# Hidden Markov Models e applicazioni al Data Mining

Andrea Marin

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino 155, 30172 Venezia Mestre, Italy
marin@dsi.unive.it

**Sommario** Relazione di conclusione del corso di *Data Mining* del programma BISS 2006 tenuto dalla prof.sa Rosa Meo.

## 1 Automi stocastici a stati finiti (SFSA)

In questa sezione vogliamo proporre un inquadramento dei Modelli Stocastici di Markov Nascosti all'interno della teoria degli automi, come viene suggerito in [2]. Il vantaggio di questo approccio è duplice: da una parte mette in relazione questo modello ad altri più noti, evidenziandone affinità e differenze, dall'altra consente di mettere in evidenza in modo intuitivo quali sono i vantaggi e i campi di applicazione degli HMM.

Prima di passare all'analisi degli automi stocastici a stati finiti, riteniamo sia opportuno richiamare, più che altro per chiarezza di notazione, la definizione *classica* di automa a stati finiti come data in [6].

**Definizione 1 (Automa a stati finiti)** *Un automa a Stati Finiti (SFA) $\mathcal{M}$ è una macchina astratta costituita da:*

- *Un insieme di stati $\mathcal{Q} = \{I, 1, \ldots, k, \ldots, K, F\}$ in cui $I$ è lo stato iniziale e $F$ lo stato finale (di accettazione)*[1]
- *Un insieme $\mathcal{Y}$ di simboli in ingresso.*
- *Un insieme $\mathcal{Z}$ si simboli in uscita.*
- *Una funzione di transizione $f$. Se $q_t$ indica lo stato dell'automa $M$ all'istante $t$ allora $q_t = f(y_t, q_{t-1})$.*
- *Una funzione di emissione $g$. Nel caso di automi di* Mealy *la funzione di emissione dipende dalla transizione tra gli stati, cioè, se $z_t$ rappresenta l'output dell'automa all'istante $t$, $z_t = g(q_t, q_{t-1})$; Nel caso di automi di Moore la funzione di emissione dipende esclusivamente dallo stato corrente dell'automa, cioè: $z_t = g(q_t)$. È ovvio comunque ricordare che la classe di automi di Mealy è equivalente alla classe di automi di Moore.*

[1] la definizione può essere estesa prevedendo più stati iniziali e/o finali

Riteniamo opportuno aggiungere qualche informazione alla definizione data; solitamente nella teoria degli automi si ricorre a definizioni più precise di quella fornita, introducendo un alfabeto di simboli in input ed output ad esempio; in questo caso, in previsione delle applicazioni che tra poco vedremo, cerchiamo di focalizzare l'attenzione sugli aspetti salienti. Vedremo ad esempio che nelle applicazioni di *speech recognition*, i simboli in input saranno vettori di numeri reali rappresentanti il risultato di un'analisi spettrale di un segnale acustico ad un certo istante, mentre nelle applicazioni di *pattern recognition* torniano ad avere un alfabeto di simboli in ingresso.

Facciamo anche notare che il determinismo dell'automa è garantito dal determinismo della funzione di transizione $f$ e della funzione di emissione $g$. Quando la funzione di emissione o di transizione sono probabilistiche, allora ci troviamo di fronte ad un automa *stocastico* a stati finiti.

**Modelli di Markov (Markov Models)** Come suggerito ad esempio in [2] i modelli di Markov classici possono essere visti come SFSA con funzione di transizione probabilistica e funzione di emissione deterministica (l'identità) dipendente dallo stato corrente. La probabilità di osservare il modello nello stato $q$ all'istante $t$ dipende esclusivamente dallo stato del modello all'istante $t-1$; questa caratteristica della legge di transizione individua l'assenza di memoria dei modelli di Markov[2]. La funzione di emissione può essere considerata lo stato stesso della catena.

Con l'utilizzo di un modello Markoviano possiamo introdurre due problemi di grande interesse:

1. Data una sequenza di $T$ stati $X = x_1 \dots x_T$ e un modello di Markov $\mathcal{M}$ qual è la probabilità che, partendo dallo stato iniziale $I$, $\mathcal{M}$ produca $X$ come sequenza e poi termini[3]?
2. Consideriamo il modello markoviano $\mathcal{M}$ con $K$ stati. Possiamo rappresentare le probabilità di transizione tra gli stati con una matrice $P$ di dimensione $K \times K$ in cui l'elemento $p_{ij}$ rappresenta la probabilità di osservare lo stato $j$ subito dopo aver osservato lo stato $i$. Formalmente se $\lambda$ rappresenta il parametro associato a $\mathcal{M}$, cioè la matrice delle probabilità di transizione, allora il problema consiste nello stimare, data una o più sequenze di addestramento $\lambda^* = \arg\max_\lambda p(X|\mathcal{M}, \lambda)$.

Entrambi i problemi sono di facile risoluzione; nel primo caso, dal momento che la sequenza di cui si desidera stimare la probabilità è esattamente la sequenza di di stati assunti da $\mathcal{M}$ abbiamo che la probabilità di osservare $X = x_1 x_2 \dots x_T$

[2] Questa restrizione può essere indebolita introducendo l'idea di modello Markoviano di ordine $t$. Tuttavia non prenderemo in considerazione il caso, dal momento che è sempre possibile rappresentare un modello Markoviano di ordine $t$ in un modello Markoviano semplice.

[3] Il concetto di terminazione è strettamente legato alla scelta di presentazione del modello basata sulla teoria degli automi.

è:

$$p(X) = p(F|x_T)p(x_1|I)\prod_{t=2}^{T} p(x_t|x_{t-1})$$

Il secondo problema ci chiede una stima (di massima verosimiglianza) del parametro $\lambda$ di $M$ tale che massimizzi la probabilità di osservare il training set. Data la proprietà di assenza di memoria di $M$ possiamo facilmente mostrare che:

$$P(x_t = l|x_{t-1} = k) = \frac{n_{kl}}{n_k}$$

dove $n_{kl}$ rappresenta il numero di volte che nel training set si osserva lo stato $l$ dopo lo stato $k$, mentre $n_k$ rappresenta il numero di volte in cui è visitato lo stato $k$. Possiamo usare quindi $P(x_t = l|x_{t-1} = k)$ come valore di stima per $\lambda^*$.

**Modelli di Markov Nascosti (HMM)** Dal punto di vista degli automi, un HMM si differenza da un modello di Markov per due aspetti:

1. Non è possibile osservare lo stato del modello, quindi non è noto $q_t$
2. La legge di emissione non è una funzione deterministica ma piuttosto una distribuzione di probabilità.

In qualche modo quindi vi sono due processi stocastici, uno osservabile e uno non osservabile, legati dalla legge di emissione.

Oltre quindi alle probabilità di transizione tra gli stati, abbiamo un altro parametro caratterizzante la HMM, cioè la probabilità che, dato lo stato $q_t$ sia osservato $x_t$.

Per i modelli di Markov nascosti possiamo individuare tre problemi estremamente interessati [10]:

1. Data la sequenza $X = x_1 \dots x_T$, qual è la probabilità che essa sia osservata sul modello $\mathcal{M}$?
2. Data la sequenza $X = x_1 \dots x_T$, qual è la sequenza *ottima*[4] di stati di $\mathcal{M}$ che ha prodotto $X$?
3. Dato un training set $X$, come possiamo stimare i parametri del modello $\mathcal{M}$ per massimizzare $P(X|\mathcal{M})$?

A differenza dei Modelli di Markov, in questo caso le soluzioni non sono più elementari. Nei prossimi paragrafi vedremo come affrontare in modo efficiente questi problemi. Prima peró di addentrarci in questo campo, rivisitiamo in modo piú formale, sganciandoci dall'approccio strettamente legato agli automi, la definizione di Hidden Markov Model come proposta nel classico lavoro di Rabiner [10].

**Definizione 2 (Hidden Markov Model)** *Uno Hidden Markov Model è caratterizzato da:*

[4] In seguito approfondiremo il significato di questa richiesta in modo più rigoroso

- *$N$, ovvero il numero di stati del modello; indicheremo ciascuno stato con $S_i$ dove ovviamente $1 \leq i \leq N$ e con $S$ l'insieme degli stati;*
- *$M$, ovvero il numero di simboli osservabili per ogni stato: $V = \{v_1, \ldots, v_M\}$;*
- *$A$, ovvero la matrice $N \times N$ di probabilità di transizione di stato; indicando con $q_t$ lo stato del modello all'istante $t$ abbiamo che $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ con $1 \leq i, j \leq N$;*
- *$B$, insieme di distribuzioni di probabilità dei simboli per ciascuno stato: $B = \{b_j(k),\ 1 \leq j \leq N, 1 \leq k \leq M\}$, dove $b_j(k)$ rappresenta la probabilità di osservare il simbolo $v_k$ ad un certo istante $t$ quando lo stato del modello nel medesimo istante è $S_j$;*
- *$\pi$, distribuzione iniziale $\pi = \{\pi_i,\ 1 \leq i \leq N\}$ dove $\pi_i$ rappresenta la probabilità che lo stato iniziale del modello sia lo stato $S_i$*[5].

Indicheremo ancora con $\mathcal{M}$ un modello HMM, con $\lambda = (A, B, \pi)$ i parametri del modello.

## 2 Il problema della valutazione, *evaluation problem*

Definiamo formalmente il problema: dato un HMM $\mathcal{M}$ e una sequenza di osservazioni $X = x_1 \ldots x_T$ si vuole determinare la probabilità $p$ che quella sequenza sia stata prodotta dall'automa dato.

Naturalmente un approccio *brute-force* prevederebbe la valutazione delle probabilità di ottendere $X$ da tutte le possibili sequenze di stati lunghe $T$ e quindi di effettuarne la somma. Tuttavia un HMM con $N$ stati prevede la valutazione di $N^T$ possibili sequenze, cosa inaccettabile dal punto di vista computazionale anche per piccoli modelli.

Una soluzione efficiente al problema della valutazione è presentata in [1] ed è chiamata *forward and backward procedure*. Definiamo la probabilità $\alpha_t(i)$ di osservare la sequenza $O_1 \cdots O_t$ e che il modello $\lambda$ abbia come stato finale $q_t = S_i$:

$$\alpha_T(i) = Pr(O_1 O_2 \ldots O_T, q_t = S_i | \lambda).$$

Possiamo affermare che la probabilità di osservare solo $O_1$ e che il sistema si trovi al primo passo nello stato $S_i$ è:

$$\alpha_1(i) = Pr(O_1, q_t = S_i | \lambda) = Pr(O_1 | q_t = S_i, \lambda) Pr(q_t = S_i | \lambda) = \pi_i b_i(O_1).$$

Immaginiamo ora di voler calcolare $\alpha_{t+1}(j)$; allo stato $q_j$ al tempo $t+1$ possiamo arrivare da un qualunque altro stato al tempo $t$ con probabilità $a_{ij}$, quindi possiamo impostare la seguente relazione ricorsiva:

$$\alpha_{t+1}(j) = \Big[ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \Big] b_j O_{t+1} \quad 1 \leq t \leq T-1 \wedge 1 \leq j \leq N$$

[5] Possiamo osservare che la definizione è leggermente differente rispetto a quella proposta nell'approccio orientato agli automi dove eravamo comunque in presenza dell'idea di stato iniziale, ma i due modelli sono equivalenti.

In questo modo possiamo calcolare la $P(O|\lambda)$, cioè la somma delle probabilità di riconoscimento della sequenza su un qualsiasi stato finale:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

La complessità dell'algoritmo è $N^2T$. Per risolvere il primo problema possiamo anche ricorrere alla *backward procedure*. L'impostazione ricorsiva è molto simile, e la complessità computazionale è la medesima. Se con $\alpha_t(i)$ indicavamo la probabilità di osservare i primi $t$ caratteri della sequenza e di terminare allo stato $i$, con $\beta_t(i)$ indichiamo ora la probaiblità di osservare gli ultimi $t$ caratteri della sequenza noto lo stato $i$:

$$\beta_t(i) = P(O_{t+1} \cdots O_T | q_t = S_i, \lambda).$$

L'impostazione della ricorsione allora è:

$$\beta_t(i) = \begin{cases} 1 & t = T \ \wedge \ 1 \leq i \leq N \\ \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) & 1 \leq t \leq T-1 \ \wedge \ 1 \leq i \leq N \end{cases}$$

## 3 Ricerca della sequenza ottima

Formalmente il problema richiede che data la sequenza di osservazioni $O$ e il modello $\lambda$ cercare la sequenza di stati $Q$ che massimizzi $P(Q|O, \lambda)$. Innanzitutto osserviamo che:

$$P(Q|O, \lambda) = \frac{P(Q, O|\lambda)}{P(O|\lambda)}.$$

da cui deduciamo che il problema è equivalente alla massimizzazione di $P(Q, O|\lambda)$. Se indichiamo con $\gamma_t(i)$ la probabilità che nella produzione della sequenza $O$, all'istante $t$ il modello $\lambda$ sia allo stato $i$, possiamo dire:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}$$

e quindi la soluzione al nostro problema diventa, localmente, per ogni istante:

$$q_t = \arg\max_{1 \leq i \leq N} [\gamma_t(i)]$$

Questo metodo purtroppo funziona localmente, cioè garantisce di individuare lo stato $q_t$ che massimizza la probabilità dell'osservazione $O_t$, ma non garantisce che calcolando $q_t$ per $1 \leq t \leq T$ sia prodotta una sequenza di stati accettabile per il modello $\lambda$ (alcune $a_{ij}$ potrebbero essere nulle).

Una soluzione che propone una massimizzazione tenendo conto delle sequenze lecite, è invece fornita dal Viterbi Algorithm ([5]). Questo algoritmo è molto simile alla forward procedure con la sostanziale differenza che tenta di massimizzare ricorsivamente i percorsi. Se indichiamo con $\delta_t(i)$ la probabilità della

sequenza di stati ottima che ha prodotto $O_1, \cdots, O_t$ terminante con $q_t = i$ allora possiamo stabilire la seguente relazione ricorsiva:

$$\delta_{t+1} = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1}),$$

e naturalmente $\delta_1(i) = \pi_i b_i(O_1)$. In questo modo calcoliamo sia la probabilità del percorso massimo che l'effettiva sequenza di stati (è sufficiente tenere traccia in un vettore delle scelte effettuate nelle massimizzazioni).

## 4 Addestramento del modello

Il terzo problema riguardante gli HMM è relativo l'addestramento del modello. Si tratta di stimare i parametri del modello $\lambda = (A, B, \pi)$ che, data una sequenza finita di osservazioni $O$ (i.e. il training set), massimizzano $P(O|\lambda)$.

Il risultato è ottenibile mediante metodi di discesa del gradiente ottenendo quindi un minimo (massimo) locale e non globale. Molto spesso la computazione del gradiente è piuttosto difficoltosa, per cui la discesa avviene mediante algoritmi approssimati come il *random direction descent* o il *line search* descritti brevemente in [9]. L'algoritmo più utilizzato, che comunque fornisce massimi locali, è basato su *Expectation-Maximization* o semplicemente *EM*, ed è il Baum-Welch presentato in [4].

Definiamo $\xi_t(i, j)$ come la probabilità di essere nello stato $S_j$ al tempo $t$ e allo stato $S_j$ al tempo $t + 1$, dati modelli ed osservazioni, formalmente:

$$\xi_t(i, j) = Pr(q_t = S_i, q_{t+1} = S_j | O, \lambda).$$

Possiamo esprimere questa probabilità come:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)},$$

dove $P(O|\lambda) = \sum_i \sum_j \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$. Ricordando la definizione di $\gamma_t(i)$ possiamo scrivere la relazione:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

Per effettuare iterativamente una nuova stima dei parametri osserviamo che $\sum_{t=1}^{T-1} \gamma_t(i)$ può essere vista come il numero medio di visite allo stato $S_i$, mentre $\sum_{t=1}^{T-1} \xi_t(i, j)$ è il numero medio di transizioni dallo stato $S_i$ allo stato $S_j$. Da cui otteniamo i seguenti parametri del modello $\lambda'$:

$$\pi' = \gamma_1(t)$$

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$b'_j(k) = \frac{\sum_{t=1 \wedge O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Si può dimostrare che applicando iterativamente queste stime, l'algoritmo converge ad un massimo locale.

Per migliorare le performance dell'addestramento talvolta, in caso di modelli complessi, si ricorre al *Viterbi Learning*: sia per quanto riguarda i metodi del gradiente, che i metodi EM, l'idea di base consiste nel calcolare un solo path, o un gruppo di path più probabili, invece di *tutti* i path possibili. Tuttavia la semplificazione *Viterbi Learning* non sembra essere molto utile in quando ad un miglioramento delle prestazioni (nella pratica si ha un raddoppio della velocità di calcolo) corrisponde una qualità molto inferiore dell'addestramento ottenuto mediante il calcolo di tutti i possibili path.

## 5 Applicazioni di HMM

Le applicazioni degli HMM sono molto vaste nel campo dello *speech recognition* [10] [2] e nel campo della bioinformatica. Alcuni tentativi sono stati fatti per la classificazione di immagini, ma in questo caso il costo computazionale ed i risultati ottenuti sono dubbi. Tuttavia vale la pena citare [8] dove si evidenzia il fatto che la classificazione bidimensionale con HMM soffre meno del problema di minimi (massimi) locali[6] rispetto ad algoritmi con CART e VQ.

Per quanto riguarda le applicazioni degli HMM al data mining in bioinformatica, essenzialmente essi posso contribuire su questi due aspetti:

- Database mining e classificazione di sequenze
- Pattern discovery

### 5.1 Premessa

Gli HMM applicati al data mining sono spesso usati nella bioninformatica. Nei successivi paragrafi faremo riferimento alle applicazioni in questo campo, con particolare riferimento al riconoscimento, confronto di sequenze proteiche. Una proteina è descritta da una sequenza di amminoacidi (che costituiscono un alfabeto di simboli finito); tuttavia, da una parte non siamo certi che le sequenze di amminoacidi che studiamo non contengano errori, dall'altra accade talvolta che un amminoacido, su due proteine con la medesima funzione, possa prendere il posto di un altro. Questo rende impossibile uno studio esatto del problema e apre la strada alla ricerca di metodi efficienti probabilistici. Uno di questi metodi si basa sulle grammatiche probabilistiche, l'altro appunto sugli HMM (sono comunque disponibili in letteratura anche algoritmi sviluppati ad hoc per i problemi di bioinformatica).

### 5.2 Database Mining e classificazione

Se prendiamo un modello HMM addestrato possiamo utilizzarlo per trovare in un database sequenze *simili* a quelle utilizzate per l'addestramento. Il campo di

[6] questo si traduce nella classificazione con l'introduzione di meno rumore. Il rovescio ella medaglia è che gli algoritmi basati su HMM colgono meno dettagli

applicazione principale in questo caso è la bioinformatica: ad esempio tutte le proteine di una stessa famiglia sono associate ad una sequenza di amminoacidi *simile*. È possibile pertanto addestrare un modello su una training set di sequenze note appartenenti ad una stessa famiglia e quindi ricercare in una sequenza di amminoacidi (o un suo frammento) la codifica di proteine appartenenti alla stessa famiglia con un certo grado di probabilità.

Una cosa di cui tener conto nell'uso degli HMM per risolvere questa tipologia di problemi è che la probabilità fornita dall'agoritmo Viterbi tende in generale ad essere sempre più bassa all'allungarsi della sequenza: ciò è facilmente superabile se le sequenze di riconoscimento hanno tutte la stessa lunghezza nota ma, nella maggior parte dei casi, richiede una formula di adattamento dello score fornito dal Viterbi che tenga conto della lunghezza della sequenza analizzata. A seconda dei dati analizzati l'algoritmo per il calcolo dello score può variare.

Per quanto riguarda la classificazione è possibile addestrare un modello per ogni famiglia di proteine, e quindi associare alla sequenza di amminoacidi una probabilità (score) di appartenenza ad una certa famiglia. In questo caso il problema è quello di disporre di un training set sufficientemente ampio da rendere significativa questa operazione. In caso contrario è possibile l'applicazione di algoritmi di clusterning *unsupervised* e quindi l'addestramento dei modelli sui cluster individuati. Esiste anche una possibilità ibrida, usata con successo ancora nella bioinformatica. Si supponga di avere $N$ classi di oggetti all'interno di un database, con $N$ incognito. Si ha una rappresentanza sufficiente come training set per $M$ classi di oggetti; a questo punto si effettua l'addestramento di $M$ modelli, ciascuno con il training set associato ad una delle classi note. È possibile ora sia classificare le sequenze di amminoacidi presenti nel database in base agli $M$ modelli noti, ma anche raggruppare le rimanenti sequenze non classificate in base agli score (normalizzati) ottenuti; l'idea è che se due sequenze di amminoacidi ottengono punteggi simili sugli stessi modelli, allora è probabile che appartengano ad una stessa famiglia. Naturalmente maggiore è il numero $M$ di classi noti nel training set, migliori saranno i risultati derivanti dall'applicazione di questa tecnica. I risultati ottenuti nell'ambito della bioinformatica sono soddisfacenti [9], ma questo è dovuto soprattutto alla particolare struttura delle proteine dal classificare.

Nel rapporto [11] si mettono invece in relazione le performance degli HMM con quelle dei più noti *Learning Vector Quantization* e *Template Matching* sul campo del riconoscimento generico di pattern bidimensionali su immagini. La ricerca prevede l'addestramento dei modelli sullo stesso training set vagliato con K-Mean, e quindi ne studia la percentuale di errore in funzione del rumore introdotto; l'esito mostra come le performance dei tre approcci siano pressochè uguali per bassi livelli di rumore ma anche che al suo crescere, gli algoritmi basati su HMM introducono maggiori errori rispetto agli altri due.

### 5.3 Due esempi dalla biologia

Ci pare inevitabile, per quanto fin qui detto, chiederci perchè gli HMM, che sembrano meno performanti rispetto ad altri analoghi per funzionalità, siano tanto

apprezzati specialmente nel campo della bioinformatica. Per rispondere a questa domanda vediamo due esempi. Il primo, molto semplice, è tratto dal lavoro di Churchill [3]: il DNA è una sequenza di basi che possono essere etichettate con le lettere $A, C, G, T$. Per alcune analisi di sequenziamento si è osservato che i segmenti alternano sezioni in cui sono presenti prevalentemente le basi A-T (stato 1) e sezioni in cui sono prevalentemente presenti le basi G-C (stato 2). Il problema consiste nel capire quando ci si trova, analizzata una sequenza, nello stato 1 o 2. Churchill ha sviluppato sulla base di osservazioni sperimentali, l'HMM di figura 1, costruito sulla base di addestramento ed informazioni biologiche. In

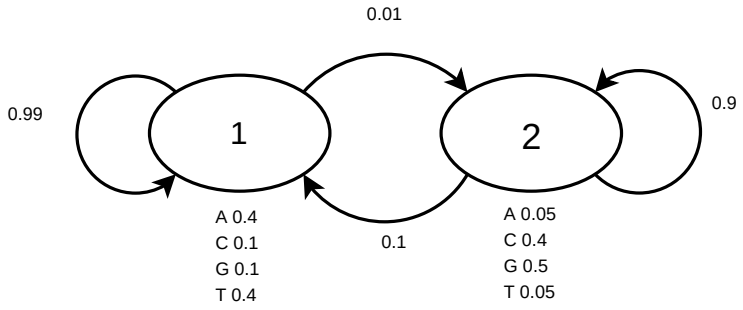


**Figura 1.** HMM per il riconoscimento del DNA di Churchill, attenendosi all'originale sono omessi lo stato iniziale e di riconoscimento

questo caso l'HMM è stato utile perchè ha permesso di intrecciare facilmente le informazioni ottenute dall'addestramento a quelle note per altri studi propri della biologia.

Il secondo esempio evidenza come gli HMM diventino utili in quei casi in cui si vuole avere uno stretto controllo sulle penalità da assegnare alle inserzioni o alle cancellazioni nel riconoscimento della sequenza. Intuitivamente con gli HMM riusciamo a trattare variazioni diverse sui pattern in modo diverso, e questo li rende estremamente flessibili. Una proteina è codificata da amminoacidi: questi costituiscono un alfabeto di venti simboli: per ragioni biologiche, ma anche per errori nel campionamento, due proteine con medesima funzione possono differire nella loro sequenza per:

- sostituzione di un amminoacido con uno simile (tabelle di sostituzione danno il grado di similarità degli amminoacidi)
- inserzione di un amminoacido
- eliminazione di un amminoacido

Naturalmente più dissimilarità presentano due sequenze, meno probabilità vi è che appartengano alla stessa famiglia di proteine. In figura 2 vediamo la rete proposta in [7] per il riconoscimento di sequenze proteiche. Per ogni Matching

si definisce anche uno stato di cancellazione e uno di inserzione. Per ogni transizione verso questi stati si definiscono probabilità basse; inoltre per gli stati di Matching e di inserzione si definiscono le probabilità di emissione di amminoacidi per quella famiglia di proteine. Per ogni prolungamento della sequenza da riconoscere si aggiungono al modello 49 parametri (20 probabiltià di emissione per lo stato M, 20 per lo stato D, più 9 probabilità di transizione). Questo modello è molto flessibile: ad esempio è noto che le inserzioni all'inizio della sequenza sono meno gravi di quelle centrali (potrebbe trattarsi di un problema sperimentale di allineamento), e questo può essere modellato evitando di penalizzare con basse probabilità le transizioni verso lo stato $I_0$. In [7] si mostrano i risultati sperimentali dell'applicazione del modello, e gli esiti del training.

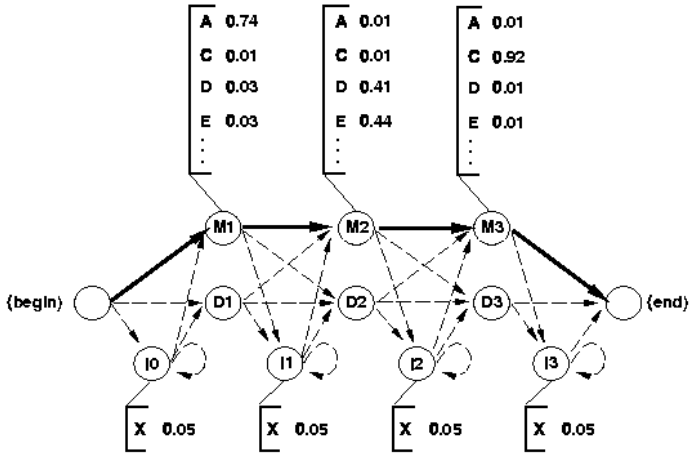


**Figura 2.** HMM per il riconoscimento di proteine proposto da Krogh et al.

### 5.4 Pattern discovery

Nei problemi visti nel paragrafo precedente abbiamo trascurato un aspetto non certo secondario. Come si costruisce il modello iniziale? Infatti il suo addestramento determina le probabilità di transizione degli stati, la probabilità di emissione dei simboli ed infine la distribuzione iniziale. Resta quindi il problema della determinazione della struttura del modello (intesa come il numero degli

stati) e della sua parametrizzazione iniziale[7]. Per capire come questa osservazione conduca al pattern discovery, ritorniamo nel campo della bioinformatica, dove queste tecniche sono state applicate con maggior successo. In della biologia sono note le strutture di alcune proteine: per struttura non intendiamo la sequenza di amminoacidi vera e propria, ma il modo in cui essi si presentano. Ad esempio può esservi un segmento iniziale, quindi delle sequenze che si ripetono più volte, ed una sequenza finale. All'interno di di questo comportamento le sequenze di amminoacidi hanno una certa variabilità e determinano la funzione della proteina. Proteine che svolgono funzioni simili, hanno sequenze diverse ma strutture simili. La struttura della proteina va ad incidere sulla struttura del HMM; quindi l'esito dell'applicazione dell'agoritmo di training a partire da un modello noto, fornisce informazioni sul grado di similarità tra le strutture proteiche; tuttavia questo è un lavoro che deve essere supervised.

### 5.5 Conclusioni

I principali vantaggi dell'impiego degli HMM in task relativi al data mining sono legati da una parte al robusto impianto statistico-teorico che sostiene gli algoritmi forniti, dall'altro alla relativa efficienza degli stessi. Inoltre questi modelli si prestano alle seguente applicazione, spesso indispensabili alla bioinformatica:

- effettuano matching di sequenze che differiscono per qualche inserimento o cancellazione, con valutazione della relativa penalità;
- il training può essere unsupervised;
- trattano agevolmente sequenze di lunghezza variabile;
- possono svolgere compiti di allineamento, data mining/classificazione, analisi strutturale e pattern discovery;
- possono essere elaborati per processi di riconoscimento gerarchico.

I principali limiti nell'applicazione degli HMM sono essenzialmente due. In primo luogo i parametri da stimare crescono velocemente con il numero di stati del modello iniziale, e con l'alfabeto in input. Questo oltre a portare problemi relativi al tempo di calcolo, introducono anche serie difficoltà negli algoritmi di training dovute al moltiplicarsi di massimi locali.
Il secondo problema è che le probabilità di transizione non dipendono dalla funzione di emissione, ma solo dallo stato nascosto. Una classica correlazione che non può essere colta dagli HMM è la seguente: supponiamo che l'emissione di $X$ allo stato $S_i$ sia sovente seguita dall'emissione di $Y$ allo stato $S_j$ e che $X'$ allo stato $S_i$ sia sovente seguita da $Y'$ allo stato $S_j$: purtroppo questo chiaramente esula dalle capacità rappresentative delle catene di Markov del primo ordine, pertanto questa informazione viene persa nell'analisi HMM.

[7] Ricordiamo che tanto gli algoritmi basati sulla discesa del gradiente quanto quelli EM, determinano massimi locali e quindi, in generale, il loro risultato è influenzato dallo stato iniziale del modello

## Riferimenti bibliografici


1. Baum, L. E., and Egon, J. A. An inequality for rational functions with applications to somestatistical estimation problems. *Bull. Amer. Metereol. Soc. 73* (1968), 211–227.
2. Bourlard, H., and Bengio, S. Hidden markov models and other finite state automata for sequence processing. *Idiap Research Report* (2001).
3. Churchill, G. A. Stochastic models for etherogeneus dna sequences. *Bull Math Biol 51* (1989), 79–94.
4. Dempster, A., Laird, N., and Rubin, D. Maximun likelihood from incomplete data via em algorithm. *J. Roy. Stat. Soc. 39*, 1 (1977), 1–38.
5. Forney, G. D. The viterbi algorithm. *Proc. IEEE 61* (1973), 268–278.
6. Hopcroft, J. E., Motwani, R., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*, second ed. Addison-Wesley, 2000.
7. Krogh, A., Brown, B., Mian, I. S., Sjolander, K., and Haussler, D. Hidden markov models in computational biology: applications to protein modeling. *J Mol Biol 235* (1994), 1501–1531.
8. Li, J. S. U., Najmi, A., and Gray, R. M. Image classification by a two-dimensional hidden markov model. *IEEE Transactions on Signal Processing 48*, 2 (2000), 517–533.
9. P. Baldi, S. B. *Bioinformatics: the machine learning approach*, second ed. MIT Press, 2001.
10. Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989).
11. Shahram, M., and Milanfar, P. Hmm-based pattern detection. *Project Report for EE 262: Image Processing and Reconstruction* (2002).