Progetto Modelli Probabilistici per le Decisioni

- Relazione finale -

Lorenzo Mammana 807391
Eric Nisoli 807147

Università degli Studi di Milano-Bicocca
2018/2019

Indice

Capitolo 1

Introduzione

Almeno una volta nella vita, durante la fase di studi, un musicista dovrebbe avere eseguito l'armonizzazione di un corale. Questo compito consiste nel, data una certa melodia, creare tre melodie sottostanti che suonate assieme siano piacevoli all'ascolto.
Ciò viene richiesto ai musicisti in quanto è abbastanza aperto per valutare le capacità dello studente, ma allo stesso tempo data la presenza di strutture piuttosto rigorose, è vincolato abbastanza da non richiedere una composizione libera.
L'obiettivo del progetto è quello di verificare se questo compito è fattibile tramite algoritmi di apprendimento automatico e se i risultati prodotti siano o meno confondibili con risultati reali.
Numerose tecniche sono state presentate in letterature per svolgere questo compito, alcuni approcci basati su reti neurali, altri basati su algoritmi genetici, ma la maggior parte basati su Hidden Markov Models (HMM).
Il nostro lavoro è completamente basato sul paper [citare paper] che descrive, oltre ad un HMM in grado di eseguire il task di armonizzazione, anche un secondo HMM in grado di eseguire il compito di ornamentazione, ovvero l'aggiunta di note ininfluenti per la melodia, ma in grado di aggiungere quella fantasia compositiva in più che ci si aspetta da un musicista reale.

Capitolo 2

Operazioni preliminari

Disclaimer

Gli autori non sono musicisti, ne hanno mai studiato musica!
La nostra conoscenza in merito è stata costruita nel periodo del progetto e come tale potrà contenere inesattezze di ogni sorta.

Dataset

Il dataset utilizzato è composto da 382 corali composti da Johann Sebastian Bach in formato testuale come mostrato nel Listing 2.1. Dall'header estraiamo alcune informazioni importanti come:

- Tonart: rappresenta la tonalità con cui è composto il corale (dur = Maggiore, moll = Minore)
- Takt: rappresenta la time signature del corale
- Tempo: rappresenta la velocità del corale

In particolare la notazione utilizzata per descrivere la tonalità è quella utilizzata principalmente nelle zone di influenza germanica, la conversione ad un sistema a noi più familiare è immediata

A	H	C	D	E	F	G
Do	Re	Mi	Fa	Sol	La	Si

La composizione vera e propria si articola su diverse colonne:

- Il numero contenuto nella prima colonna rappresenta l'inizio di una frase (unità musicale che ha senso anche se suonata singolarmente)

```
Choralname = bch010
Anzahl Stimmen = 4
Tonart = E-moll
Talt = 4/4
Tempo = 100
Notentextausgabe in 16tel-Schritten

PHRASE    TAKT    SOPRAN    ALT    TENOR    BASS    HARMONIK

1    1    E 1 H 0 G 0 E 0   t

                    F#0

          E 1 H 0 G 0 G 0   t3

                    F#0

          G 1 E 1 H 0 E 0   t

                    D 0

          G 1 E 1 H 0 C 0   t5

                    H -1

     2    A 1 E 1 C 1 A -1         s
```

Listing 2.1: Struttura musicale del corale 10

```
186 26
108 26
131 23
108 26
484 17
85 20
2 26
6 16
```

Listing 2.2: Esempio di file processato

- Il numero contenuto nella seconda colonna rappresenta l'inizio di una nuova battuta
- La terza colonna rappresenta la nota eseguita dal soprano
- La quarta colonna rappresenta la nota eseguita dal contralto
- La quinta colonna rappresenta la nota eseguita dal tenore
- La sesta colonna rappresenta la nota eseguita dal basso
- L'ultima colonna ci da un'indicazione riguardo la funzione armonica di una certa battuta.

Il dataset viene inizialmente suddiviso in corali in tonalità maggiore e corali in tonalità minore in quanto la loro composizione è nettamente diversa.

Preprocessing

Il dataset così definito è ovviamente impossibile da utilizzare per addestrare un HMM, è necessario, per ogni file, costruire una coppia [stato nascosto-stato visibile] che sia in grado di rappresentarne la struttura musicale.
Questa operazione viene effettuata tramite lo script hmm-data.py, che legge i corali e costruisce l'input per il modello di Markov. In particolare gli stati visibili rappresentano le note relative al soprano, mentre gli stati nascosti rappresentano le corde e la loro relativa funzione armonica.
Per ogni corale in ingresso viene prodotto un file f con lo stesso nome contenente una sequenza $S_f(h, v)$ di coppie di indici come mostrato nel Listing 2.2, a cui fanno riferimento rispettivamente gli stati nascosti e gli stati visibili contenuti nei file "SYMBOLS_HIDDEN" e "SYMBOLS_VISIBLE". Come vediamo nel Listing 2.3 (a) uno stato nascosto è nella forma a:b:c:0/Y, dove a rappresenta il numero di semitoni in più del soprano rispetto al basso, in maniera equivalente b e c rappresentano il numero di semitoni in più di

contralto e tenore rispetto al basso.
Nel Listing 2.3 (b) vediamo invece come gli stati visibili siano semplicemente le note della melodia del soprano.

(a)

```
(unknown)
28:19:12:0/T
16:7:0:0/T
14:6:-3:0/T
17:8:0:0/Sp5
21:12:3:0/%
24:16:7:0/S
-24:-16:-7:-0/S
28:19:12:0/SS
17:8:1:0/Sp5
18:8:3:0/DP3
15:7:3:0/Tp
18:8:3:0/DD3_7
16:12:7:0/D
24:19:16:0/T
```

(b)

```
(unknown)
E_2
C_2
D_2
F_2
-F_2
H_1
-C_2
C#2
-D_2
END
C#1
D#1
H_0
B 0
```

Listing 2.3: (a) Inizio del file SYMBOLS_HIDDEN (b) Inizio del file SYMBOLS_VISIBLE

Capitolo 3

Chords HMM

Identifichiamo la sequenza di note della melodia con Y e l'armonizzazione sottostante con C, in particolare y_t rappresenta la nota della melodia al tempo t e c_t rappresenta lo stato armonico al tempo t.
E' stato utilizzato un modello di Markov con assunzioni del primo ordine, ovvero tale per cui vale:

$$P(c_t|c_{t-1}, ..., c_0) = P(c_t|c_{t-1}) \tag{3.1}$$

$$P(y_t|c_t, ..., c_0, y_{t-1}, ..., y_0) = P(y_t|c_t) \tag{3.2}$$

In particolare vengono costruiti due modelli di Markov, uno relativo ai corali in tonalità maggiore ed uno relativo ai corali in tonalità minore. Il dataset iniziale è stato così suddiviso:

- 121 file di training (tonalità maggiore)
- 81 file di test (tonalità maggiore)
- 108 file di training (tonalità minore)
- 72 file di test (tonalità minore)

Fitness function

L'accuratezza dei risultati generati dal modello è stata calcolata tramite la funzione di fitness $\phi : F \mapsto [0, 1]$ descritta nell'Equazione 3.3 e, più il valore di $\phi(F)$ si avvicina a 1, maggiore sarà l'accuratezza del modello.

$$\phi(F) = \frac{\sum_{f \in F} H(f)}{\sum_{f \in F} |S_f|} \tag{3.3}$$

- F è un insieme contenente i nomi dei file da valutare.

- S_f è la sequenza di coppie (h, v) corrispondente al file f; in particolare sono indicate con $S_{f,original}$ e $S_{f,generated}$ la sequenza originale e la sequenza generata dal modello per il medesimo f, rispettivamente.
- $|S_f| = |S_{f,original}| = |S_{f,generated}|$ è la lunghezza della sequenza corrispondente al file f, la quale risulta essere uguale per entrambe le sequenze $S_{f,original}$ e $S_{f,generated}$.
- $H(f)$ è il numero di stati nascosti h_i equivalenti tra le sequenze $S_{f,original}$ e $S_{f,generated}$, con $0 \leqslant i < |S_f|$.

Training

Il modello di Markov è stato costruito utilizzando la libreria Python "hmmlearn".
Avendo a disposizione sia gli stati visibili che gli stati nascosti non è stato necessario apprendere i parametri del modello tramite Expected Maximization, ma sono state invece costruite, utilizzando i dati, sia la matrice di transizione che la matrice di emissione.
In particolare per gestire il grande numero di zero all'interno di queste matrici si è proceduto applicando uno smoothing additivo sommando 0.01 ad ogni elemento delle matrici prima di normalizzarle per renderle stocastiche. Anche la distribuzione iniziale di probabilità è stata calcolata utilizzando i dati, andando a dare una probabilità maggiore agli stati più frequenti all'inizio dei corali.
Il modello relativo alla tonalità maggiore (chords-dur) contiene 2815 stati nascosti e 55 stati visibili, mentre il modello relativo alla tonalità minore (chords-moll) contiene 2593 stati nascosti e 52 stati visibili.

Testing

Per ogni file di test viene generata, data la relativa sequenza di stati visibili, la sequenza più probabile di stati nascosti utilizzando l'algoritmo di Viterbi. La Tabella 3.1 mostra quanto gli stati nascosti prodotti siano equivalenti a quelli reali attraverso la fitness function.

Model	$\phi(F)$
chords-dur	0.21114
chords-moll	0.22193

Tabella 3.1: Valori della funzione di fitness phi per i modelli chords-dur e chords-moll

Come ci si aspetta i risultati sono abbastanza diversi in quanto il modello non ha alcuna informazione relativa ad esempio al periodo in cui è stato composto un determi-

nato corale, oppure per quale occasione. Senza l'ausilio di ulteriori dati esterni sarebbe impossibile ricostruire perfettamente i corali originali.

(a)

```
186 26
108 26
131 23
108 26
484 17
85 20
2 26
6 16
1 26
485 20
11 17
256 17
486 29
262 26
116 26
```

(b)

```
108 26
108 26
131 23
108 26
938 17
132 20
108 26
170 16
108 26
171 20
14 17
14 17
15 29
108 26
181 26
```

Listing 3.1: (a) Sequenza originale di stati (b) Armonizzazione più probabile generata dal nostro modello

Interpretare i risultati non è semplice, il Listato 3.1 non è per nulla esplicativo, se ragionassimo in termini di accuratezza sembrerebbe che il nostro modello sia assolutamente pessimo, ma l'obiettivo non è quello di costruire risultati identici all'originale, ma armonizzazioni musicalmente accettabili e orecchiabili.
Presentare i risultati sotto forma di spartito musicale può favorirne la comprensione per chi è in grado di leggerli, ma non è abbastanza per chi non ha una approfondita conoscenza del dominio. Per questo motivo è necessario trasformare i risultati ottenuti in un formato musicale udibile.

Sampling

E' possibile anche generare sequenze di stati in accordo alla distribuzione di probabilità del modello. Consideriamo $\alpha_{t-1}(j)$, la probabilità di avere visto le prime $t-1$ osservazioni di una sequenza ed essere finiti nello stato j, possiamo calcolare la probabilità di aver visto i primi $t-1$ eventi, essere finiti in uno stato qualsiasi, e poi eseguire una transizione verso uno stato k:

$$\begin{aligned} P(y_0 = Y_{i0}, y_1 = Y_{i1}, ..., y_{t-1} = Y_{i_{t-1}}, s_{t-1} = S_j, s_t = S_k) \\ = \alpha_{t-1}(j) P(s_t = S_k | s_{t-1} = S_j) \end{aligned} \tag{3.4}$$

Possiamo usare questa equazione per calcolare $\rho_t(j|k)$ la probabilità di essere in uno stato S_j al tempo $t-1$ data la sequenza di eventi osservati $Y_{i0}, Y_{i1}, ..., Y_{it-1}$ e dato che saremo nello stato S_k al tempo t:

$$\begin{aligned}\rho_t(j|k) &= P(s_{t-1} = S_j | y_0 = Y_{i0}, y_1 = Y_{i1}, ..., y_{t-1} = Y_{i_{t-1}}, s_t = S_k) \\ &= \frac{\alpha_{t-1}(j) P(s_t = S_k | s_{t-1} = S_j)}{\sum_l P(s_t = S_k | s_{t-1} = S_l)}\end{aligned} \tag{3.5}$$

Per instanziare una sequenza di stati $s0 = S_{v0}, s1 = S_{v1}, ..., s_T = s_{vT}$ viene inizialmente scelto lo stato finale utilizzando la distribuzione di probabilità del modello:

$$P(s_T = S_j | y_0 = Y_{i0}, y_1 = Y_{i1}, ..., y_T = Y_{iT}) = \frac{\alpha_T(j)}{\sum_l \alpha_T(l)} \tag{3.6}$$

Una volta scelto v_T tale per cui lo stato finale risulti essere $s_t = S_{vt}$ possiamo usare le variabili $\rho_t(j|k)$ per muoverci all'indietro sulla sequenza:

$$P(s_T = S_j | y_0 = Y_{i0}, y_1 = Y_{i1}, ..., y_T = Y_{iT}, s_{t+1} = S_{v_{t+1}}) = \rho_{t+1}(j|v_{t+1}) \tag{3.7}$$

Ricostruzione dei risultati

Mediante l'utilizzo dello script "hmm-output-expand.py" è possibile ricostruire, a partire dalle coppie [hidden-visible] costruite tramite Sampling o Viterbi, il file in notazione musicale.
In questo punto notiamo uno dei principali problemi del modello costruito, ovvero la mancanza di informazione temporale relativa agli stati visibili e nascosti.
Per ricostruire il file musicale è necessario rispettare le cadenze della musica originale e ciò rende molto complicato ad esempio generare nuova musica con il modello. Questo banalmente perchè la musica generata, per quanto orecchiabile, non avrà una struttura temporale sensata ed ogni nota avrà la stessa lunghezza, come tale la melodia prodotta risulterà molto "robotica" e poco reale.

Generazione del file MIDI

Il protocollo MIDI è uno standard per la composizione e riproduzione di file musicali. Tramite lo script "chorale2midi.py" è possibile convertire i file testuali generati nel passaggio precedente in formato MIDI in modo tale da poterli riprodurre.

Armonizzazione di musica moderna

Il modello di Markov proposto è teoricamente in grado di armonizzare una melodia qualsiasi, dato un file MIDI è possibile estrarre le note della melodia e utilizzarle come

base per l'armonizzazione. Il task è però particolarmente complicato perchè la struttura dei file di input è molto rigida, in particolare:

- la durata delle note, nel file MIDI, deve necessariamente essere multiplo della durata di una nota semiminima (1/4)
- in una battuta ci possono essere massimo quattro note
- non c'è mai una pausa tra note nella stessa battuta

Il primo motivo è facilmente gestibile andando ad arrotondare la durata delle note al multiplo più vicino, ciò altera leggermente la melodia, ma non abbastanza da rovinarla.
Il secondo è un problema legato principalmente a canzoni veloci o complesse che non sono gestibili dal modello.
L'ultimo problema è quello forse più complicato da gestire, in quanto richiede l'aggiunta di note fittizie all'interno del file che verranno poi trattate come delle pause.
In Python non sono presenti librerie in grado di gestire facilmente file MIDI in ingresso, per questo motivo la musica moderna è stata trascritta manualmente nei file testual e, per mancanza di tempo non è stato possibile effettuare automaticamente questo compito. A questo si aggiunge il fatto che, molto spesso, i file MIDI non sono correttamente divisi per canali (idealmente uno strumento per canale) e ciò rende ancora più complicato il parsing automatico di questi file.
A puro scopo dimostrativo e ludico abbiamo provato ad armonizzare varie melodie moderne e non per vedere come suonerebbero se fossero state composte da Bach sotto forma di corale. Nella maggior parte dei casi il suono è troppo cacofonico, principalmente quando la canzone è molto veloce o presenta una melodia con molte note per battuta.

Capitolo 4

Ornamentation HMM

Il primo modello di Markov aggiunge una sola nota per battuta per le linee melodiche di contralto, tenore e basso, questa è una semplificazione rispetto ai corali reali di Bach che possono invece presentare quattro note per battuta come accade per la linea melodica del soprano. Questo problema viene risolto addestrando un secondo modello di Markov nascosto in cui gli stati visibili rappresentano quanto le tre linee musicali crescono o decrescono tra una battuta ed un altra, mentre gli stati nascosti rappresentano di quanto dovrebbe essere alzato o abbassato uno degli ultimi tre quarti della battuta.

(a)

```
0,0,0,0/0,0,0,0/0,0,0,0
0,0,0,0/0,0,2,2/0,0,0,0
0,0,0,0/0,0,5,5/0,0,0,0
0,2,4,0/0,0,0,0/0,0,0,0
0,0,-5,-5/0,0,0,0/0,0,0,0
0,0,0,0/0,0,-3,-3/0,0,0,0
0,0,0,0/0,0,0,0/0,0,-1,-1
0,0,0,0/0,0,1,1/0,0,0,0
0,0,0,0/0,0,-1,-1/0,0,0,0
0,0,-2,-2/0,0,0,0/0,0,0,0
0,0,-1,-1/0,0,0,0/0,0,0,0
0,0,5,5/0,0,0,0/0,0,0,0
0,0,-1,-1/0,0,-2,-2/0,0,0,0
0,0,2,2/0,0,1,1/0,0,0,0
0,0,-1,-1/0,0,-1,-1/0,0,2,2
```

(b)

```
0/0/0/12
0/-3/-5/-2
0/1/2/-1
0/2/1/-2
0/2/2/-2
0/0/0/0
0/-4/-2/-7
0/0/0/11
0/-1/1/-1
0/0/1/1
0/-2/-3/-3
0/5/5/1
0/0/2/-7
0/0/-5/7
0/-5/-2/-1
```

Listing 4.1: (a) Stati nascosti (b) Stati visibili

Ad esempio facendo riferimento all'ultima riga del Listato 4.1 vediamo che per una battuta in cui, nella battuta successiva, il contralto scende di 5 semitoni, il tenore scende di 2 semitoni e il basso scende di due semitoni andrà a generare le tre linee melodiche in cui la terza e la quarta nota sono rispettivamente abbassate di un semitono per contralto e tenore, ed alzate di due semitoni per il basso.

Training

Anche in questo caso prima del training viene effettuata una parte di preprocessing per ottenere gli stati visibili e nascosti, il training del modello viene eseguito esattamente come descritto nel capitolo precedente.

Testing

Anche in questo caso si opera nella stessa identica maniera descritta precedentemente, i risultati in termini di accuratezza vengono mostrati in tabella.

Ricostruzione dei risultati

Per ricostruire i risultati è stato necessario aggiungere uno script non presente nel repository originale in grado modificare il file musicale costruito dal modello di Markov precedente in accordo con la modifica di semitono delle linee melodiche.

Generazione del file MIDI

Una volta integrati i risultati dei due modelli è possibile costruire i file MIDI utilizzando lo stesso script relativo al modello precedente.

Ornamentazione di musica moderna

Non è invece possibile eseguire il task di ornamentazione della musica moderna, questo è dovuto al fatto che i file costruiti da noi contengono esclusivamente informazione relativa alla melodia del soprano e come tale non è possibile costruire gli stati visibili e nascosti richiesti dal modello per funzionare.

Capitolo 5

Valutazione risultati

Non avendo conoscenze teoriche in merito alla musica la valutazione dei risultati è stata effettuata semplicemente andando a valutare la gradevolezza uditiva delle melodie composte dal modello. In particolare sembra essere molto buona l'armonizzazione prodotta dal primo modello di Markov sui vari corali testati, questo soprattutto perchè il compito non è particolarmente complicato e molto spesso suonare più tasti del pianoforte assieme per lo stesso tempo produce suoni gradevoli.
I file prodotti tramite sampling basato su melodie note produce risultati simili a quelli di Viterbi, ma effettivamente peggiori all'ascolto.
Anche il test eseguito sulla musica moderna produce risultati interessanti, ovviamente la musica prodotta difficilmente potrebbe essere utilizzata realmente, ma soprattutto su parti lente e più vicine ai ritmi di un corale il modello armonizza bene e i risultati sono piuttosto gradevoli all'ascolto.
Il modello che esegue l'ornamentazione non ci convince invece particolarmente, molto spesso l'armonizzazione prodotta non è particolarmente gradevole, soprattutto se paragonata ai risultati prodotti dal modello precedente, in particolare abbiamo riscontrato una non riproducibilità dei risultati mostrati dall'autore dell'articolo. I suoi risultati sono eccezionali all'ascolto e sembrano veramente troppo perfetti per essere stati prodotti dal modello, è possibile che vi sia stata una manipolazione umana per rendere più interessante la proposta dell'articolo che, in ogni caso, presenta comunque un ottimo modello!

Capitolo 6

Conclusioni

L'obiettivo ultimo del nostro lavoro era quello di rendere disponibile un modello di Markov per l'armonizzazione di corali funzionante e che non richiedesse di operare su file scritti in PERL, linguaggio molto popolare fino agli anni 2000, ma ormai in lento declino. L'obiettivo è stato pienamente raggiunto, l'intero codice prodotto è scritto in Python ed è perfettamente utilizzabile come base per estendere il modello proposto con ulteriori idee e miglioramenti.
Il modello proposto da Allan si è rivelato funzionante, anche se in maniera inferiore a quanto mostrato dall'autore negli esempi che ha reso disponibili pubblicamente; il modello presenta però numerose lacune dovute soprattutto alla rigidità dei file di ingresso e alla scelta di non integrare alcuna informazione riguardante la durata delle note negli stati del modello. Questo causa l'impossibilità di generare automaticamente corali nello stile di Bach, da un certo punto di vista ciò è corretto, non è questo lo scopo che si è prefisso l'autore, da un altro punto di vista sarebbe stato interessante effettuare anche questo compito.
Il nostro maggior contributo oltre la traduzione è quello però di aver costruito script particolari in grado di prendere una linea melodica e costruirne l'armonizzazione, questo permette di verificare cosa succederebbe se Bach entrasse a far parte di un certo gruppo musicale. Questo ha fondamentalmente valenza ludica, il che non ne sminuisce comunque il valore, se consideriamo che il settore in questione è uno dei più importanti in ambito informatico.