Bach Chorales

F. Belotti, D. Samotti, G. Scarpellini

Modelli probabilistici per le decisioni

Indice

1 Introduzione

I modelli probabilistici temporali sono stati impiegati sin dalla loro origine per modellare processi stocastici musicali [1]. Nel presente lavoro, abbiamo analizzato la letteratura in materia al fine di confrontare due noti processi di Markov - Hidden Markov Models (da qui in avanti HMM) e Factorial Hidden Markov Models (da qui in avanti FHMM) - nel task di *auto-completamento* di una canzone. Abbiamo impiegato il dataset **Chorales** essendo questo uno standard ampiamente diffuso in letteratura [4, 5]. Abbiamo quindi addestrato i modelli tramite l'algoritmo di Expectation Maximization con diverse configurazioni e ne abbiamo valutato le performance in termini di likelihood. Abbiamo inoltre ritenuto interessante indagare circa l'aspetto generativo di questi modelli. A tal fine, abbiamo deciso di impiegare alcuni dei modelli addestrati per creare nuove composizioni che condividessero lo stile musicale del dataset di partenza. I risultati e le composizioni che presentiamo in 5 lasciano aperti numerosi sviluppi futuri.

2 Task

Il task proposto riguarda la predizione di una sequenza musicale a partire da una sequenza nota. La funzione principale di *auto-completamento* può, inoltre, trovare impiego nell'ambito delle indagine per plagio, laddove sia necessario verificare se due differenti composizioni musicali sono indebitamente correlate. L'interesse sempre più crescente nello letteratura verso modelli *generativi* - ossia modelli di Machine Learning capaci di *comporre* brani musicali verosimili [6, 5]- ci ha guidati nello sperimentare forme di generazione automatica di spartiti musicali che condividessero lo stile del dataset di training. La musica da noi generata (ad esempio 1) condivide pattern e stile musicale dei dati su cui i modelli di machine learning sono stati addestrati.

3 Dataset

Il dataset utilizzato in questo lavoro è composto da 100 corali (originariamente scritte per quattro voci: Soprano, Alto, Tenore e Basso) di J. S. Bach della sola voce Soprano [4].
Esso non contiene valori mancanti e gli attributi di cui ogni evento, o nota, è composto sono:

- **st**: *start time*, misurato in sedicesimi, rappresenta l'inizio della nota a partire dal tempo 0, l'inizio della corale. Il suo dominio è l'insieme $\mathbb{N}$.

- **pitch**: *pitch*, o altezza, è la frequenza fondamentale di una nota musicale o suono che viene percepita; è rappresentata mediante il cosiddetto "MIDI number" [3]. Il suo dominio è l'insieme $\{60, 61, ..., 79\} \subset \mathbb{N}$.

- **dur**: *durata*, misurata in sedicesimi, rappresenta la durata della nota. Vale inoltre che, sommando per ogni evento *start time* e *durata* dell'evento i-esimo, si ottiene lo *start time* dell'evento successivo $i+1$-esimo. Il suo dominio è l'insieme $\mathbb{N}$.

- **keysig**: *key signature*, o armatura di chiave, è l'insieme delle alterazioni poste subito dopo la chiave. Queste alterazioni (dette d'impianto o in chiave) durano per tutto il brano e sono rappresentate da tanti diesis (#) o bemolli (♭), uno per riga, quante sono le righe del pentagramma alterate. Il suo dominio è l'insieme $\{-4, ..., 4\} \subset \mathbb{Z}$, dove un numero positivo rappresenta il numero di diesis, un numero negativo il numero di bemolli e lo 0 rappresenta l'assenza di entrambi.

- **timesig**: *time signature*, o misura, è una notazione standard che rappresenta il numero di beat (ovvero l'andamento o la velocità di una composizione) in una battuta; misurato in sedicesimi il suo dominio è l'insieme $\mathbb{N}_+$.

- **fermata**: *fermata*, o punto coronato, è un segno utilizzato per aumentare il valore di una nota o di una pausa a piacimento dell'esecutore. Il suo dominio è l'insieme $\{0, 1\}$.

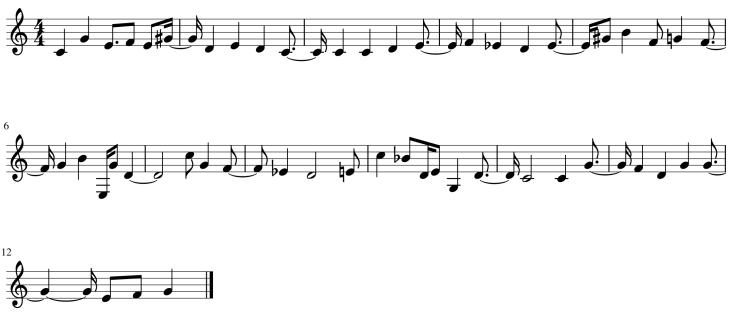

Figura 1: Spartito generato con HMM preaddestrato

Nome	Descrizione	Dominio
st	start time	$\mathbb{N}$
pitch	pitch	$\{60, ..., 79\}$
dur	durata	$\mathbb{N}$
keysig	key signature	$\{-4, ..., 4\}$
timesig	time signature	$\mathbb{N}_+$
fermata	fermata	$\{0, 1\}$

Tabella 1: Attributi del dataset **Chorales**

Il dataset è in formato LISP, di cui di seguito ne è presentato un estratto di una delle corali:

```
(1
    ((st 8) (pitch 67) (dur 4) (keysig 1) (timesig 12) (fermata 0))
    ((st 12) (pitch 67) (dur 8) (keysig 1) (timesig 12) (fermata 0))
    ...
    ((st 252) (pitch 67) (dur 8) (keysig 1) (timesig 12) (fermata 1))
)
```

Non viene specificata la semantica del numero che compare all'inizio di ogni corale, anche se presumibilmente rappresenta il numero della stessa come originariamente assegnato da J. S. Bach.
Altre due sono le informazioni assenti: la chiave in cui è scritta la corale (chiave di Do, Fa, o di Sol), simbolo posto all'inizio del pentagramma con la funzione di fissare la posizione delle note e l'altezza dei relativi suoni; e il tempo, ovvero l'andamento o velocità della composizione.

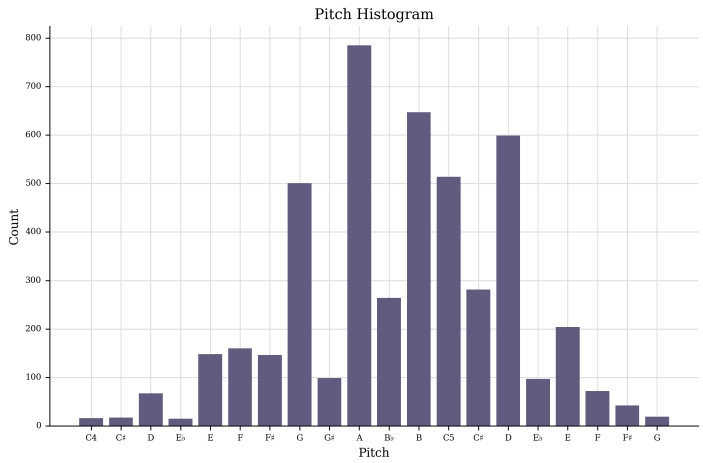

Figura 2: Distribuzione dei valori per l'attributo *pitch* nel dataset **Chorales**

3.1 music21

music21 è un toolkit reso disponibile dal MIT scritto in Python per la cosiddetta "computer-aided musicology", o musicologia assistita.
Il toolkit permette di trovare momenti interessanti e avere un'idea del profilo generale di un brano o di un repertorio di brani; mette inoltre a disposizione un elevato numero di API di ricerca, analisi, creazione, rappresentazione grafica e di conversione musicale molto ricche e ben documentate [2].
music21 è stato utilizzato in questo lavoro principalmente per, ma non solo:

- ottenere una rappresentazione del dataset **Chorales** "musicalmente accettbile", ovvero che rispettasse lo standard musicale vigente
- effettuare analisi musicali precise, come ad esempio ottenere la tonalità di una composizione
- reperire ulteriori dataset, sia di Bach che di altri compositori, da utilizzare durante la fase di testing dei modelli addestrati
- normalizzare i dataset a disposizione, ad esempio traslando tutte le altezze da una tonalità ad un'altra (ad esempio in C maggiore)

I dataset recuperati mediante musi21 sono riassunti nella tabella 2

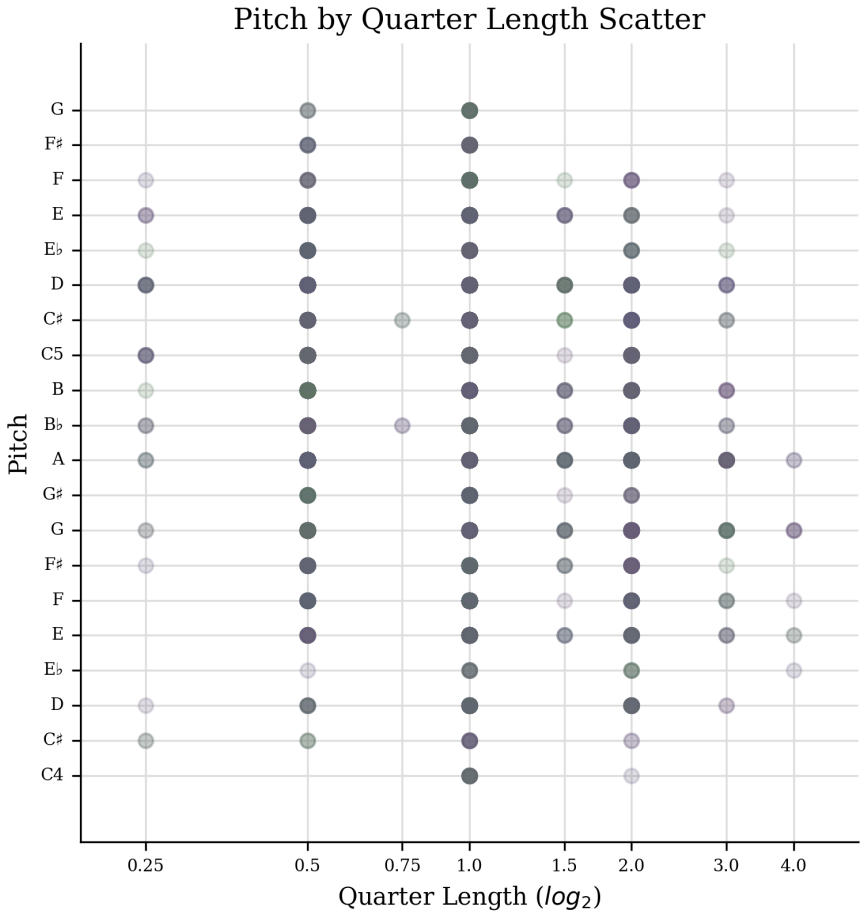

Figura 3: Distribuzione dei valori di *pitch* rispetto alla *durata*, espressa in quarti

3.2 Parsing

Ad ogni corale del dataset **Chorales** è stata applicata una trasformazione che ne converte ogni evento in un oggetto di tipo `music21.note.Note` (oggetto del toolkit music21 che rappresenta una nota e tutte le informazioni ad essa associate) e ogni corale in un oggetto di tipo `musci21.stream` (oggetto del toolkit music21 che rappresenta una composizione in ogni sua forma) in

Dataset	Descrizione
Bach	362 corali della sola voce Soprano, utilizzato per testare la verosimiglianza con **Chorales**
Mozart	11 brani della sola parte per viola, utilizzato per testare la verosimiglianza con **Chorales**
Beethoven	20 brani della sola parte per viola, utilizzato per testare la verosimiglianza con **Chorales**

Tabella 2: Dataset reperiti con music21

questo modo:

- viene ignorato il numero all'inizio di ogni corale, che potrebbe rappresentare il numero della corale, in quanto non rilevante
- l'attributo *start time* viene ignorato, e si considera l'inizio della corale al tempo 0; le note vengono dunque separate temporalmente dall'attributo *durata*
- l'attributo *durata* viene convertito da sedicesimi in quarti dividendone il valore per 4
- l'attributo *time signature* viene convertito da sedicesimi in quarti dividendone il valore per 4, e viene considerato come il numeratore della misura (il denominatore sarà sempre uguale a 4)
- gli attributi *pitch*, *keysig* e *fermata* vengono mantenuti così come si presentano

Recuperati gli attributi di un evento viene creato un oggetto di tipo `music21.note`, e alla fine di ogni corale, con le note, la time signature e la key signature ottenute, viene creato un oggetto di tipo `music21.stream`.
Un esempio di trasformazione è la seguente:

```
((st 8)
(pitch 67)              <music21.note.Note G>
(dur 4)                 <music21.note.Note G duration 1>
(keysig 1)              <music21.key.KeySignature of 1 sharp>
(timesig 12)            <music21.meter.TimeSignature 3/4>
(fermata 0)             <music21.note.Note G fermata 0>
```

Non avendo a disposizione la chiave in cui è stata scritta la corale, viene utilizzata di default la chiave di Sol; il calcolo del tempo per ogni nota e la divisione in battute che rispetti la *time signature* vengono effettuati automaticamente da musci21.
Gli attributi che per il solo dataset **Chorales** vengono trasformati, vengono invece recuperati per tutti gli altri dataset (tabella 2).

3.3 Normalizzazione

Com'è noto, le composizioni variano molto da artista ad artista, sono influenzate sia dalla fantasia di quest'ultimo sia dal periodo storico e sociale in cui vengono composte.
Per questo motivo si è deciso di applicare ad ogni corale o composizione di ogni dataset una "normalizzazione musicale", ovvero tale da mantenerne la melodia, ma che ne uniformasse la tonalità. La tonalità standard è solitamente quella di C maggiore, ed ogni corale o canzone è stata "traslata" dalla sua tonalità, recuperata grazie ad un analisi effettuata da music21, a quella, appunto, di C maggiore.
Il *time signature* è invece lasciato invariato in quanto avrebbe "corrotto musicalmente" l'andamento della composizione.

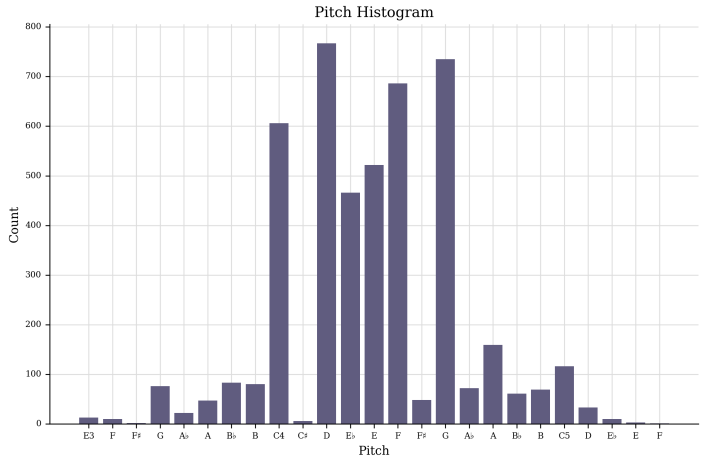

Figura 4: Distribuzione normalizzata dei valori per l'attributo *pitch* nel dataset **Chorales**

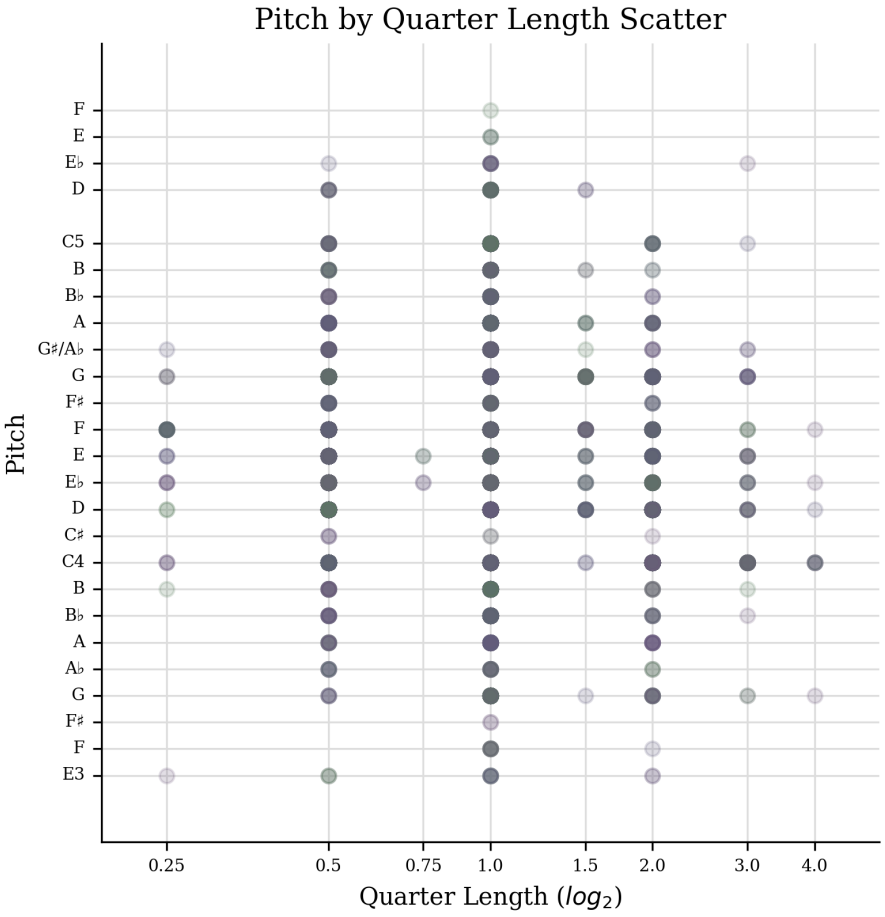

Figura 5: Distribuzione normalizzata dei valori di *pitch* rispetto alla *durata*, espressa in quarti

3.4 Spazio delle features

A fronte delle considerazioni fatte finora si è deciso di utilizzare come features la coppia di attributi $(pitch, durata)$, in quanto ritenuti molto significativi nel modellare la melodia di una composzione musicale.
Da un'analisi esplorativa dei dataset, sia **Chorales** sia quelli recuperati con music21, emerge che ci sono precisamente 33 differenti pitch e 24 durate, con

valori compresi tra $48 \leq pitch \leq 81$ e $0 \leq duration \leq 6$ rispettivamente.
È stato quindi effettuato il prodotto cartesiano tra questi due insiemi ottenendo una lista di 792 possibili combinazioni.
Dopo aver effettuato il processo di normalizzazione, risulta che ci sono 41 differenti pitch e 24 durate, con valori compresi tra $39 \leq pitch \leq 81$ e $0 \leq duration \leq 6$ rispettivamente.

4 Algoritmi e modelli

Dopo aver analizzato il dataset a nostra disposizione, abbiamo stabilito un elenco di caratteristiche e assunzioni che i nostri algoritmi di machine learning dovevano rispettare:

- Probabilistici
- Temporali, ossia in grado di evidenziare pattern in sequenze distribuite nel tempo
- A tempo discreto

Abbiamo quindi analizzato la letteratura [1, 7] interente ai task di speech and music prediction, al fine di prendere una decisione informata. La nostra scelta è ricaduta su una famiglia di modelli probabilistici temporali di natura markoviana: gli Hidden Markov Models. In particolare, abbiamo scelto di confrontare due approcci basati su HMM spesso comparati anche in letteratura [1]. Abbiamo pertanto adottato un metodo scientifico, che ha previsto l'esecuzione di alcuni esperimenti i cui risultati sono riportati in **??**. Per ciacsun modello abbiamo indagato un insieme di iperparametri chiave nel processo di apprendimento; abbiamo addestrato i modelli sul dataset che inizialmente abbiamo denominato **Chorales** e ne abbiamo valutato la bontà sui dataset, recuperati mediante music21, di **Bach**, **Mozart** e **Beethoven**.

4.1 HMM

Il modello HMM, che riportiamo in figura 6a, è un modello probabilistico indicato per l'apprendimento di pattern in una sequenza di osservazioni; è caratterizzato da un insieme di *stati nascosti*, ovvero non direttamente osservabili, con alfabeto discreto e una sequenza di *stati osservabili* direttamente.

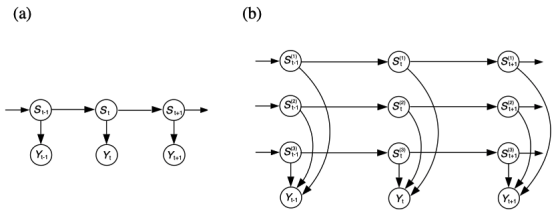

Figura 6: Confronto tra HMM (a) e FHMM (b), trattato da [1]

Nel caso in esame, abbiamo interpretato la composizione (*pitch*, *durata*) come un'osservazione, a cui è stato assegnato un numero intero (come specificato in 3.4). Abbiamo quindi adottato l'algoritmo di EM (si veda 4.3) al fine di apprendere le matrici di transazione tra gli stati nascosti del modello, nonché la loro distribuzione iniziale e la matrice di probabilità condizionale di ciascuna osservazione.
Una HMM è formalmente caratterizzata da:

- K, il numero di stati del modello. Si denotano i singoli stati con l'insieme $\{1, 2, .., K\} \subset \mathbb{N}$ e lo stato al tempo t come S_t
- L, la dimensione dell'alfabeto distinto di simboli, che corrispondono all'output del modello. Nel caso in esame i simboli sono i numeri appartenenti all'insieme $\{0, ..., 791\} \subset \mathbb{N}$ per i dataset non normalizzati, $\{0, ..., 983\} \subset \mathbb{N}$ per quelli normalizzati
- $A = \{a_{ij}\} \in \mathbb{R}^{K \times K}$, la matrice di transizione, dove

$$a_{ij} = P(S_{t+1} = j | S_t = i),\ 1 \leq i, j \leq K \tag{1}$$

- $B = \{b_j(k)\} \in \mathbb{R}^{K \times L}$, la matrice delle osservazioni, dove

$$b_j(l) = P(\text{osservazione } l \text{ al tempo } t | S_t = j), \tag{2}$$

  $l \in \{0, ..., 791\} \subset \mathbb{N}$ o $l \in \{0, ..., 983\} \subset \mathbb{N}$, a seconda che si tratti dei dataset non normalizzati oppure normalizzati, rispettivamente

- $\pi = \{\pi_i\}$, la distribuzione a priori della probabilità iniziale di ciascun stato, ossia in assenza di precedenti osservazioni, dove

$$\pi = [P(1), P(2), ..., P(K)] \tag{3}$$

  dove $1, ..., K$ sono sono gli stati assunti dal modello

Una generica HMM verrà dunque denotata come $\lambda = (A, B, \pi)$.
Il processo markoviano modellato da HMM gode delle proprietà delle catene di markov e pertanto presenta due importanti assunzioni. In particolare, si assume che la sequenza di stati nascosti sia una catena di markov di primo ordine, ossia che *ciascuno stato dipenda soltanto dallo stato precedente* (1). Si assume inoltre che ciascuna osservazione dipenda soltanto dallo stato da cui è stata *causata* e non dalle osservazioni precedenti. Pertanto, l'osservazioni tra loro sono condizionalmente indipendenti dato lo stato (2).

4.2 FHMM

Il modello HMM può essere generalizzato rappresentando il singolo stato come una collezione di variabili di stato discrete:

$$S_t = S_t^{(1)}, ..., S_t^{(m)}, ..., S_t^{(M)}, \tag{4}$$

ognuno dei quali può potenzialmente assumere $K^{(m)}$ differenti valori. Lo spazio degli stati consiste dunque del prodotto cartesiano di queste variabili di stato, e se si suppone per semplicità che $K^{(m)} = K$, allora esso è nell'ordine di K^M: in assenza di ulteriori restrizioni sulla topologia della catena, la matrice di transizione A sarebbe nell'ordine di $K^M \times K^M$.
Un tale modello, oltre ad essere intrattabile computazionalmente (l'algoritmo di EM 4.3 sarebbe esponenziale in M), sarebbe equivalente ad una HMM con K^M stati.
Una restrizione plausibile sarebbe dunque quella in cui ogni variabile di stato evolve secondo la propria dinamica, ed è a priori indipendente dalle altre variabili di stato (figura 6b), ovvero:

$$P(S_t|S_{t-1}) = \prod_{m=1}^{M} P(S_t^{(m)}|S_{t-1}^{(m)}) \tag{5}$$

Con questo tipo di restrizione, la rappresentazione della transizione tra stati può essere rappresentata mediante l'utilizzo di M distinte matrici $K \times K$.

L'implementazione di FHMM che abbiamo impiegato è una versione modificata di quella presentata in `https://github.com/regevs/factorial_hmm` [10]. Abbiamo adattato la libreria al fine di impiegare in input un'intera corale come sequenza di osservazioni. I parametri in fase di update del modello sono ottenuti come media dei parametri ricavati dalla fase di maximization per ogni osservazione (si faccia riferimento a 4.3 per l'algoritmo di Expectation-Maximization).

4.3 Expectation Maximization

L'Expectation Maximization (EM) è un algoritmo impiegato in statistica per stimare i parametri di un modello probabilistico in maniera tale da massimizzarne la likelihood.
Tale algoritmo stima i parametri di un modello in modo iterativo, a partire da alcune ipotesi iniziali; ogni iterazione consiste in due step:

1. **Expectation (E)**, che trova la distribuzione delle variabili non osservate dati i valori delle variabili osservate e la stima corrente dei parametri

2. **Maximization (M)**, che aggiorna i parametri così da ottenere la likelihood massima, supponendo che la distribuzione trovata nel passaggio precedente sia corretta.

In generale, dato un modello parametrizzato da θ (nel caso dei modelli definiti in precedenza sarà $\theta = \lambda = (A, B, \pi)$) e una sequenza di osservazioni O, l'algoritmo di EM aggiusta i parametri del modello affinchè sia massima la quantità $L(\theta) = \log P(O|\theta) = \log \sum_S P(S, O|\theta)$, dove S è l'insieme delle variabili non osservate.
Si può dimostrare che ciascuna delle iterazioni dell'algoritmo migliora la likelihood del modello o la lascia invariata, se è già stato raggiunto un massimo locale [8].

4.4 Inferenza

Per entrambi i modelli, HMM e FHMM, l'inferenza esatta risulta essere esponenziale nel numero degli stati, in particolare, data una sequenza di osservazioni lunga T, si avrebbe una complessità nell'ordine di $O(2TK^T)$ e $O(TK^{2M})$ rispettivamente per HMM e FHMM.
Tale complessità può essere notevolmente ridotta grazie all'utilizzo dei cosiddetti algortimi di Forward-Backward, riducendola a $O(TK^2)$ [9] e $O(TMK^{M+1})$ [1] rispettivamente.

5 Risultati

Entrambi i modelli, HMM e FHMM, sono stati addestrati, con diversi iper-parametri, con il dataset denominato precedentemente **Chorales**, composto da 100 corali di J. S. Bach per la sola voce soprano, opportunamente trasformato e sia normalizzato traslandone la tonalità di ogni corale in scala di C maggiore, sia in versione originale (3.1, 3.2, 3.3). Al termine dell'addestramento ogni modello è stato poi testato calcolando la likelihood media dei dataset denominati **Bach**, **Beethoven** e **Mozart** (tabella 2), recuperati con l'utilizzo di music21, trasformati e anch'essi sia normalizzati in scala di C maggiore, sia in versione originale.
Entrambi i modelli sono stati addestrati in una singola run fino ad un massimo di 100 iterazioni, oppure fino a convergenza con una tolleranza di 10^{-4} , qualora occorresse prima.

5.1 HMM

Abbiamo svolto una serie di esperimenti al fine di valutare la bontà del modello addestrato con diverse combinazioni di iper-parametri; nel caso di HMM l'unico iper-parametro è il numero di stati K, che è stato variato da un minimo di 5 ad un massimo di 100, di cui vengono riportati i risultati nelle tabelle seguenti. Abbiamo deciso di mostrare distintamente i risultati prima e dopo il processo di normalizzazione per mostrare la bontà del nostro approccio.

K	Bach	Beethoven	Mozart
100	-9187.85	-337708.09	-47351.68
50	-2509.14	-92598.35	-12975.00
20	-338.79	-12185.90	-1715.10
15	-275.40	-9866.70	-1389.64
10	-197.34	-7032.67	-991.41
5	-221.49	–7921.27	-1115.99

Tabella 3: Likelihood media di una HMM al variare del numeri di stati K per i dataset Bach, Beethoven e Mozart (**non normalizzati**)

K	Bach	Beethoven	Mozart
100	-6943.08	-250125.94	-35193.89
50	-512.18	-13643.01	-2037.62
20	-553.21	-16285.98	-2380.92
15	-338.05	-12084.01	-1702.60
10	-378.24	-13474.57	-1899.67
5	-383.84	-13723.19	-1933.50

Tabella 4: Likelihood media di una HMM al variare del numeri di stati K per i dataset Bach, Beethoven e Mozart (**normalizzati**)

5.2 FHMM

Abbiamo svolto una serie di esperimenti impiegando diverse combinazioni di iper-parametri. Nelle tabelle (6, 8) e (5, 7) sono riportati i risultati ottenuti con FHMM rispettivamente sui dataset normalizzati e non. A differenza di HMM, per FHMM i possibili iper-parametri da variare sono due: K, ovvero il numero degli stati ed M, il numero di catene, di cui vengono riportati i risultati nelle tabelle seguenti. Abbiamo deciso di mostrare distintamente i risultati prima e dopo il processo di normalizzazione per mostrare la bontà del nostro approccio.

K	Bach	Beethoven	Mozart
2	-234.48	-12397.18	-1688.03
3	-237.63	-12341.87	-1664.66
4	-235.30	-12237.85	-1673.18
5	-233.75	-12260.46	-1673.18

Tabella 5: Likelihood media di una FHMM con $\mathbf{M = 2}$ al variare del parametro K per i dataset Bach, Beethoven e Mozart (**non normalizzati**)

K	Bach	Beethoven	Mozart
2	-197.00	-11861.62	-1588.12
3	-187.03	-11927.62	-1609.28
4	-189.24	-11913.43	-1612.30
5	-186.88	-11825.61	-1586.96

Tabella 6: Likelihood media di una FHMM con $\mathbf{M = 2}$ al variare del parametro K per i dataset Bach, Beethoven e Mozart (**normalizzati**)

K	Bach	Beethoven	Mozart
2	-246.42	-12916.95	-1801.28
3	-244.86	-12843.69	-1783.24
4	-244.89	-12830.09	-1782.60

Tabella 7: Likelihood media di una FHMM con $\mathbf{M = 3}$ al variare del parametro K per i dataset Bach, Beethoven e Mozart (**non normalizzati**)

K	Bach	Beethoven	Mozart
2	-191.29	-11936.70	-1624.24
3	-190.94	-11749.73	-1638.00
4	-187.20	-11889.43	-1588.21

Tabella 8: Likelihood media di una FHMM con $\mathbf{M = 3}$ al variare del parametro K per i dataset Bach, Beethoven e Mozart (**normalizzati**)

5.3 Analisi dei risultati

I modelli sono stati addestrati sul dataset di corali di Bach del dataset Chorales e sono stati successivamente valutati, oltre che su una porzione di test dello stesso autore, anche sulle raccolte di brani di Mozart e Beethoven. La likelihood viene calcolata in modo esatto, tramite Forward.

In particolare, la likelihood del modello rispetto al testset di Bach è la minore per ogni configurazione di iper-parametri. Questo risultato è facilmente spiegabile dalla vicinanza tra trainingset e testset.

Abbiamo deciso di valutare la bontà dei nostri modelli anche su autori di musica classica che pur condividendo il genere musicale presentano stili e melodie tra loro molto differenti.

In particolare, la likelihood rispetto al set di brani di Mozart, per entrambi i modelli, ha presentato un discostamento molto inferiore rispetto

al discostamento con i brani di Beethoven. La spiegazione più plausibile è che i due autori, Mozart e Bach, condividano nei propri brani alcune scelte stilistiche e melodiche.

Beethoven si distacca invece notevolmente rispetto ai valori di likelihood. La likelihood su un testset prodotto da quest'ultimo si attesta essere, per entrambi i modelli, un'ordine di grandezza superiore rispetto a Mozart.

Per quanto riguarda HMM, dai risultati in tabella 3 si nota come all'aumentare del numero di stati, pur riuscendo sempre a distinguere in maniera univoca il compositore di un brano, aumenti l'overfitting dei dati risultando in una generale perdita di peformance. I risultati presentano inoltre, sia per i dataset normalizzati che non, la tipica forma ad U, a dimostrazione del compromesso tra bias e varianza [1].

Abbiamo inoltre deciso di confrontare i risultati di likelihood ottenuti con FHMM a due catene di markov (parametro M = 2, tabelle 5 e 6) e a tre catene di markov (parametro M = 3, tabelle 7 e 8). I risultati sono stati generalmente migliori, in particolare sul dataset di Bach, a riprova della bontà della scelta di generalizzare le HMM al fine di estenderne la capacità di rappresentazione.

Entrambi i modelli sembrano giovare della normalizzazione dei dataset, ottenuta applicando la traslazione della tonalità dall'originale a quella in C maggiore. In particolare: HMM migliora notevolmente le performance sopratutto all'aumentare del numero di stati del modello, ottenendo ottime performance anche con un modello a 50 stati (tabella 4); FHMM, con i dataset normalizzati, migliora invece le performance per tutte le combinazioni di iper-parametri testate in questo lavoro (tabella 6, 8).

6 Conclusione

Nel presente lavoro abbiamo impiegato due modelli probabilistici temporali, HMM e FHMM, al fine di modellare processi stocastici musicali. Il task del progetto, ossia l'auto-completamento di sequenze di armonie anche complesse, è stato perseguito seguendo un approccio scientifico e affidandosi alla letteratura in materia. Abbiamo impiegato quale dataset una raccolta di corali di J. S. Bach per la voce soprano. La rappresentazione musicale di ciascuna corale è stato estratta tramite il toolkit music21 [2]. Abbiamo inoltre ricondotto ciascuna corale alla tonalità *standard* di C maggiore. Tale processo è spesso citato in letteratura ed è considerabile una *normalizzazione* di un

set di brani musicali - che potrebbero presentare tonalità molto differenti-. La normalizzazione del dataset ci ha permesso di concentrarci su un task semplificato, ma egualmente interessante, di auto-completamento della *melodia*. Abbiamo svolto esperimenti addestrando i due modelli con diversi set di iperparametri tramite l'algoritmo di expectation maximization; i risultati, in particolare i valori di likelihood dei modelli rispetto al dataset di test, sono presentati in 5. L'interesse recente verso modelli *generativi*, anche di musica, è giustificato dall'impiego che questi potrebbero avere nell'industria musicale e dell'intrattenimento. Abbiamo quindi deciso di impiegare alcuni dei modelli ottenuti dai nostri esperimenti per generare spartiti musicali e melodie. music21 mette a disposizione le API necessarie a generare spartiti e sintetizzare musica a partire da quest'ultimi.

Crediamo fortemente nell'open-source e nella diffusione e nella riproducibilità della ricerca. Pertanto, nel repository del presente progetto è possibile trovare il codice impiegato per gli esperimenti, i dataset impiegati e un set di brani musicali da noi generati coi relativi spartiti (`https://github.com/gianscarpe/chorals_hmm`).

In conclusione, il nostro lavoro lascia interrogativi e nuove prospettive per il futuro nel settore della predizione di sequenze musicali. Nel settore sono emersi negli ultimi anni nuovi approcci che impiegano algoritmi avanzati quali Recursive Neural Network e Generative Adversial Network per generare brani e strutture musicali. Di questi approcci, alcuni sono già proiettati all'impiego in produzione nel settore del gaming e della musica: non possiamo quindi che prospettare nuovi interessanti sviluppi per gli anni a venire.

Elenco delle figure

Riferimenti bibliografici

[1] Zoubin Ghahramani, Michael I. Jordan, Factorial Hidden Markov Models

[2] music21, `https://web.mit.edu/music21/doc/about/about.html`

[3] The MIDI Association, `https://www.midi.org/`

[4] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/Bach+Chorales`. Irvine, CA: University of California, School of Information and Computer Science

[5] Mainous, Frank D. and Robert W. Ottman, Chorales: Mainous and Ottman edition, eds. 1966

[6] Jean-Pierre Briot, Gaëtan Hadjeres, François-David Pachet: Deep Learning Techniques for Music Generation – A Survey

[7] Beth Logan: Factorial Hidden Markov Models for Speech Recognition: Preliminary Experiments

[8] Radford M. Neal and Geoffrey E. Hinton: A New View of the EM Algorithm that Justifies Incremental and Other Variants. 1993

[9] Stamp, Mark. (2004). A revealing introduction to hidden markov models. Science. 1-20.

[10] Regev Schweiger, Yaniv Erlich, Shai Carmi, FactorialHMM: fast and exact inference in factorial hidden Markov models, Bioinformatics, Volume 35, Issue 12, June 2019, Pages 2162–2164, https://doi.org/10.1093/bioinformatics/bty944