondenza della corteccia visiva, la messa a terra in posizione Fz e il reference a Fpz:



Lo stimolo luminoso impiegato è stato riprodotto sun monitor LCD, BenQ XL2420T con refresh rate di 120 Hz. L'unico pre-processing applicato ai dati è un filtro notch a 50 Hz.

Il dataset è composto da due diversi tipi di misurazione:

1. Un primo data set, Single-target, dove sono presenti le registrazioni di diversi soggetti sottoposti a singoli stimoli a diverse frequenze, ognuno della durata di 30s secondi ciascuno.
2. Il secondo dataset, multi-target, sono presenti diverse misurazioni per altrettanti soggetti, diversi da quelli del primo dataset, dove lo stimolo non è singolo ma al soggetto sono posti davanti diversi stimoli a diverse frequenze, e il soggetto deve focalizzare l'attenzione su uno di questi per 16 secondi.

Preprocessing

Il segnale, considerando entrambe le modalità di acquisizione è stato sottoposto allo stesso tipo di pre-processing. Innanzitutto, tutti i segnali sono stati partizionati un sequenze di 5 secondi ciascuna in modo aumentare il quantitativo di dati disponibili ad allenare i predittori. Una finestra di 5 secondi è stata scelta dopo diverse prove, e notando che questa era la migliore in quanto non distorceva troppo il contenuto informativo presente nel segnale originale e permetteva di disporre di abbastanza dati per l'allenamento.

Per prima cosa ogni segnale presente nel dataset è stato già filtrato con un filtro notch a 50 Hz per rimuovere l'interferenza causata dalla frequenza alternata utilizzata nel sistema di alimentazione elettrica. Il filtro notch a 50 Hz è progettato per attenuare specificatamente questa frequenza e le sue armoniche senza influenzare significatamene altre frequenze nel segnale EEG.

Successivamente il segnale viene filtrato tramite un filtro passabanda di tipo Butterworth di quart'ordine, lasciando inalterate la banda di frequenze 4-35 Hz. Questo intervallo è stato scelto in modo da non alterare le frequenze di interesse e le armoniche principali, in quanto, in alcuni soggetti si registra un picco in frequenza maggiore in queste piuttosto che nella componente principale.

Features

Con feature selection si identifica quel processo utilizzato per identificare un sottoinsieme rilevante di caratteristiche da utilizzare per la creazione di un modello predittivo. Come già detto la caratteristica principale del segnale di controllo scelto, le SSVEP, è quello di essere molto stabili in frequenza e di avere picchi in prossimità delle frequenze di interesse. Per questa ragione un'analisi di frequenza tramite trasformata di Fourier è abbastanza per estrarre le feature necessarie alla classificazione.

Come features si è deciso di utilizzare le componenti in frequenza in diverse misure:

1. Il primo approccia si focalizza nell'utilizzare un singolo feature: la componente in frequenza di maggiore intensità, selezionata come la componente in frequenza con maggiore intensità nell'intervallo delle frequenze di interesse.
2. Dal momento che certi individui presentano più sensibilità nelle armoniche piuttosto che nella componente principale in frequenza, come secondo approccio ho deciso di utilizzare come features sia la componente principale che la prima armonica. Questa è stata estratta considerando l'intervallo di frequenze compreso tra prima armonica della frequenza minore e prima armonica della frequenza maggiore.
3. Come terza classe di features ho scelto di produrre, per ogni segnale un vettore, contenente, per ogni frequenza di interesse, la sua componente di intensità nel segnale corrente. Poiché tale frequenza potrebbe non essere perfettamente centrata, si considera una media delle intensità in un intorno di [- 0.1, +0.1] centrato nella frequenza bersaglio.

Classificazione

SVM – Support Vector Machines

Nell'ambito del Machine Learning, le Support Vector Machine sono modelli di classificazione nel contesto di apprendimento supervisionato: significa, cioè, che l'algoritmo produce un classificatore allenato a distinguere due o più classi, partendo da un insieme di dati le cui classi sono conosciute.

Definiamo innanzitutto l'idea delle SVM nel contesto di classificazione binaria, e successivamente nel contesto di classificazione multi-classe.

L'idea alla base delle SVM è quello di trovare un iperpiano che separi i dati, considerati d-dimensionali, cioè ogni oggetto è descritto da un insieme di d features, perfettamente in due classi. Poiché i dati reali sono difficilmente linearmente separabili, viene spesso impiegata la nozione di kernel che permette di considerare i dati in uno spazio di dimensioni maggiori rispetto a quello di partenza, dove i dati sono effettivamente separabili. Questa trasformazione non viene direttamente utilizzata in quanto la si oltrepassa per motivi di carico computazionale e di memoria tramite il cosiddetto "kernel trick" di cui dopo. Consideriamo un training set $S = \{(x1, y1), \ldots, (xm, ym)\}$ di cui ogni $x_i \in R^d$, ed ogni $y_i \in \{-1, 1\}$. Ogni iperpiano in R^d è parametrizzato da un vettore $w \in R^d$, ortogonale allo stesso iperpiano, e da una costante b :

$$w \cdot x + b = 0$$

La funzione utilizza la posizione di x rispetto all'iperpiano per predire il valore della sua y .

L'iperpiano così definito è detto non omogeneo in quanto non passa per l'origine, altrimenti avremo $b = 0$. In questo caso l'iperpiano è definito omogeneo. Ogni iperpiano non omogeneo $\{x \in R^d : w^T x = b\}$è equivalente all'iperpiano omogeneo $\{x \in R^{d+1} : v^T x = 0\}$, dove $v = (w, -b)$ e ad ogni x viene aggiunta una dimensione $x = (x, 1)$. Per questa ragione, senza perdita di generalità o funzionalità, consideriamo unicamente iperpiani omogenei.

La funzione che classifica i punti divide lo spazio di input in due semipiani, uno positivo ed uno negativo è:

$$h(x) = sgn(w \cdot x + b)$$

La quale è equivalente a:

$$h(x) = \begin{Bmatrix} 1 & se\ w \cdot x + b \geq 0 \\ -1 & se\ w \cdot x + b < 0 \end{Bmatrix}$$

L'iperpiano canonico, quello che separa i dati dall'iperpiano con una "distanza" pari a 1 è definito da:

$$\begin{Bmatrix} x \cdot w + b \geq 1 & se\ y = 1 \\ x \cdot w + b < 1 & se\ y = -1 \end{Bmatrix}$$

Oppure in forma compatta:

$$y(x \cdot w + b) \geq 1$$

La distanza considerata non deve essere intesa come distanza geometrica, è detta infatti margine dell'iperpiano; infatti, per definire una distanza geometrica di un punto del training set dall'iperpiano, è prima necessario normalizzare la norma di w:

$$d\big((w, b), x\big) = \frac{y(x \cdot w + b)}{||w||} \geq \frac{1}{||w||}$$

Quello che intuitivamente si vuole ottenere è quell'iperpiano che massimizzi la distanza dei punti più vicini ad esso. In questi termini, il problema di trovare l'iperpiano migliore può essere espresso in termini di un problema di ottimizzazione vincolata:

$$\min_{w \in R^d} \ \frac{1}{2} \left|\left|w\right|\right|^2$$

$$y_t w^T x_t \ \geq \ 1 \quad t \ = \ 1, \ \dots \ , \ m.$$

Tale problema può essere risolto in diverse maniere; in particolare, si può dimostrare che l'iperpiano ottimale può essere scritto come la combinazione lineare di un sottoinsieme di training data:

$$w \ = \ \sum_{t \in I} \ \alpha_t \, y_t x_t$$

Dove I è l'insieme di quei punti $(x_t \ , \ y_t)$che soddisfano l'equazione: $y_t w^T x_t \ = \ 1$; quei punti per cui il margine dell'iperpiano è effettivamente 1. Questi punti sono detti *support vectors*, e sono quegli unici punti che servono a definire l'iperpiano ottimo. In aggiunta, possiamo considerare i coefficienti α come un valore che determina l'importanza dell'attuale punto nel definire w. Se rimuoviamo tutti i punti del training set, lasciando unicamente i support vectors, la soluzione trovata per SVM non cambia.

Poiché nella maggior parte delle applicazioni reali i dati non sono linearmente separabili, è necessario implementare un approccio più generale che permetta all'algoritmo di fare qualche errore. Per integrare questa funzione, il problema di ottimizzazione diventa come segue:

$$\min_{(w, \, \xi) \in R^{d+m}} \ \frac{\lambda}{2} \left|\left|w\right|\right|^2 + \frac{1}{m} \sum_{t=1}^{m} \quad \xi_t$$

$$y_t w^T x_t \ \geq \ 1 \ - \ \xi \quad t \ = \ 1, \ \dots \ , \ m$$

$$\xi_t \ \geq \ 0 \qquad t \ = \ 1, \ \dots \ , \ m.$$

La componente ξ è detta variabile di supporto ed indica quanto ogni vincolo è violato da una soluzione potenziale del problema. Viene poi considerata da minimizzare la media delle variabili di supporto perché altrimenti la soluzione più semplice sarebbe quella di scegliere valori molto elevati per loro ed ogni vincolo sarebbe sempre soddisfatto. Questa media, questo fattore di penalizzazione, viene chiamato regolarizzatore: il risultato diventa in questo caso l'iperpiano che massimizza il margine ed al contempo ha il minor errore possibile. Il fattore $\lambda \ > \ 0$ è introdotto per bilanciare i due termini.

Possiamo inoltre aggiungere un coefficiente per pesare la somma totale delle variabili di supporto (al posto di pesarli come media): $\alpha \ , \ 0 \ < \ \alpha \ \leq C$. Dove possiamo distinguere due casi: se C è infinito allora l'iperpiano ottimale è quello che separa completamente i dati. Se, invece, C ha un valore finito, il problema diventa il cosiddetto "soft-margin classifier", dove alcuni dati possono essere classificati erroneamente. Se C viene inizializzato con un valore elevato allora viene data molta importanza a classificare tutti i punti correttamente, invece, per valori piccoli di C, il modello risulta più flessibile agli errori.

Quando i dati non sono linearmente separabili nel loro spazio dimensionale, lo potrebbero essere in uno spazio con più dimensioni: per esempio potrebbero essere separabili non con una retta, nel caso bidimensionale, ma con una curva polinomiale. Innanzitutto, definiamo una funzione per mappare ogni dato in una sua corrispondente nello spazio con più dimensioni:

$$z \ = \ \phi(x)$$

$$x \ \in \ R^d, \ z \ \in \ R^N, \ N \ \gg d$$

Per risolvere il problema legato ad SVM, basta scambiare ogni occorrenza di x con la sua corrisponde z, così da ottenere:

$$w = \sum_i \quad \alpha_i y_i \phi(x_i)$$

Ed al contempo:

$$h(x) = sgn\left(\sum_i \alpha_i\, y_i\left(\phi(x_i) \cdot \phi(x)\right) + b\right)$$

Notiamo che non è necessario calcolare per ogni x la sua mappatura, ma basta riuscire a trovare una formula per il calcolo del prodotto vettoriale nel nuovo spazio di valori, per non passare mai per la funzione $\phi(x)$;

$$K(x, x') = \phi(x) \cdot \phi(x')$$

In questo modo il problema può essere riscritto nei seguenti termini:

$$h(x) = sgn(\sum_i \quad \alpha_i y_i K(x_i, x) + b)$$

Il Kernel Polinomiale

Consideriamo il kernel polinomiale

$$K_p(x, x') = (x \cdot x' + 1)^p$$

Dove è il grado del polinomio che vogliamo utilizzare. In generale $p = 1 \sim 10$. Grazie a questo kernel possiamo impiegare una curva di grado maggiore al primo per dividere le classi, infatti, se il dataset non è linearmente separabile nel suo dominio iniziale, aumentandone la dimensionalità è probabile che lo diventi, così. In questo nuovo spazio, una retta che li divide si traduce in una curva polinomiale nello spazio di partenza, come un'ellissi o un cerchio.

Il kernel Gaussiano

$$K_\gamma(x, x') = \exp\left(-\frac{1}{2\gamma}\left||x - x'|\right|^2\right)$$

Possiamo vedere il kernel gaussiano come centrare per ogni esempio nel training set una gaussiana il cui raggio di influenza è stabilito dal parametro γ . Ogni nuovo dato viene classificato in base all'influenza che riceve da ogni gaussiana.

SVM Multiclasse

L'algoritmo di SVM può essere impiegato anche in un approccio multi -classe; in questo caso esistono due approcci possibili: one-vs-all e one-vs-one.

Nell'approccio "uno contro tutti", per classificare gli oggetti in K classi diverse, vengono costruiti K diversi classificatori binari. Ogni classe viene considerata singolarmente come la classe positiva, mentre tutte le altre classi coincidono nella classe negativa. In questo modo il problema converge in K problemi di classificazione binaria. Nel caso in cui la classificazione di un nuovo oggetto fosse ambigua, cioè, più di una classe positiva risulta la sua classe, vengono applicate certe euristiche per determinarne una classificazione determinata.

Nell'approccio "uno-contro-uno" invece, invece di distinguere una classe dalle altre, si cerca di discriminare tra coppie di classi; in questo caso viene allenato un classificatore per ogni coppia di classi, il che porta ad avere, per K classi distinte: $\frac{K(K-1)}{2}$ classificatori. Per predire un nuovo dato, questo viene passato ad ogni classificatore e ogni risultato viene registrato. La classe con più voti è quella assegnatagli.

Ensamble methods

I metodi di combinazione, o ensemble methods, sono algoritmi di classificazione che combinano il risultato di più modelli di apprendimento per migliorare le prestazioni complessive del sistema. L'idea alla base di queste tecniche è che l'integrazione di più modelli deboli può portare a risultati migliori rispetto all'impiego di un singolo modello. In particolare, questo genere di combinazioni può migliorare ("boosting") di molto quegli algoritmi più semplici di classificazione, quelli cioè le cui performance sono poco al di sopra di un classificatore random.

Una combinazione ("ensemble") contiene un certo numero di modelli chiamati "base learners", generati da un algoritmo d'apprendimento base, quali alberi di classificazione o reti neurali. La maggior parte di questi algoritmi utilizza un unico tipo di modello per i modelli più semplici.

Più precisamente, il termine boosting ("migliorare") si riferisce ad una famiglia di algoritmi che sono in grado combinare e convertire classificatori deboli in classificatori più forti: i primi sono riferiti a quei modelli le cui performance sono poco al di sopra di un modello random, mentre i secondi sono quei modelli più vicino ad un classificatore ottimale. Questi algoritmi costruiscono dei modelli in modo sequenziale, dove i successivi sono costruiti in modo da correggere quelli precedenti.

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

Per definire questo genere di algoritmi è necessario definire come la distribuzione dei dati di allenamento viene alterata ad ogni iterazione di boosting e come tutto l'insieme di predittori prodotto viene utilizzato per classificare un dato.

AdaBoost

AdaBoost è un algoritmo di boosting adattivo, cioè non necessita di conoscere a priori il numero di classificatori deboli da utilizzare. L'idea alla base di AdaBoost è quella di allenare sequenzialmente

un certo numero di modelli deboli usando ogni volta una versione alterata del training set iniziale. Queste modifiche vengono eseguite ad ogni iterazione e consistono nell'assegnare ad ogni esempio un certo coefficiente; inizialmente tutti i campioni sono equamente pesati. Ad ogni iterazione dell'algoritmo i campioni la cui classificazione risulta sbagliata da parte del modello debole attuale vedono i loro pesi aumentare, di modo che, con il procedere delle iterazioni, questi campioni vengano considerati maggiormente, così l'algoritmo è forzato a concentrarsi su quegli esempi che nel passo precedente hanno prodotto un errore.

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:
1. $\mathcal{D}_1(\boldsymbol{x}) = 1/m$. % Initialize the weight distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; % Train a classifier h_t from D under distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. **if** $\epsilon_t > 0.5$ **then break**
6. $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of h_t
7. $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$ % Update the distribution, where
% Z_t is a normalization factor which
% enables $\mathcal{D}_{t+1}$ to be a distribution
8. **end**

Output: $H(\boldsymbol{x}) = \mathtt{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

Come prima cosa l'algoritmo inizializza i pesi da assegnare a ciascun esempio nel training set originale, questi sono infatti pesati uniformemente e normalizzati di modo che la somma totale dei pesi sia 1. Ad ogni iterazione di boosting, viene allenato un modello debole sull'attuale distruzione dei dati pesati, il cui errore viene poi considerato nel calcolo di α, l'importanza assegnata al modello debole corrente. L'errore, generalmente una funzione di errore esponenziale, viene calcolato tenendo conto dei pesi degli esempi di addestramento e da maggiore importanza agli esempi che sono stati classificati erroneamente nei cicli precedenti.
Il peso di ogni esempio viene aggiornato in funzione delle performance raggiunte dal modello debole corrente: lo scopo è quello di aumentare il peso degli esempi classificati male, e ridurre il peso degli esempi classificati correttamente. È in questa fase che viene aggiornata la distribuzione degli esempi. Il temine Z_t è un fattore di normalizzazione utilizzato perché D_{t+1} sia effettivamente una distribuzione.
La predizione finale è costruita combinando le predizioni di tutti i modelli semplici:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

I modelli deboli che vengono utilizzati sono generalmente dei classificatori ad albero poco profondi. Un predittore ad albero è un predittore con una struttura di un albero ordinato ("ordered") e radicato ("rooted") dove ogni interno ha almeno due figli ed ha associato un test. Un test è una funzione che associa i possibili valori di una feature dei dati in ingresso ad uno dei nodi figlio:

$$f : X_i \rightarrow \{1, 2, \ldots, N\}$$

Ogni foglia è associata ad una classe; nel caso binario ogni foglia avrà associato un valore in {-1, 1}. Ogni dato che viene presentato al modello viene testato feature per feature ed attraversa l'albero fino ad una foglia, la cui classe associata è il valore della predizione sul dato.

Su quale feature costruire il successivo modello debole viene deciso tramite una misura di impurità: l'indice di Gini. Questa misura rappresenta la purezza della classificazione, in altri termini calcola la probabilità che una feature specifica sia erroneamente classificata quando selezionata randomicamente. In questi termini, ad ogni iterazione viene scelta la feature con indice di impurità minore sulla quale basare il modello.

Gli alberi che vengono utilizzati nell'algoritmo AdaBoost sono generalmente di piccole dimensioni, per esempio di profondità pari ad uno, cioè ogni albero testa un unico feature.

La versione utilizzata per la classificazione è chiamata Adaboost SAMME, *Stagewise Additive Modeling using a Multi-class Exponential loss function.* Questa versione comporta due principali modifiche al classico algoritmo di AdaBoost, oltre che adattarlo ad una classificazione multi-classe: gli esempi erroneamente classificati vedono il loro peso maggiormente incrementato e la combinazione finale per la predizione viene eseguita in modo leggermente diverso. In questo caso il peso dei modelli deboli è calcolato come:

$$\alpha_m \;=\; \log\left(\frac{1-err_m}{err_m}\right) \;+\; \log(K-1)$$

Mentre il coefficiente di ogni esempio:

$$w_i \;=\; w_i \cdot \exp\Big(\alpha_m \cdot \mathbb{I}\big(c_i \neq T_m(x_i)\big)\Big), \quad i \;=\; 1,\; \ldots,\; n$$

Dove *K* è il numero delle classi considerate, c_i la classe dell'*i*-esimo esempio e $T_m(x_i)$ la classificazione su x_i; se K=2, l'algoritmo si riduce ad essere AdaBoost classico. La presenza di questo termine permette sia di considerare la presenza delle altre classi nella classificazione sia di incrementare maggiormente il peso di un esempio mal classificato. Inoltre, questa versione dell'algoritmo non *break* quando il predittore presenta un errore maggiore di ½, perché comunque il coefficiente assegnato risulta positivo. La funzione che combina i modelli deboli è invece:

$$C(x) \;=\; \underset{k}{\arg max} \;\sum_{m=1}^{M} \alpha_m \cdot \mathbb{I}(T_m(x) \;=\; k)$$

Naive Bayes

Consideriamo ora l'impiego di un classificatore Bayesiano. Sia X un esempio rappresentato da un vettore di attributi $(x_1,\; \ldots,\; x_n)$, dove x_i è il valore dell'attributo X_i. Sia C la classe di variabili per la classificazione e c il suo valore. In questa definizione consideriamo solo il caso di classi binarie: + e -; l'estensione al caso di classificazione multi-classe è diretto. Dal punto di vista probabilistico, secondo il Teorema di Bayes, la probabilità di un esempio $X \;=\; (x_1,\; \ldots,\; x_n)$ di appartenere alla classe *c* è

$P(c \mid X) \;=\; \frac{P(E \mid c)P(c)}{P(X)}$.

X sarà classificato come positivo se:

$$h(X) \;=\; \frac{P(C \;=\; +\mid E)}{P(C \;=\; -\mid E)} \geq 1$$

Dove h(X) è il classificatore Bayesiano voluto. Se assumiamo che tutti gli attributi sono indipendenti dalla classe dell'esempio, allora

$$P(E \mid c) = P(x_1, \ldots, x_n \mid c) = \prod_{i=1}^{n} P(x_i \mid c)$$

Nel caso di classificazione multi-classe, la regola diventa:

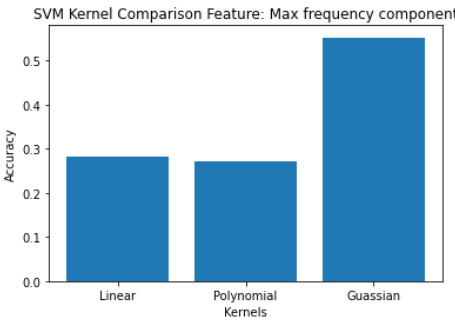
$$h(X) = \underset{y}{\arg\max}\, P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

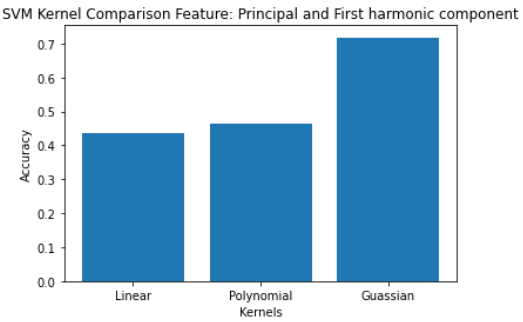
È cioè, la classe con la probabilità maggiore.

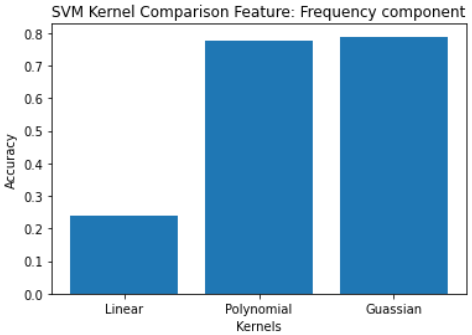
Il classificatore adottato è una particolare variante del caso Naive Bayes, in cui i valori associati agli attributi sono considerati estratti da una distribuzione gaussiana:

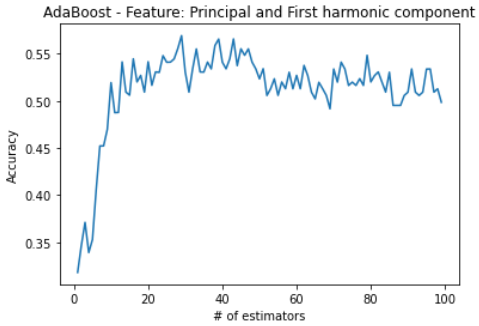
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma_y^2}\right)$$

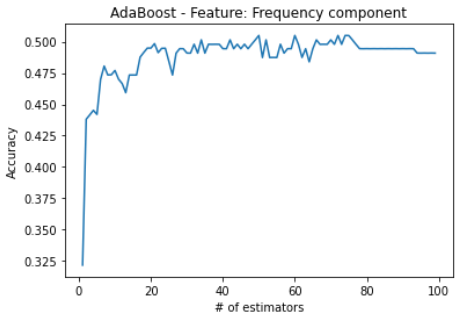
RISULTATI

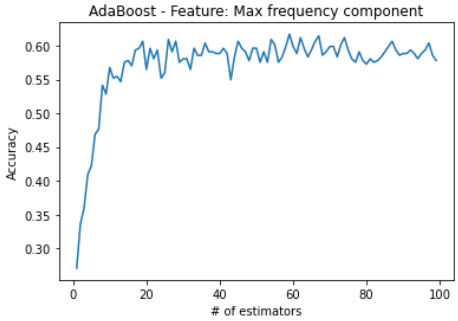












Feature	Max frequency component	Frequency component	Principal and First harmonic component
Accuracy	38.5416	78.8732	47.8873

Naive Bayes classifier accuracy table

I modelli prodotti danno risultati in accordo con il loro impiego: le SVM sono il modello che ha generato una classificazione migliore rispetto agli altri, soprattutto considerando la versione che impiega i kernel Gaussiano e polinomiale, i quali riescono a dividere i dati in uno spazio a più dimensioni rispetto che a quello originale. Si notano le scarse performance del classificatore Bayesiano, in quanto, questo, per performare al meglio, ha bisogno di molte features su cui condizionare l'etichetta di un esempio in input; infatti, laddove sono vengono utilizzate più features, questi raggiunge le stesse performance delle SVM. Notiamo inoltre che l'algoritmo AdaBoost migliora quando viene utilizzato su dati con meno features, rispetto che a dati con molte features. C'è inoltre da considerare il fatto che i dati presentati sono relativi a misurazioni EEG, i cui valori sono spesso rumorosi, e nonostante il passaggio di pre-processing, questa influenza la si nota in questo caso.