Interpretazione affettiva di feature audio

Marco Faleri

Abstract—Il progetto si presuppone lo scopo di ricercare quali feature audio siano piú rappresentative per un'interpretazione affettiva. Per questo è stato scelto un database molto usato in letteratura: il Berlin Database for Emotional speech, contenente frasi pronunciate da attori e categorizzate in 7 emozioni: gioia, rabbia, disgusto, paura, neutro, noia, tristezza. Da tali frasi sono state estratte varie feature, tra cui MFCC, pitch, root-mean-square energy e queste feature sono state valutate mediante diversi classificatori, con l'ulteriore scopo di paragonare le performance con la letteratura. In particolare sono stati usati K-NN, K-NN dopo PCA e SVM con (kernel lineare e rbf) i quali hanno ottenuto performance (misurate mediante cross-validaton) interessanti (72,3% di accuracy di K-NN dopo PCA, 83,1% di accuracy di SVM con kernel lineare). In conclusione è stata sviluppata una rete neurale Multi-Layer Perceptron che ha raggiunto la performance di 89,3% accuracy.

1 Introduzione

Il parlato è certamente uno dei principali mezzi di comunicazione tra esseri viventi. Specialmente quando non sono possibili informazioni visive il parlato diventa l'unica (o quasi) forma di comunicazione. Tale importanza nella vita biologica ha fatto sì che gli animali cominciassero a veicolare tramite il parlato ed i versi un'ampia pletora di informazioni, prime tra tutte le emozioni. In primis tramite gemiti e versi degli animali, in seguito con la comparsa dell'homo sapiens, si è sviluppato il linguaggio parlato, e tramite esso si sono quindi sviluppati altri metodi di comunicazione affettiva, il metodo semantico/verbale (legato al significato di *cosa* si dice) e prosodico/non verbale (correlato al *come* si dice qualcosa, ad esempio al ritmo del parlato ed al tono di voce). È in questo contesto che si inserisce la **Speech Emotion Recognition** (letteralmente: "Riconoscimento delle Emozioni a partire dal Parlato"), la branca dell'interazione naturale con lo scopo di comprendere le emozioni trasmesse dal parlato delle persone.

Importanza del problema: La speech emotion recognition risulta quindi un ambito di ricerca di fondamentale importanza per l'interazione naturale, potendo potenzialmente influire in molti degli ambiti applicativi dei modelli di computazione affettiva. Infatti non soltanto permette di aumentare l'accuratezza con cui si ricava l'emozione di un individuo dal suo comportamento, ma consente di coprire ambiti applicativi esclusivi, ovvero quelli in cui non è possibile ricavare informazioni visive: si pensi ad una telefonata.

Approccio seguito: Tale progetto è scomponibile in 4 parti principali:

1) La scelta di un Database, durante la quale sono stati cercati database molto usati in letteratura con lo scopo di poter paragonare il proprio lavoro con quanto fatto da altri, ulteriori criteri sono stati la reperibilità, intesa nel fatto che il database fosse liberamente scaricabile, e la qualità del segnale. A tali criteri ha corrisposto il Berlin Database for Emotional Speech [1]
2) La scelta e ricerca delle feature, in cui sono state ricercate non solo le feature standard per la speech emotion recognition, ma anche feature più sperimentali. Infine sono state ricercate librerie adibite all'estrazione di tali feature, in conclusione è stato scritto il notebook jupyter adibito ad estrarre le feature in diverse modalità: includendo la possibilità di fare zero-padding e data augmentation, e la possibilità di scegliere se estrarre le feature *globali* oppure le feature *locali* (pitch e mfcc sono stati previsti per l'estrazione *locale*)
3) La classificazione tramite algoritmi baseline e la rete MLP. Durante tale fase sono stati scritti i vari algoritmi di apprendimento e i conseguenti classificatori, tutti quanti validati tramite cross-validazione. Da K-NN, a K-NN eseguito dopo una PCA in cui sono stati mantenute le componenti con un explained variance pari al 99% o al 90% in caso di feature *locali*; SVM con kernel lineare e RBF; Gaussian Naive Bayes. Inoltre è stata sviluppata la rete neurale MLP tramite l'utilizzo della libreria tensorflow [2].
4) Le varie simulazioni per valutare l'efficienza delle feature. Durante la quale è stata testata l'efficacia di varie feature sui baseline e poi su quelle più promettenti è stata fatta girare la rete neurale.

Contributi:

- Studio di feature audio per l'analisi affettiva del parlato

2 Analisi dello stato dell'arte

La speech emotion recognition, pur essendo relativamente giovane come ambito di ricerca, presenta una florida letteratura. In particolare, parlando delle feature:

M. Faleri, Corso di Interazione Naturale e corso di Modelli di Computazione Affettiva, A/A 2017-2018, Università di Milano, via Comelico 39/41, Milano, Italy
E-mail: marco.faleri@studenti.unimi.it

sono considerate da tutti centrali le feature prosodiche [3], specialmente pitch, zero-crossing, magnitudine ed energia [4]. Tuttavia le feature prosodiche non sono le uniche considerate, si possono infatti distinguere altre tipologie di feature per l'analisi affettiva del parlato [5]:

- Feature basate sullo spettro: Solitamente le feature prosodiche sono strettamente correlate al dominio temporale, mentre possono fornire ottimi risultati anche feature correlate al dominio delle frequenze, in particolare MFCC (*Mel-Frequency Cepstral Coefficients*, che approfondiremo in 3).
- Feature relative alla voice quality: C'è un certo grado di correlazione tra la qualità della voce. Per qualità della voce s'intende sia il livello della voce (amplitudine, magnitudine) che l'aspetto qualitativo di essa (si pensi a voce roca o pulita): questo può essere particolarmente interessante per mostrare come le corde vocali siano stimolate nel parlato.

In conclusione risulta sempre spesso consigliato di utilizzare più tipi di feature per fare la classificazione, in particolare viene sottolineata l'efficacia degli MFCC per la classificazione tra le emozioni distinte, mentre le feature prosodiche sono particolarmente utili per distinguere tra alta e bassa attivazione.

Per quanto riguarda l'utilizzo di feature *globali* (ovvero feature che descrivono il segnale intero, in generale) oppure *locali* (feature che descrivono il singolo frame in cui sono state calcolate), in letteratura vengono ritenute più efficaci le feature globali [5]. Questo perché anche se possono dimostrarsi meno sensibili al segnale, in quanto molte delle variazioni al suo interno non vengono considerate, sono in un numero molto ridotto, e questo facilita il lavoro della maggior parte dei classificatori.

Per quanto riguarda la riduzione di dimensionalità, quando necessaria, sono state provate sia la PCA quanto la LDA, con esiti ambivalenti, dipendentemente dal paper e dalle feature considerate; nel fare questo, essendo le feature molto eterogenee, vengono solitamente normalizzate, sottraendone la media e dividendole per la deviazione standard.

Per quanto riguarda gli algoritmi di classificazione, sono utilizzati tutti gli algoritmi più noti: HMM, ANN ([3], [6]), GNN ed anche SVM ([7], [8]).

3 MODELLO TEORICO

Nell'entrare nel contesto del progetto, occorre definire accuratamente le feature che sono state utilizzate e studiate. Come visto in 2, ci sono diverse tipologie di feature potenzialmente interessanti, adesso saranno elencate quelle studiate in questo progetto:

- Magnitudine: La magnitudine rappresenta quanto un punto del segnale differisca da zero, a prescindere dalla negatività o positività di esso.
- Root-mean-square energy: La root-mean-square energy è definita semplicemente dalla seguente formula:

$$\sqrt{\frac{1}{N}\sum_{n}|x(n)|^2} \tag{1}$$

- Spectral Centroid: Il centroide dello spettro indica a quale frequenza l'energia dello spettro sia centrata in un determinato frame. Funziona sostanzialmente come una media pesata:

$$f_c = \frac{\sum_k S(k)f(k)}{\sum_k S(k)} \tag{2}$$

con S(k): magnitudine dello spettro al k-esimo bin di frequenza, f(k) frequenza del k-esimo bin.
- Spectral flatness: La spectral flatness serve ad indicare quanto un segnale sia *tone-like* piuttosto che simile al rumore bianco, in particolare indica il numero di picchi in uno spettro di potenza rispetto alla piattezza di questo.
Come possiamo vedere dalla formula, definendo P(k) come la magnitudine del k-esimo bin nello spettro di potenza, e K è il numero totale dei bin:

$$\text{Flatness} = \frac{\sqrt[K]{\prod_{k=0}^{K-1} P(k)}}{\frac{\sum_{k=0}^{K-1} P(k)}{K}} = \frac{\exp\left(\frac{1}{K}\sum_{k=0}^{K-1}\ln P(k)\right)}{\frac{1}{K}\sum_{k=0}^{K-1} P(k)} \tag{3}$$

La spectral flatness si ottiene pertanto dividendo la media geometrica del power spectrum per la media aritmetica di quest'ultimo, ergo valore 1 solo per power spectrum con valori costanti, in cui non sono presenti picchi (il white noise, appunto).
- Spectral contrast: Il contrasto spettrale è definito semplicemente come la differenza tra i picchi e le valli nello spettro. In particolare la libreria in cui l'ho trovato implementato utilizza una versione chiamata "Octave based spectral contrast", la quale, prima di calcolare i picchi e le valli dello spettro, suddivide attraverso filtri passa-banda il risultato della FFT in sotto-bande relative ad ottave, ad esempio nelle seguenti sotto-bande: [0hz, 200hz], [200hz, 400hz], [400hz, 800hz], [800hz, 1.6khz], [1.6khz, 3.2khz], [3.2khz, 8khz]. Per ciascuna di queste sottobande vengono calcolati dunque i picchi, le valli e la loro differenza, ovvero il contrasto.
- Zero crossing rate: Lo zero crossing indica quando un segnale attraversa il valore zero in un determinato frame, ovvero quando un segnale cambia segno. Per utilizzare tale feature globalmente calcoliamo lo zero crossing rate, ovvero calcoliamo il numero di zero crossing in ciascun frame e lo dividiamo per la dimensione totale del frame.

$$zcr = \frac{1}{N-1}\sum_{n=1}^{N-1}\chi_{\mathbb{R}_{<0}}(sgn(x(n))sgn(x(n-1))) \tag{4}$$

dove $\chi(\cdot)$ rappresenta la classica funzione indicatrice definita come

$$\chi_{\mathbb{R}_{<0}}(x) = \begin{cases} 0 & \text{se } x \geq 0 \\ 1 & \text{altrimenti} \end{cases}$$

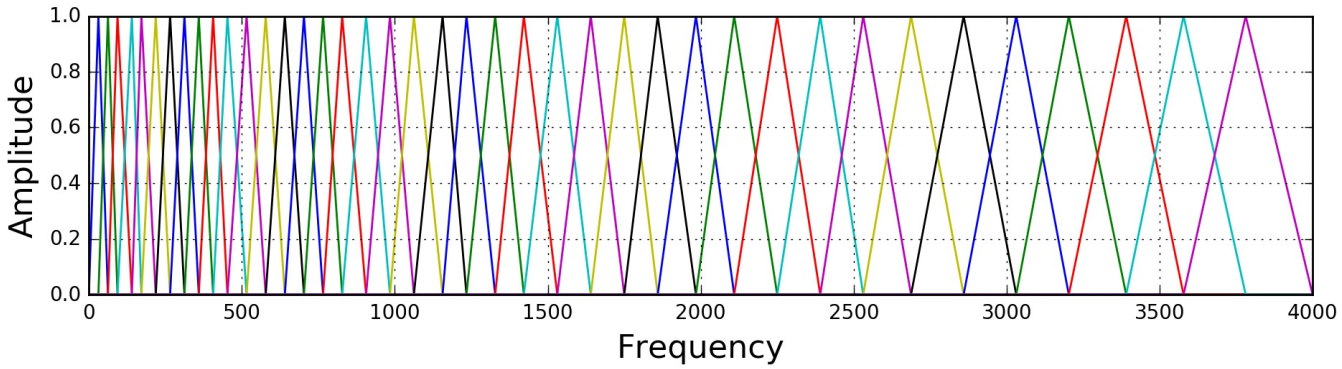

Fig. 1: Esempio di banco di filtri per la scala di mel

Ed infine ne facciamo una semplice media lungo tutto il segnale.

- Mel-scaled power spectogram: Power Spectrum in cui le bande di frequenza sono equalmente distribuite secondo la scala di Mel. La scala di Mel è una scala percettiva estrapolata empiricamente, con lo scopo di assegnare un valore ai suoni seguendo la percezione umana di questi.
  I **Mel** vengono definiti in relazione agli Hertz ed ai deciBel, ovvero si dice che due toni a 1000 Hz hanno una differenza di 1000 Mel se vengono **percepiti** con una distanza di 40 Db.
  In sostanza, la conversione tra Hertz e Mel viene fatta seguendo tali formule:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{5}$$

$$f = 700(10^{m/2595} - 1) \tag{6}$$

  Come vengono costruiti i filtri: ogni filtro del banco è triangolare, assumendo il valore 1 al centro della frequenza ad esso assegnata (tali frequenze sono appunto equalmente spaziate in mel, e di conseguenza tali filtri sono più discriminanti tra frequenze base e lo sono di meno tra frequenze basse) e descrescendo linearmente fino a raggiungere lo 0 nel centro di frequenza assegnati ai filtri adiancenti. Un chiaro esempio è la figura 1, in cui sono usati 40 filtri. Per applicare tali filtri al power spectogram basta moltiplicare il power spectrum per ogni filtro individualmente, e a questo punto il risultato di ciascun filtro può essere usato come feature (come in questo caso), oppure si possono sommare e quindi applicare un'altra trasformata per ottenere la feature seguente: i MFCC.
- MFCC: Come mostreremo più avanti, i Mel-Frequency Cepstral Coefficients sono la feature che più ha avuto successo nel classificare le emozioni, per questo diventa necessario trattare questa feature in particolare. Il Mel Frequency Cepstrum si ottiene applicando una trasformazione coseno discreta al logaritmo del periodogramma (scalato rispetto alla scala di mel) del segnale. Come avviene per il mel-scaled power spectogram, in MFC, le bande di frequenza sono equalmente distribuite secondo la di mel, la quale approssima la percezione auditiva dell'orecchio umano.
  Per essere più chiari, i MFCC sono calcolati classicamente seguendo la seguente procedura:
  1) Discrete Fourier transform sul segnale opportunamente finestrato.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} \tag{7}$$

     Qui k scorre il dominio delle frequenze, mentre n il dominio del tempo. N indica la lunghezza del frame.
  2) Calcolare il periodogramma, ricavabile dalla Fourier transform frame per frame con la semplice formula:

$$P = \frac{|DFT(x_i)|^2}{N} \tag{8}$$

     dove x_i indica l'i-esimo frame del segnale, N sono i punti in cui viene calcolata la Fourier transform. E mapparlo nella scala di mel, usanto opportuni filtri triangolari come visto in precedenza.
  3) Si prende il logaritmo del periodogramma risultante, sempre per emulare la percezione umana del suono.
  4) Per decorrelare il risultato ottenuto dai filtri e darne una rappresentazione compressa, si usa la trasformata coseno discreta (DCT) su di esso (andando quindi ad ottenere un cepstrum).

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \tag{9}$$

     con $k = 0, \ldots, N-1$
  5) I coefficienti risultanti sono gli MFCC, che solitamente vengono normalizzati rispetto alla media: sottraendo la media di ciascun coefficiente da tutti i frame.
- Cromagramma: Feature abbastanza sperimentale, solitamente utilizzata per lo studio della musica,

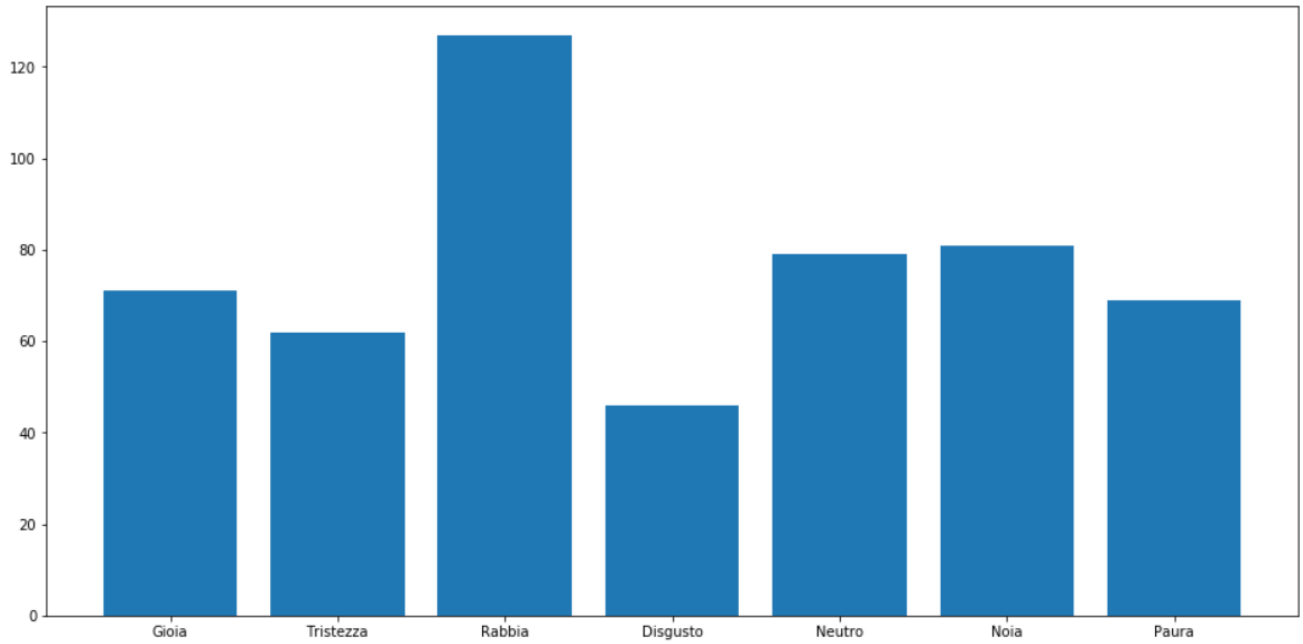

Fig. 2: Distribuzione delle etichette nel dataset

difatti si basa sul concetto di chroma. Una nota può essere definita tramite due componenti, la tonalità (tone height) ed il colore (chroma). Per tone height s'intende l'ottava a cui la nota appartene, mentre per "chroma" si intende il valore assunto dalla nota all'interno dell'ottava, ad esempio la nota C2 è la nota C della seconda ottava. Difatti note si dicono aventi lo stesso chroma se queste sono separate da ottave, in particolare un cromagramma indica quanta energia di ciascuna delle 12 classi di "chroma", ovvero le classi composte da note con lo stesso chroma: {tutte le C, tutte le C#, tutte le D, tutte le D#, tutte le E, ..., tutte le B} sia presente nel segnale, frame per frame. Tale array di 12 classi prende il nome di "chroma vector", ed il chromagramma è composto dai vari chroma vector frame per frame.
L'idea è alquanto semplice, in quanto basta definire dei filtri passa-banda opportunamente calcolati, in quanto ciascuna nota dello stesso chroma ha il doppio della frequenza della nota dell'ottava precedente. Dunque, il cromagramma viene quindi calcolato a partire da uno spettogramma tramite l'applicazione di questi filtri lungo i vari frame, aggregando valori di note diverse nella classe a cui tali note appartengono.

- Pitch: Il pitch ha una definizione ambigua in letteratura. In questo progetto noi consideriamo il pitch come la frequenza fondamentale di un segnale, aderendo alla definizione univocamente condivisa. Oltre a stimare il pitch del segnale, abbiamo anche provato ad utilizzare il pitch tuning offset, il quale indica quanto un suono sia vicino ad una nota pura.

4 Simulazione e Esperimenti

4.1 Dataset

Il dataset utilizzato è il Berlin Database for Emotional Speech [1]. Tale database contiene 535 brevi frasi in tedesco pronunciate da 5 uomini e 5 donne, tali sentenze sono registrate appositamente (ovvero sono *acted*, non naturali). Ad ogni frase è associata una label che rappresenta l'emozione corrispondente, le possibili etichette sono, quindi: gioia, tristezza, noia, neutro, disgusto, rabbia, paura. Come possiamo vedere dall'immagine 2, il dataset non ha una distribuzione verosimile: è irreale che le persone esprimano così spesso rabbia rispetto alle altre emozioni, sarebbe stato verosimile se la moda del dataset fosse stata l'emozione Neutro.

4.2 Architettura del sistema

Il sistema è composto da 3 notebook jupyter:

1) Estrazione Delle Feature: In tale notebook si scorre l'intero dataset per estrarre le feature da ogni file.
2) Baseline: In tale notebook si eseguono K-NN, K-NN dopo PCA, SVM lineare e rbf sul dataset composto dalle feature estratte da ogni file.
3) Neural Network: In tale notebook viene eseguita la rete neurale MLP sul dataset composto dalle feature estratte da ogni file.

4.3 Dettagli implementativi

Per estrarre le feature è stata utilizzata la libreria librosa [9].

Onde evitare inconsistenze durante l'estrazione di feature locali (MFCC locale o pitch locale) tutti i segnali sono stati portati alla stessa durata, facendo del semplice *zero-padding*. In tale caso, nel finestrare, abbiamo scelto

valori molto popolari nello speech processing: 25 ms per la dimensione della finestra, e 10ms di frame stride (quindi con 15 ms di overlap).

Nel caso di feature globali non è stato necessario fare padding, e abbiamo lasciato la decisione della dimensione delle finestre alla libreria.

Per quanto riguarda la classificazione, è stata usata la cross-validazione sia per gli algoritmi baseline che per la rete neurale; in particolare è necessario sottolineare che non è stata fatta una suddivisione del dataset che mantenesse la distribuzione di quest'ultimo: questo perché come accennato nel paragrafo 4.1 la distribuzione del dataset EMODB non rappresenta una distribuzione verosimile, e poteva quindi essere scorretto fare l'assunzione contraria e fare una suddivisione che mantenesse tale distribuzione (su SKlearn si chiama cross-validation stratificata).

4.4 Simulazioni

Per verificare la validità delle feature proposte, sono stati composti diversi dataset di feature in cui sono state incluse diverse combinazioni di tali feature. Inoltre sono stati provate diverse strade che non verrano incluse nel progetto finale:

- LDA al posto di PCA, la quale ha portato dei risultati piuttosto scarsi
- Valori di K per K-NN più alti di 50, in quanto con K alti è sempre stata riscontrata un'accuracy bassa.
- Gaussian Naive-Bayes è sempre stato calcolato ad ogni esecuzione dei baseline, ma non avendo ottenuto risultati interessanti non è mai stato effettivamente considerato.

Ultima nota interessante: per provare a supperire alla bassa cardinalità del dataset (la quale poteva potenzialmente nuocere a K-NN ed a MLPNN) è stata fatta in un secondo momento data augmentation. Il training set è stato triplicato attraverso le due seguenti modalità:

1) Aggiungendo del leggero rumore bianco

$$N \sim \mathcal{N}(\mu = 0, \sigma^2 = 0.002)\,. \tag{10}$$

2) Spostando il segnale di 0.5 secondi.

Dopo diversi tentativi sui due tipi di data augmentation, non sono stati riscontrati vantaggi significativi.

4.5 La rete neurale

L'architettura è riassunta dalla figura 3. In particolare si deve evidenziare come la dimensionalità dell'input è stata estremamente variabile nel corso del progetto, dalle centinaia per le feature globali fino alle decine di migliaia per le feature locali; in particolare ad avere ottenuto le migliori performance è stata l'utilizzo dei soli Pitch e MFCC (a 50 Coefficienti) calcolati globalmente, con una dimensionalità di 155 (Pitch e MFCC con 50 coefficienti, globali). Per quanto concerne l'output, come indica l'immagine, indica la distribuzione di probabilità delle etichette sul particolare input. Chiaramente per fare classificazione si sceglie il valore massimo di tale array. È stata utilizzata una funzione di drop-off tra l'hidden layer e l'output con lo scopo di ridurre l'overfitting.

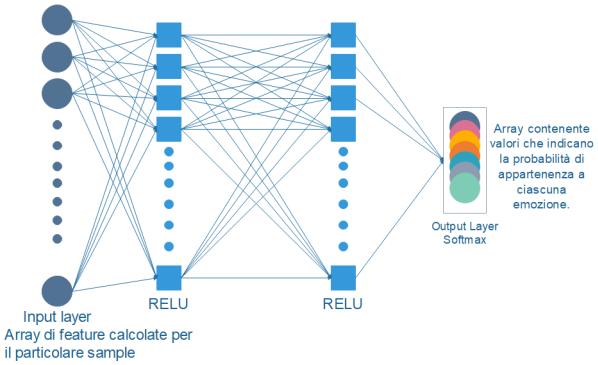

Fig. 3: La rete neurale multi-layer perceptron

Sono state eseguite numerosissime run (con un numero di epoche variabile tra 5000, 10000 e 20000) per fare il tuning dei parametri della rete, includendo tra i parametri:

- I layer ed il numero di neuroni ad ogni layer, trovando con due layer interni l'ottimalità, e con una cardinalità di tali layer tra 125 e 160 neuroni per entrambi.
- Le funzioni di attivazione dei neuroni componenti i layer, alla fine si è rivelata quasi sempre migliore la coppia RELU, RELU.
- Il valore di drop-off per il calcolo dell'output layer, che è stato preso come 0.5, seguendo la letteratura, ma è stato testato per vedere se effettivamente offrisse un vantaggio usarlo o meno (e con quale valore);
- Il batch-size, che è stato alla fine scelto tra i più piccoli divisori del trainsize, ovvero 13.
- Consultando la letteratura è stato scelto Adam come metodo di discesa del gradiente (la funzione di costo è la cross-entropia, come da standard), anche se è stata fatta una piccola prova per verificare che questo desse effettivamente le performance migliori rispetto agli altri, fissati tutti gli altri parametri.

5 Risultati ottenuti

5.1 Baseline

Come detto precedentemente, tutti i risultati corrispondono all'accuracy misurata mediante cross-validation a 10 folds. Come possiamo vedere dalla tabella 1, MFCC sono di fondamentale importanza in questo ambito, riuscendo da sole (se usate globalmente, tramite valore medio, massimo e deviazione standard di ciascun coefficiente lungo tutto il segnale) a garantire una cross validation rate del 81,4%, se poi combinati con il pitch stimato dell'intero segnale (in particolare il valore massimo, il valore medio e la deviazione standard) si ottiene un rate del 82,8%. Usando anche tutte le altre feature combinate

TABLE 1: Risultati più interessanti dei baseline

	K-NN	K-NN post PCA	SVM con kernel lineare	SVM con kernel RBF
MFCC globale con 13 coefficienti	0.564	0.685	0.697	0.736
MFCC globale con 50 coefficienti	0.635	0.723	0.814	0.786
MFCC locale con 13 coefficienti	0.443	0.455	0.668	0.566
MFCC&Pitch globali	0.625	0.708	0.828	0.792
Tutto senza MFCC	0.558	0.540	0.704	0.676
Tutto senza MFCC e pitch	0.551	0.515	0.685	0.656
Tutte feature globali	0.641	0.706	0.835	0.805

	Accuracy	Algoritmo
Chromagram	0.411	SVM rbf
Spectral Centroid	0.365	SVM lineare
Mel-scaled power spectogram	0.558	SVM rbf
Spectral Contrast	0.588	SVM lineare
zero-c,flatness,magnitude,rms energy	0.546	SVM lineare

TABLE 2: Altre features, in cui si mostra la miglior performance ottenuta con ciascuna di tali feature, e l'algoritmo con cui è stata ottenuta

si ha un ulteriore piccolo incremento, ottenendo ovvero un 83,5%.

Per quanto riguarda tutte le altre feature, i risultati di quest'ultime, utilizzate globalmente, sono mostrati nella tabella 2.

Per confrontare il risultato ottenuto con la letteratura:

- In [8] ottengono, tramite SVM, una performance del 95,1%, tuttavia lavorando soltanto su 3 emozioni: gioia, neutro e tristezza. Su tale sottoinsieme del dataset noi otteniamo una performance del 99,5%.
- In [7] ottengono un'accuracy di 82,5%, lavorando sultanto su 5 emozioni: disgusto, noia, tristezza, neutro e gioia. Su tale sottoinsieme del dataset noi otteniamo una performance del 89,6%.

È possibile, usando la matrice di confusione in figura 4, giudicare quanto sia efficace tale classificatore per le singole emozioni. In particolare si nota come questi abbia qualche difficoltà a distinguere tra Paura e Disgusto, e questo può avere senso anche in generale, in quanto effettivamente l'espressione della paura, essendo un'emozione estremamente eterogenea (si pensi appunto alle varie fobie, come l'etomofobia), può talvolta risultare simile al disgusto. Allo stesso modo il classificatore fatica a distinguere tra noia e tristezza, in quanto sono entrambe emozioni con una bassa attivazione ed hanno effettivamente un'espressione simile. Mentre risulta molto meno giustificabile la confusione tra Noia e Gioia.

5.2 Rete Neurale

Anche la rete è stata valutata mediante cross-validation a 10 folds. Il risultato migliore è di 89,3%, realizzato usando come feature MFCC a 50 coefficienti (deviazione standard, valore massimo, valore medio) e Pitch (valore medio, minimo, deviazione standard e valore massimo), l'architettura è la stessa indicata in 4.5, in particolare il numero di neuroni nei due layer è pari a 150. Tale risultato si mostra in ottima tendenza nei confronti della

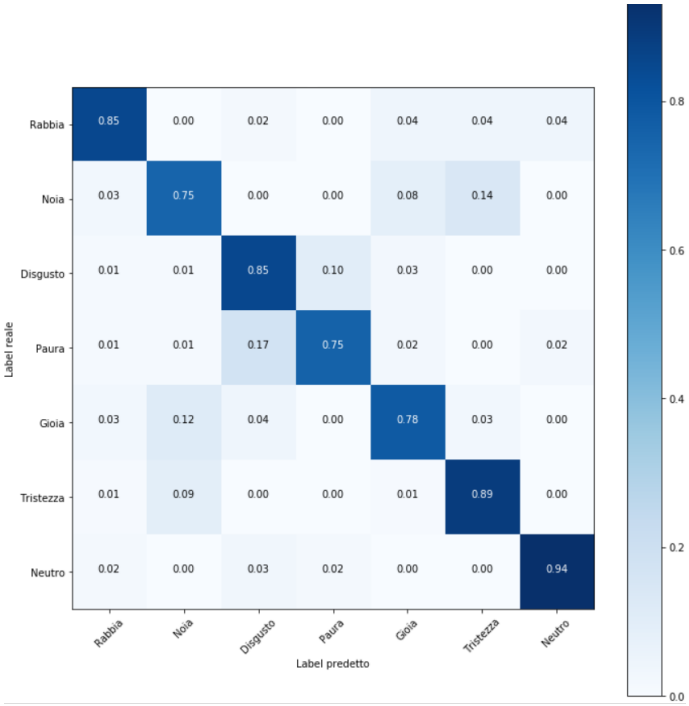

Fig. 4: Matrice di confusione del miglior SVM

letteratura, ad esempio [6] ottiene 89,62% di accuracy, che tuttavia non è stimata con cross-validation.

6 Commenti conclusivi

I risultati finali mostrano quanto siano importanti gli MFCC per la speech emotion recognition. In particolare abbiamo evidenziato come tali coefficienti siano più efficaci quando usati globalmente (attraverso media, deviazione standard e valore massimo per ciascun coefficiente lungo tutto il segnale), specialmente se combinati con il pitch. Le altre feature si sono dimostrate decisamente meno efficaci.

6.1 Criticità dell'approccio e problemi aperti

Il fatto che il database fosse *acted* potrebbe inoltre aver introdotto qualche bias, sia per la possibilità di innaturalezza o forzatezza di qualche espressione emotiva, sia perchè potrebbe essere interessante vedere se è possibile dire qualcosa anche nelle comunicazioni in cui si cerca di dissimulare o nascondere lo stato emotivo (chiaramente già sapendo di perdere molta accuratezza in questo).

Infine, anche avendo provato a rendere tutto il più possibile language independent, va sottolineato come si sia per ora lavorato soltanto con il database in linguaggio tedesco, pertanto particolari accenti o intonazioni tipici della lingua potrebbero aver introdotto del rumore che potrebbe essere scoperto con semplicità usando contemporaneamente altri dataset con altre lingue. Inoltre potrebbe anche essere interessante includere la semantica del parlato nella speech emotion recognition onde tentare a migliorare l'accuratezza dei classificatore, questa idea chiaramente renderebbe il classificatore language-dependent, e potrebbe richiedere di riconoscere la lingua

References

[1] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, "A database of german emotional speech," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[3] M. W. Bhatti, Y. Wang, and L. Guan, "A neural network approach for human emotion recognition in speech," in *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, vol. 2. IEEE, 2004, pp. II–181.

[4] N. R. Kanth and S. Saraswathi, "A survey on speech emotion recognition."

[5] M. El Ayadi, M. S. Kamel, and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition*, vol. 44, no. 3, pp. 572–587, 2011.

[6] K. Khanchandani and M. A. Hussain, "Emotion recognition using multilayer perceptron and generalized feed forward neural network," 2009.

[7] P. Shen, Z. Changjun, and X. Chen, "Automatic speech emotion recognition using support vector machine," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, vol. 2. IEEE, 2011, pp. 621–625.

[8] Y. Pan, P. Shen, and L. Shen, "Speech emotion recognition using support vector machine," *International Journal of Smart Home*, vol. 6, no. 2, pp. 101–108, 2012.

[9] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.