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

2.1 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.

2.2 Dataset

Il dataset utilizzato è composto da 382 corali composti da Johann Sebastian Bach in formato testuale come mostrato in figura 2.1.

Figura 2.1: Struttura musicale del corale 10

```
Choralname = bch010
Anzahl Stimmen = 4
Tonart = E-moll
Takt = 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
```

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)
- 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.

2.3 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 con lo stesso nome contenente una serie di coppie di indici come mostrato in figura 2.2.

Figura 2.2: Esempio di file processato

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

Queste coppie di indici fanno riferimento rispettivamente agli stati nascosti e a quelli visibili contenuti nei file "SYMBOLS_HIDDEN" e "SYMBOLS_VISIBLE". Come vediamo dalla figura 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.
In figura 2.3 (b) vediamo invece come gli stati visibili siano semplicemente le note della melodia del soprano.

Figura 2.3: (a) Inizio del file SYMBOLS_HIDDEN, (b) Inizio del file SYMBOLS_VISIBLE

```
(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
```

(a)

```
(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
```

(b)

Capitolo 3

Hidden Markov Model

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)

3.1 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.

3.2 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. In tabella viene mostrato quanto gli stati nascosti prodotti siano equivalenti a quelli reali.
INSERIRE TABELLA RISULTATI 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 determinato corale, oppure per quale occasione. Senza l'ausilio di ulteriori dati esterni sarebbe impossibile ricostruire perfettamente i corali originali.

Figura 3.1: A sx l'originale, a dx l'armonizzazione più probabile del nostro modello

186 26	108 26
108 26	108 26
131 23	131 23
108 26	108 26
484 17	938 17
85 20	132 20
2 26	108 26
6 16	170 16
1 26	108 26
485 20	171 20
11 17	14 17
256 17	14 17
486 29	15 29
262 26	108 26
116 26	181 26

Interpretare i risultati non è semplice, la figura 3.1 non è per nulla esplicativa, 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.

3.3 Ricostruzione dei risultati

Mediante l'utilizzo dello script "hmm-output-expand.py" è possibile ricostruire, a partire dalle coppie [hidden-visible] costruite da 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 più "robotica" e poco reale.

3.4 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.

3.5 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 di 240 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 di ingresso, 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 per vedere come suonerebbero se fossero state composte da Bach sotto forma di corale.