مجله علوم آماری، پاییز و زمستان ۱۳۸۷
جلد ۲، شماره ۲، ص ۱۳۱–۱۴۸

الگوی مارکف پنهان دو طرفه با حافظهٔ خطی

نسیم اجلالی، حمید پزشک

دانشکده ریاضی، آمار و علوم کامپیوتر و قطب زیست ریاضی، دانشگاه تهران و پژوهشگاه دانش‌های بنیادی

تاریخ دریافت: ۱۳۸۷/۱۰/۱۰ تاریخ آخرین بازنگری: ۱۳۸۷/۱۲/۲۲

چکیده : الگوی مارکف پنهان در مسائل بیوانفورماتیک کاربرد فراوانی دارد. برای مثال این الگو در هم‌ردیفی دنباله‌ها، تفسیر خانواده‌های پروتئین و پیش‌بینی ژن بکار می رود. پارامترهای این الگو از طریق الگوریتم بام–ولش تعلیمی که یک الگوریتم EM است برآورد می‌شوند. بکارگیری کارآمدترین الگوریتم‌ها برای دنباله‌های طویل نیازمند حجم وسیعی از حافظه می‌باشد. در این مقاله روش‌های مختلفی از جمله استراتژی پیشرو و استراتژی پسرو را که به منظور کاهش حافظهٔ این الگوریتم ارائه شده‌اند معرفی می‌کنیم. در ادامه الگوریتمی براساس مشاهدات از راست به چپ و از چپ به راست اعضای دنباله ارائه می‌شود که دارای حافظهٔ خطی است. کارایی این الگوریتم بر روی داده‌های شبیه‌سازی شده از خانوادهٔ پروتئین‌ها بررسی می‌شود.

واژه‌های کلیدی : ا لگوی مارکف پنهان، الگوریتم بام–ولش، الگوی دوطرفه، الگوی مارکف پنهان پروفایل، حافظه خطی.

° آدرس الکترونیک مسئول مقاله: حمید پزشک، pezeshk@khayam.ut.ac.ir
° کد موضوع‌بندی ریاضی (۲۰۰۰): ۶۰J۲۰

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

۱ مقدمه

الگوی مارکف پنهان[۱] یک دنبالهٔ متناهی از وضعیت‌های پنهانی است که هر انتقال بین این وضعیت‌ها به گسیل نمادی منجر می‌شود. لذا در این الگو دنباله‌ای از مشاهدات داریم که توسط دنباله‌ای از وضعیتهای پنهان تولید شده‌اند. این الگو از پنج مولّفهٔ زیر تشکیل می‌شود:

۱) یک مجموعهٔ N تایی از وضعیت‌های ممکن، که با نماد $S = \{S_۱, S_۲, \ldots, S_N\}$ نشان می‌دهیم و وضعیتی را که در زمان t رخ می‌دهد با نماد q_t نشان می‌دهیم.

۲) یک مجموعهٔ M تایی از نمادهایی که ممکن است در الگو مشاهده شوند، این مجموعه را با $V = \{v_۱, v_۲, \ldots, v_M\}$ نشان می‌دهیم.

۳) ماتریس احتمال انتقال $A = \{a_{ij}\}$ که برای هر $۱ \le i, j \le N$، $a_{ij} = P(q_{t+۱} = S_j | q_t = S_i)$.

۴) توزیع احتمال مشاهدات، یعنی احتمال گسیل نماد v_k از وضعیت jام:

$$b_j(k) = P(o_t = v_k | q_t = S_j) \qquad ۱ \le j \le N \qquad ۱ \le k \le M$$

که در آن o_t مشاهدهٔ زمان tام است. احتمال‌های فوق را با یک ماتریس $N \times M$، مانند $B = \{b_j(k)\}$ نشان می‌دهیم.

۵) بردار توزیع اولیهٔ وضعیت‌ها، $\Pi = \{\pi_i\}$ است، که در آن

$$\pi_i = P(q_۱ = S_i) \qquad ۱ \le i \le N$$

اجزای ۱ و ۲ ساختمان الگو، و اجزای ۳، ۴ و ۵ پارامترهای آن را مشخص می‌کنند. در ادامه برای راحتی، پارامترها را با سه‌تایی $\lambda = (A, B, \Pi)$ نشان می‌دهیم.

زمانی که از این الگو برای تفسیر یک دنباله استفاده می‌کنیم، این تفسیر یا پیش‌بینی ما به مجموعهٔ پارامترها، یعنی احتمال انتقال‌ها و احتمال گسیل مشاهدات بستگی دارد. برآورد پارامترها از طریق الگوریتم بام‌ـولش که نوع خاصی از الگوریتم EM می‌باشد، امکان پذیر است. الگوریتم EM یک روش کلی برای برآورد

[۱] Hidden Markov Model

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

ماکسیمم درستنمایی پارامترهای مدلی است که شامل داده‌های پنهان می‌باشد. این الگوریتم توسط دمپستر و همکاران (۱۹۷۷) مطرح شد.

الگوریتم بام‌ـولش از طریق مقادیر پسرو و پیشرو به برآورد پارامترها می‌پردازد. بنابراین جدول کامل این مقادیر در حافظه ذخیره می‌شود، اگر T طول دنبالهٔ مشاهدات الگو باشد، ماتریس $N \times M$ از مقادیر پیشرو و پسرو در حافظه ذخیره می‌شود.

الگوی مارکف پنهان کاربرد وسیعی در بیوانفورماتیک دارد از جمله در پیش‌بینی ساختار ژن، همردیفی دنباله‌های پروتئین، آنالیز خانواده‌های پروتئینی. در این موارد معمولاً طول دنبالهٔ مشاهده شده خیلی بزرگ است، لذا برای اجرای الگوریتم بام‌ـولش جدول برنامه‌ریزی پویای بزرگی در حافظه ذخیره می‌شود. برای مثال در مسئلهٔ پیش بینی ژنها طولانی‌ترین کروموزم انسان، تقریباً از ۲۴۵ میلیون باز تشکیل شده است. اگر الگو ۱۰۰ وضعیت داشته باشد، برای ذخیرهٔ جدول برنامه‌ریزی پویا حداقل به ۲۴/۵ گیگابایت حافظه نیاز داریم.

هیرشبرگ (۱۹۷۵) اولین الگوریتم با پیچیدگی خطی برای حافظه را معرفی کرد. این الگوریتم براساس روش حل و تقسیم، نقطهٔ میانی بهترین مسیر را بدون نگه داشتن تمام ماتریس $N \times M$، بدست می‌آورد سپس دو مسئلهٔ کوچکتر را از طریق همان روش حل می‌کند. هیرشبرگ این الگوریتم را برای همردیفی دنباله‌ها ارائه داده بود. اما الگوریتم هیرشبرگ برای یافتن مسیر بهینه ارائه شده بود و در مورد الگوریتم بام–ولش، که تمام مسیرها را در نظر می‌گیرد، کارایی ندارد. گریس و همکاران (۱۹۹۷) خانواده‌ای از الگوریتم‌ها را که برای اجرا به فضای کمتری نیاز داشتند معرفی کردند و نام این الگوریتم را به دلیل اینکه تنها ستونهای خاصی (ستونهای بازرسی) از ماتریس برنامه‌ریزی پویا را نگه می‌دارد، نقطهٔ بازرسی[۲] گذاشتند. این الگوریتم ابتدا برای همردیفی دنباله‌ها (گریس و همکاران، ۱۹۹۷) ارائه شد و بعد از آن توسط هوگی (۱۹۹۷) و تارناس (۱۹۹۸) برای الگوی مارکف پنهان بکار برده شد. این الگوریتم در هنگام محاسبهٔ مقادیر ماتریس برنامه‌ریزی پویا تنها ستون‌های بازرسی را ذخیره می‌کند، سپس در زمان برگشت دوباره ستون‌های

[۲] Checkpointing

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

پاک شده را توسط ستون‌های بازرسی ذخیره شده محاسبه می‌کند. بنابراین زمان محاسبات افزایش می‌یابد.

میکلوز و میر (۲۰۰۵) با تغییرات جزئی در جدول برنامه‌ریزی پویا الگوریتمی برای برآورد پارامترها پیشنهاد دادند. حافظهٔ مورد استفاده توسط این الگوریتم مستقل از طول دنبالهٔ مشاهدات اما بطور خطی وابسته به تعداد وضعیت‌ها می‌باشد. این الگوریتم تنها با محاسبهٔ مقادیر پیشرو پارامترها را برآورد می‌کند. قربانوف و وینترزهیلت (۲۰۰۸) الگوریتم ارائه شده توسط میکلوز و میر را اصلاح کردند. این الگوریتم با محاسبهٔ مقادیر پسرو پارامترها را برآورد کرده و پارامترهای توزیع اولیهٔ مدل را سریعتر برآورد می‌کند.

۲ برآورد پارامترهای الگوی مارکف پنهان با الگوریتم EM

الگوریتم EM ابتدا به طور تصادفی یا با استفاده از اطلاعات قبلی مقادیر اوّلیه‌ای برای پارامترها در نظر می‌گیرد، سپس دو مرحلهٔ زیر را تا رسیدن به همگرایی به طور متناوب تکرار می‌کند:

مرحلهٔ محاسبهٔ امید ریاضی (گام E): محاسبهٔ امید ریاضی تابع درستنمایی با این شرط که پارامترهای مرحلهٔ قبل و داده‌های مشاهده شده را داریم این مقدار را با تابع Q نشان می دهیم.

مرحلهٔ ماکسیمم سازی (گام M): در این مرحله با ماکسیمم کردن تابع Q که از مرحلهٔ قبل بدست آمد، پارامترها را برآورد می‌کند.

با هر تکرار مقدار تابع درستنمایی افزایش می‌یابد و در انتها به یک ماکسیمم نسبی همگرا می‌شود.

الگوی مارکف پنهانی را در نظر بگیرید که دنبالهٔ مشاهدات آن $O = (o_1, \ldots, o_T)$ و دنبالهٔ وضعیت‌های پنهان آن $q = (q_1, \ldots, q_T)$ باشد. اوّلین مرحله، تشکیل تابع Q می‌باشد. مقادیر پارامترها در مرحلهٔ قبل یعنی مرحلهٔ nام بدست آمده‌اند اگر این مقادیر را با $\lambda^{(n)}$ نشان دهیم تابع Q ای که باید ماکسیمم شود

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

برابر است با

$$\begin{aligned}
Q(\lambda,\lambda^{(n)}) &= \sum_{q\in Q}\ln P(O,q|\lambda)P(q,O|\lambda^{(n)}) \\
&= \sum_{q\in Q}\ln[\pi_{q_1}\prod_{t=1}^{T-1}a_{q_tq_{t+1}}\prod_{t=1}^{T}b_{q_t}(o_t)]P(q,O|\lambda^{(n)}) \\
&= \sum_{q\in Q}[\ln(\pi_{q_1})+log(\prod_{t=1}^{T-1}a_{q_tq_{t+1}})+\ln(\prod_{t=1}^{T}b_{q_t}(o_t))]P(q,O|\lambda^{(n)}) \\
&= \sum_{q\in Q}[\ln(\pi_{q_1})P(q,O|\lambda^{(n)})]+\sum_{q\in Q}[\ln(\prod_{t=1}^{T-1}a_{q_tq_{t+1}})P(q,O|\lambda^{(n)})] \\
&+ \sum_{q\in Q}[\ln(\prod_{t=1}^{T}b_{q_t}(o_t))P(q,O|\lambda^{(n)})]
\end{aligned}$$

با کمی محاسبات جبری برآورد پارامترهایی که از ماکسیمم کردن تابع فوق حاصل می‌شوند، عبارتند از

$$\begin{aligned}
\hat{\pi}_i &= \frac{P(q_1=i,O|\lambda^{(n)})}{P(O|\lambda^{(n)})} \\
\hat{a}_{ij} &= \frac{\sum_{t=1}^{T}P(q_t=i,q_{t+1}=j,O|\lambda^{(n)})}{\sum_{t=1}^{T}P(q_t=i,O|\lambda^{(n)})} \\
\hat{b}_j(v_k) &= \frac{\sum_{t=1}^{T}\delta(o_t=v_k)P(O,q_t=i|\lambda^{(n)})}{\sum_{t=1}^{T}P(O,q_t=i|\lambda^{(n)})}
\end{aligned}$$

۳ استراتژی رفت وبرگشت پیشرو

الگوریتم بام–ولش برای برآورد پارامترهای a_{ij} و $b_i(\gamma)$ مقادیر

$$t_{ij}(O) = \sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j) \qquad (1)$$

$$e_i(\gamma,O) = \sum_{t=1}^{T}\delta(o_t=\gamma)\alpha_t(i)\beta_t(i) \qquad (2)$$

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

را بدست می‌آورد، که برای محاسبهٔ آن‌ها احتمالات پیشرو و پسرو به طور همزمان مورد نیاز هستند. بنابراین کل جدول $N \times T$ باید در حافظه ذخیره شود. الگوریتمی که میکلوز و میر (۲۰۰۵) معرفی کردند تنها بر اساس مقادیر پیشرو، عبارات (۱) و(۲) را بدست می‌آورد.ایدهٔ اساسی این الگوریتم این است که مقادیر $t_{ij}(O)$ و $e_i(\gamma, O)$ در واقع مجموع وزنی احتمال عبور از مسیرهایی می باشند که دارای ویژگی‌های خاصی هستند. به عبارت دیگر $t_{ij}(O)$ مجموع وزنی احتمال عبور از تمام مسیرهایی است که دنبالهٔ O را گسیل می کنند و حداقل شامل یک انتقال از i به j می باشند و وزن هر کدام از این احتماها تعداد دفعات انتقال از i به j می‌باشد. $e_i(\gamma, O)$، مجموع وزنی احتمال‌های عبور از تمام مسیرهایی است که دنبالهٔ O را گسیل میکنند و نماد γ حداقل یکبار از وضعیت i گسیل می‌شود. وزن هر کدام از این احتمال‌ها تعداد دفعات گسیل این نماد از وضعیت i می‌باشد. حال نشان می‌دهیم که چطور $t_{ij}(O)$ می‌تواند با حافظهٔ از مرتبهٔ $O(N)$ و رتبه زمانی $O(TNQ_{max})$ محاسبه شود.

قرارداد ۱ : فرض کنید $t_{ij}(t, m)$ مجموع وزنی احتمال عبور از مسیرهایی است که زیر دنبالهٔ $o_1, o_2, \ldots, o_t$ را گسیل می‌کنند و در موقعیت t به وضعیت m ختم می‌شوند. این مسیرها شامل حداقل یک انتقال از i به j می‌باشند و وزن هر کدام از این احتمالها تعداد دفعات این انتقالها می‌باشد.

قضیه ۱ : الگوریتم زیر مقدار $t_{ij}(O)$ را با رتبه حافظهٔ $O(N)$ و رتبه زمانی $O(TNQ_{max})$ محاسبه می کند. مقدار $t_{ij}(t, m)$ در این الگوریتم مجموع وزنی احتمال عبور از تمام مسیرهایی است که دنبالهٔ $o_1, \ldots, o_t$ را گسیل می کنند و حداقل یک انتقال از i به j دارند و وزن آنها برابر تعداد دفعات این انتقال می‌باشد.

$$
\begin{aligned}
&i) \quad \alpha_0(m) = \begin{cases} 1 & m = Start \\ 0 & m \neq Start \end{cases} \\
&\qquad t_{ij}(0, m) = 0 \\
&ii) \quad \alpha_t(j) = [\sum_{i=0}^{N} \alpha_{t-1}(i) a_{ij}] b_j(o_t), \quad t = 1, \ldots, T
\end{aligned}
$$

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

$$t_{ij}(t,m) = \alpha_{t-1}(i)a_{im}b_m(o_t)\delta(m=j) + \sum_{n=1}^{N} t_{ij}(t-1,n)a_{nm}b_m(o_t)$$

$$iii) \quad t_{ij}^{END} = \begin{cases} \alpha_T(i)a_{iEnd} & j = End \\ \sum_{m=1}^{N} t_{ij}(T,m)a_m & j \neq End \end{cases}$$

که $out(s_i)$ مجموعهٔ وضعیت‌های متصل به s_i می‌باشد و

$$\delta(m=j) = \begin{cases} 1 & m=j \\ 0 & m \neq j \end{cases}$$

برهان برای محاسبهٔ مقادیر $\alpha_{t+1}(m)$ و $t_{ij}(t,m)$ $(m \in 1,\ldots,N)$ تنها به N مقدار قبلی آنها نیاز داریم، لذا محاسبهٔ این مقادیر تنها به $O(N)$ حافظه نیاز دارد. برای محاسبهٔ مقادیر پیشرو و $t_{ij}(t,m)$، برای هر موقعیت و هر وضعیت حداکثر Q_{max} عبارت را جمع می‌کنیم. حال از طریق استقراء ثابت می‌کنیم که $t_{ij}(t,m)$ مجموع وزنی احتمال می‌باشد.

۱) در موقعیت $t=0$ هیچ انتقالی رخ نداده است، بنابراین واضح است که مقدار صفر برای $t_{ij}(0,m)$ مجموع وزنی احتمال توالی‌های ممکن می‌باشد.

۲) فرض کنید برای $t=k$ مقدار $t_{ij}(k,m)$ بدست آمده از الگوریتم بالا، مجموع وزنی احتمال تمام توالی های ممکنی باشد که $o_1,\ldots,o_k$ را گسیل می‌کنند و به وضعیت m ختم می‌شوند و حداقل یک انتقال از i به j دارند و وزن هر کدام تعداد این انتقالها می‌باشد. ثابت می‌کنیم $t_{ij}(k+1,m)$ هم این خاصیت را دارد. باتوجه به الگوریتم ارائه شده داریم

$$t_{ij}(k+1,m) = \alpha_k(i)a_{im}b_m(o_{k+1})\delta(m=j) + \sum_{n=1}^{N} t_{ij}(k,n)a_{nm}b_m(o_{k+1}) \quad (۳)$$

عبارت اول سمت راست رابطه (۳) مجموع احتمال عبور از مسیرهایی است که دنبالهٔ $o_1,\ldots,o_{k+1}$ را گسیل کرده‌اند، در موقعیت $k+1$ به وضعیت m ختم شده‌اند و آخرین انتقال این مسیرها انتقال از i به j بوده است $(m=j)$. عبارت دوم سمت راست رابطه (۳) مجموع احتمال عبور از تمام مسیرهای ممکنی است که دنبالهٔ $o_1,\ldots,o_{k+1}$ را گسیل کرده‌اند، در موقعیت $k+1$ به وضعیت m ختم شده‌اند و

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

شامل حداقل یک انتقال از i به j می‌باشند (بنا به فرض $t_{ij}(k,n)$ شامل حداقل یک انتقال از i به j می‌باشد). اگر در مسیری این انتقال در موقعیت k به $k+۱$ رخ داده باشد این احتمال با احتمال یکی از مسیرهای عبارت (۳) جمع می‌شود، لذا یکی به وزن آن مسیر اضافه می‌شود. اما اگر این انتقال رخ نداده باشد وزن آن مسیر برابر با همان وزن قبلی مسیر $t_{ij}(k,n)$ باقی خواهد ماند. در انتها مقادیر بدست آمده را روی m جمع می‌کند تا مجموع وزنی احتمال عبور از مسیرهایی، که دنبالهٔ O را گسیل کرده‌اند بدست بیاورد. این احتمال یعنی t_{ij}^{END} همان رابطهٔ (۱) می‌باشد.

قرارداد ۲ : فرض کنید $e_i(\gamma,t,m)$ مجموع وزنی احتمال عبور از تمام مسیرهای ممکنی است که زیر دنبالهٔ $o_۱,o_۲,\ldots,o_t$ را گسیل می‌کنند، به وضعیت m ختم می‌شوند و نماد γ حداقل یکبار از وضعیت i گسیل شده است. وزن هر کدام از این احتمالها تعداد دفعات گسیل نماد γ از i می‌باشد.

قضیه ۲ : الگوریتم زیر مقدار $e_i(\gamma,O)$ را با حافظه از رتبهٔ $O(N)$ و رتبه زمانی $O(TNQ_{max})$ محاسبه می‌کند. مقدار $e_i(\gamma,t,m)$ در این الگوریتم مجموع وزنی احتمال عبور از تمام مسیرهایی است که دنبالهٔ $o_۱,\ldots,o_t$ را گسیل می‌کنند و نماد γ حداقل یکبار از وضعیت i گسیل شده است. وزن هر کدام از این احتمالها تعداد دفعات گسیل نماد γ از i می‌باشد.

$$i)\quad \alpha_۰(m)=\begin{cases}۱ & m=Start\\ ۰ & m\neq Start\end{cases}$$
$$e_i(\gamma,۰,m)=۰$$
$$ii)\quad \alpha_t(j)=[\sum_{i=۰}^{N}\alpha_{t-۱}(i)a_{ij}]b_j(o_t),\quad t=۱,\ldots,T$$
$$e_i(\gamma,t,m)=\alpha_t(m)\delta(i=m)\delta(\gamma=o_t)+\sum_{n=۱}^{N}e_i(\gamma,t-۱,n)a_{nm}b_m(o_t)$$
$$iii)\quad e_i^{END}(\gamma)=\sum_{m=۱}^{N}e_i(\gamma,T,m)a_{mEnd}$$

برهان مانند قضیهٔ ۱ است.

اگر پارمترها بصورت سری برآورد شوند آنگاه میزان حافظه از مرتبهٔ $O(N)$ خواهد

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

بود.

۴ استراتژی رفت و برگشت پسرو

استراتژی رفت و برگشت پیشرو را برای الگوی با وضعیت‌های خاموش[۳]، یعنی وضعیتی که هیچ مشاهده‌ای را گسیل نمی‌کند، شرح دادیم. در الگوریتمی که ارائه شد برآورد احتمال‌های اولیه یعنی π_iهابه طور خودکار در داخل الگوریتم بدست می‌آمدند. در واقع این احتمالها، احتمال انتقال از وضعیت شروع به دیگر وضعیت‌های غیرخاموش بودند ($a_{Start,i}$). اگر تمام این احتمال‌ها مقادیر غیرصفر داشته باشند این الگوریتم برای هر یک از این N انتقال، Nتا از مقادیر قبلی $t_{ij}(t-۱,m)$ را ذخیره می‌کند پس به میزان $N^۲$ حافظهٔ اضافی نیاز دارد. لذا قربانوف و وینترزهیلت (۲۰۰۸) الگوریتم ارائه شده توسط میکلوز و میر(۲۰۰۵) را بدون وضعیت‌های خاموش بکار بردند. اما برای برآورد پارامترهای احتمال اولیه باید مقادیر پسرو در موقعیت $t=۱$ محاسبه شوند که محاسبهٔ آن‌ها نیاز به محاسبهٔ کل جدول الگوریتم پسرو دارد و این کار میزان محاسبات را بطور قابل توجهی افزایش می‌دهد. استراتژی پسرو برای حل این مشکل ارائه شده است.

الگوریتمی که معرفی می‌کنیم شبیه الگوریتم معرفی شده در قسمت قبل می‌باشد با این تفاوت که این الگوریتم براساس مقادیر پسرو می‌باشد. قبل از تشریح الگوریتم، مقادیر احتمال زیر را تعریف می‌کنیم.

قرارداد ۳: فرض کنید $T_{ij}(t,m)$مجموع وزنی احتمال عبور از مسیرهایی است که زیر دنبالهٔ $o_{t+۱},\ldots,o_T$ را گسیل می‌کنند و در موقعیت t ام در وضعیت m بوده‌اند، این مسیرها شامل حداقل یک انتقال از i به j می‌باشند و وزن هر کدام از این احتمال‌ها تعداد دفعات این انتقال‌ها می‌باشد.

قرارداد ۴: فرض کنید $E_i(\gamma,t,m)$مجموع وزنی احتمال عبور از تمام مسیرهای ممکنی است که زیر دنبالهٔ $o_{t+۱},\ldots,o_T$ را گسیل می‌کنند، در موقعیت t ام در وضعیت m بوده‌اند و نماد γ حداقل یکبار از وضعیت i گسیل شده است. وزن هر

[۳] Silent State

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

کدام از این احتمال‌ها تعداد دفعات گسیل نماد γ از i می‌باشد.
الگوریتم با مقدار اولیهٔ زیر شروع می‌شود:

$$\beta_T(m) = ۱ \qquad T_{ij}(t,m) = ۰$$

سپس بقیهٔ مقادیر $T_{ij}(t,m)$ را از طریق رابطهٔ بازگشتی

$$\begin{aligned}\beta_t(m) &= \sum_{n=۱}^{N} a_{m,n} b_n(o_{t+۱}) \beta_{t+۱}(n) \\ T_{ij}(t,m) &= \beta_{t+۱}(j) a_{m,j} b_j(o_{t+۱}) \delta(i=m) \\ &+ \sum_{n=۱}^{N} a_{m,n} T_{ij}(t+۱,n) b_n(o_{t+۱})\end{aligned}$$

بدست می‌آورد و داریم

$$T_{ij}^{END}(t,m) = \sum_{m=۱}^{N} T_{ij}(۱,m) \pi_m b_m(o_۱).$$

الگوریتم زیر پارامترهای مشاهدات را بر اساس این استراتژی برآورد می‌کند.

$$\begin{aligned}E_i(\gamma,T,m) &= \beta_T(i)\delta(o_T=\gamma) \\ E_i(\gamma,t,m) &= \beta_t(m)\delta(o_t=\gamma)\delta(m=i) \\ &+ \sum_{n=۱}^{N} b_n(o_{t+۱}) a_{m,n} E_i(\gamma,t+۱,n)\end{aligned}$$

در آخر مجموع وزنی احتمال تمام توالی‌های ممکنی را که $o_۱,\ldots,o_T$ را گسیل کرده‌اند و از وضعیت m شروع شده‌اند و نماد γ حداقل یکبار از وضعیت i گسیل شده است را روی m جمع می‌کند. بنابراین

$$E_i^{END}(\gamma) = \sum_{m=۱}^{N} E_i(\gamma,۱,m) \pi_m b_m(o_۱)$$

برای مجموعهٔ آموزشی S داریم

$$a_{ij} = \frac{\sum_{O\in S} T_{ij}^{END}/p(O)}{\sum_{j\in out(s_i)} \sum_{O\in S} T_{ij}^{END}/p(O)}, \quad i \in \tau$$

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

و

$$\hat{b}_j(\gamma) = \frac{\sum_{O\in S} E_j^{END}(\gamma)/p(O)}{\sum_{\gamma=1}^{D}\sum_{O\in S} E_j^{END}(\gamma)/p(O)}, \quad j \in \epsilon$$

که در آن ϵ مجموعهٔ تمام وضعیت‌های است که می‌توانند نمادی را گسیل کنند و τ مجموعهٔ تمام وضعیتهایی است که به وضعیت دیگری منتقل می‌شوند.

۵ الگوی دو طرفه با حافظهٔ خطی

در این قسمت اطلاعات دو طرف گسیل‌ها را در نظر گرفته، و الگوی مناسبی برای برآورد پارامترها ارائه می‌دهیم. به عبارت دیگر یک بار مشاهدات را از راست به چپ و بار دیگر از چپ به راست روی یک دنباله در نظر می‌گیریم. با توجه به این که ارتباط از دو سو درنظر گرفته شده است به طور حتم نتایج بهتری در پیش‌بینی حاصل خواهد شد. اگر دنبالهٔ مشاهدات از راست به چپ را با O و از چپ به راست را با O' نشان دهیم، پارامترها را طوری برآورد می کنیم که احتمال $P(q,O|\lambda)$ و $P(q,O'|\lambda)$ ماکسیمم شوند. در این صورت از طریق میانگین هندسی دو احتمال $P(q,O|\lambda)$ و $P(q,O'|\lambda)$ تابع Q راتشکیل می‌دهیم. قرارمی‌دهیم $P_H = \sqrt{(P(q,O|\lambda))\,(P(q,O'|\lambda))}$ دراین صورت

$$\begin{aligned} Q^*(\lambda,\lambda^{(n)}) &= E\big(\ln P_H|O,O',\lambda^{(n)}\big) \\ &= \frac{1}{2}\sum_{q\in Q}\ln P(O,q|\lambda)P(q|O,\lambda^{(n)}) \\ &+ \frac{1}{2}\sum_{q\in Q}\ln P(O',q|\lambda)P(q|O',\lambda^{(n)}) \end{aligned}$$

اگر Q_1 و Q_2 به ترتیب توابع Q برای O و O' باشند، رابطهٔ بالا بیانگر آن است که $Q^* = \frac{1}{2}Q_1 + \frac{1}{2}Q_2$. برآوردها با ماکسیمم کردن تابع Q^* از طریق روابط

$$\begin{aligned} \hat{\pi}_i &= \frac{P(q_1=i,O|\lambda^{(n)}) + P(q_1=i,O'|\lambda^{(n)})}{P(O|\lambda^{(n)}) + P(O'|\lambda^{(n)})} \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1}\left(P(q_t=i,q_{t+1}=j,O|\lambda^{(n)}) + P(q_t=i,q_{t+1}=j,O'|\lambda^{(n)})\right)}{\sum_{t=1}^{T}\left(P(q_t=i,O|\lambda^{(n)}) + P(q_t=i,O'|\lambda^{(n)})\right)} \end{aligned}$$

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^{T} \left(\delta(o_t = v_k)P(O, q_t = i|\lambda^{(n)}) + \delta(o'_t = v_k)P(O', q_t = i|\lambda^{(n)})\right)}{\sum_{t=1}^{T} \left(P(O, q_t = i|\lambda^{(n)}) + P(O', q_t = i|\lambda^{(n)})\right)}$$

بدست می‌آیند. در اینجا الگوریتم بام–ولش دو بار فراوانی‌ها را می‌شمارد، با توجه به قسمت ۴ برای محاسبهٔ این مقادیر در حافظهٔ خطی (برای مجموعهٔ آموزشی S) داریم

$$a_{ij} = \frac{\sum_{O\in S} T_{ij}^{END}/p(O|\lambda) + \sum_{O'\in S'} T_{ij}^{'END}/p(O'|\lambda)}{\sum_{j\in out(s_i)} \left(\sum_{O\in S} T_{ij}^{END}/p(O|\lambda) + \sum_{O'\in S'} T_{ij}^{'END}/p(O'|\lambda)\right)} \quad (۴)$$

و

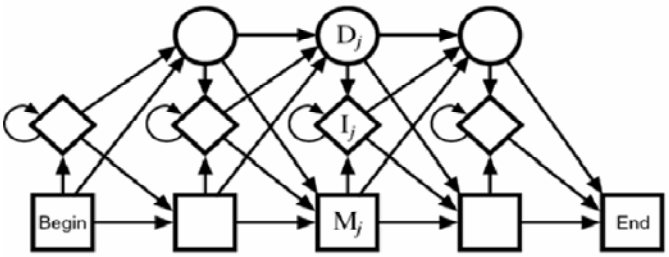
$$\hat{b}_j(\gamma) = \frac{\sum_{O\in S} E_j^{END}(\gamma)/p(O|\lambda) + \sum_{O'\in S'} E_j^{'END}(\gamma)/p(O'|\lambda)}{\sum_{\gamma=1}^{D} \left(\sum_{O\in S} E_j^{END}(\gamma)/p(O|\lambda) + \sum_{O'\in S'} E_j^{'END}(\gamma)/p(O'|\lambda)\right)}$$

۱.۵ مقایسهٔ الگوریتم‌ها

یکی از کاربردهای الگوی مارکف پنهان در زیست‌شناسی ملکولی، ساختن پروفایل احتمالی از یک خانواده از پروتئین‌ها می‌باشد، که به آن HMM پروفایل می‌گویند. ساختن یک HMM پروفایل از دنباله‌هایی که همردیف نشده‌اند از طریق الگوریتم بام–ولش امکان پذیر است. ساختار این مدل در شکل ۱ نشان داده شده است. ردیف پایین، مربع ها وضعیت‌های انطباق[۴] هستند، که به آن‌ها وضعیت‌های اصلی می‌گوییم، زیرا تعداد ستون‌های همردیفی را تعیین می‌کنند. ردیف دوم، وضعیت‌هایی که با شکل لوزی نشان داده شده‌اند وضعیت‌های درج[۵] نامگذاری شده‌اند. این وضعیت‌ها برای الگوبندی قسمتی از پروفایل استفاده می‌شود که تغییرات خیلی زیادی در همردیفی آن مشاهده می‌شود. ردیف بالایی یعنی دایره‌ها وضعیت‌های حذف[۶] نام دارند که هیچ مشاهده‌ای را گسیل نمی‌کنند و و وضعیت‌های خاموش نام دارند. این وضعیت‌ها تنها این امکان را فراهم می‌کنند

۴ Match
۵ Insert
۶ Delete

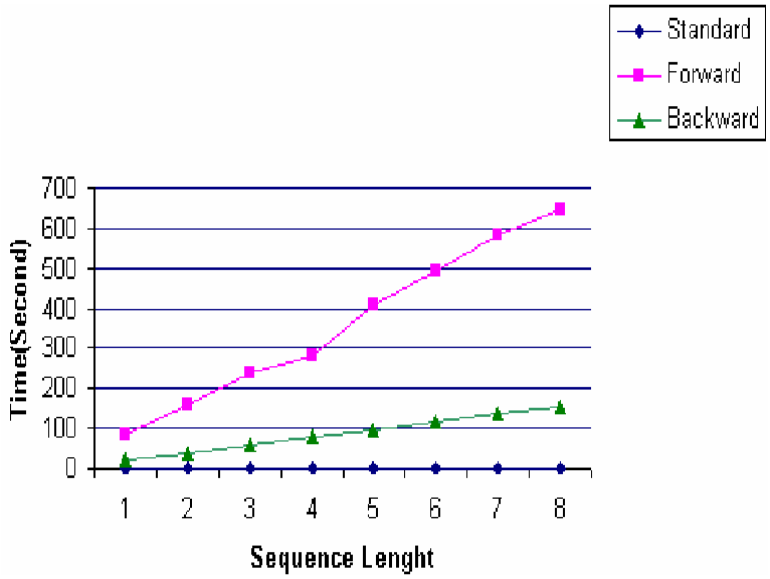
[DOR: 20.1001.1.17358183.1387.2.2.1.7]



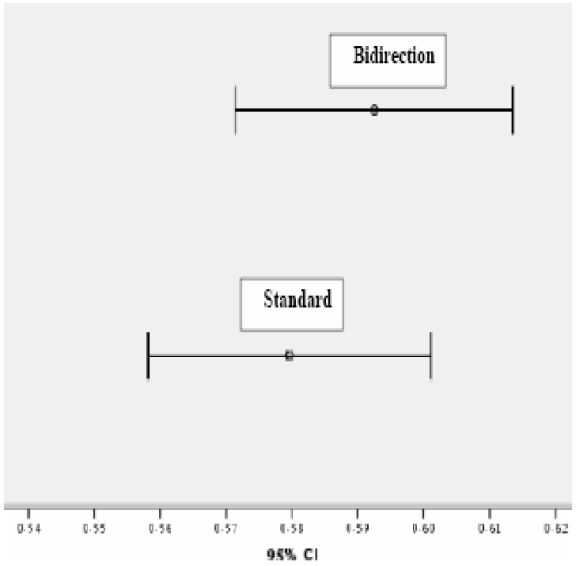
شکل ۱: ساختار الگوی مارکف پنهان یک پروفایل.

که از یک یا چند ستون در همردیفی رد شویم. به منظور مقایسهٔ زمان اجرای الگوریتم‌های استاندارد (بام–ولش)، استراتژی پیشرو و پسرو، به اجرای این الگوریتم‌ها بر روی چندین دنباله از اسید آمینه با طول‌های متفاوت پرداخته‌ایم. زمان ها برای یک دور از اجرای این الگوریتم‌ها بدست آمده‌اند. در اینجا ۱۶۶ وضعیت انطباق در نظر گرفته‌ایم، یعنی به طور کلی ۵۰۱ وضعیت داریم. طول دنباله‌ها بترتیب ۵۰۰، ۱۰۰۰، ۱۵۰۰، ۲۰۰۰، ۲۵۰۰، ۳۰۰۰، ۳۵۰۰ و ۴۰۰۰ می‌باشد. با توجه به شکل ۲ زمان اجرای الگوریتم با استراتژی پیشرو، بیشتر از الگوریتم استاندارد است. شایان ذکر است که میزان حافظهٔ الگوریتم با استراتژی پیشرو خیلی کمتر از الگوریتم استاندارد است. الگوریتم با استراتژی پسرو بدون در نظر گرفتن وضعیت‌های خاموش شروع و پایان، زمان اجرای کمتری نسبت به الگوریتم پیشرو دارد بنابراین کارایی الگوریتم پسرو بهتر از پیشرو است. برای مقایسهٔ دقت الگوی استاندارد با الگوی دوطرفه یک HMM پروفایل با ۶ وضعیت انطباق شبیه‌سازی کردیم، سپس پارامترها را یکبار از طریق الگوی استاندارد و بار دیگر از طریق الگوی دوطرفه برآورد کردیم. در ادامه با پیش‌بینی مسیرهای بهینه از طریق الگوریتم پیشرو–پسرو به مقایسهٔ دقت این دو الگو پرداخته‌ایم، نتایج در نمودار شکل ۳ نشان داده شده است. همانطور که ملاحظه می‌شود میانگین دقت افزایش یافته است. میانگین برای الگوی استاندارد ۵۷/۹۶ درصد و برای الگوی دو طرفه ۵۹/۲۴ درصد می‌باشد. فاصله اطمینان ۹۵ درصد برای میانگین دقت از (۶۰/۱

[DOR: 20.1001.1.17358183.1387.2.2.1.7]



شکل ۲: مقایسه‌ی زمان اجرای الگوریتم‌ها



شکل ۳: نمودار خطای دقت‌های برآورد شده

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

، ۵۵/۹) به (۶۱/۴ ، ۵۷/۲) تغییر یافته است. HMM پروفایل‌ها با وضعیت‌های خاموش شروع و پایان مدل‌بندی شده‌اند. بنابراین با توجه به قسمت ۳ برای برآورد پارامترها براساس الگوی دوطرفه با حافظهٔ خطی، برای مجموعهٔ آموزشی s، روابط

$$\hat{a}_{ij} = \frac{\sum_{O\in S} \frac{t_{ij}^{END}}{p(O|\lambda)} + \sum_{O'\in S'} \frac{t_{ij}'^{END}}{p(O'|\lambda)}}{\sum_{j\in out(s_i)} \left(\sum_{O\in S} \frac{t_{ij}^{END}}{p(O|\lambda)} + \sum_{O'\in S'} \frac{t_{ij}'^{END}}{p(O'|\lambda)} \right)}$$

و

$$\hat{b}_j(\gamma) = \frac{\sum_{O\in S} \frac{e_j^{END}(\gamma)}{p(O|\lambda)} + \sum_{O'\in S'} \frac{e_j'^{END}(\gamma)}{p(O'|\lambda)}}{\sum_{\gamma=1}^{D} \left(\sum_{O\in S} \frac{e_j^{END}(\gamma)}{p(O|\lambda)} + \sum_{O'\in S'} \frac{e_j'^{END}(\gamma)}{p(O'|\lambda)} \right)}$$

را داریم. اما همانطور که دیدیم سرعت اجرای این الگوریتم خیلی کم است، بنابراین اجرای آن بخصوص برای این الگو که فراوانی‌ها دوبار شمرده می‌شوند خیلی زمانگیر است. ما برای حل این مشکل الگوریتمی بر اساس استراتژی پیشرو–پسرو ارائه می‌دهیم.

همانطور که بیان شد استراتژی پسرو، بدلیل محاسبهٔ مقادیر پسرو، پارامترهای انتقال از وضعیت شروع به دیگر وضعیت‌ها($a_{Start,i}$) را سریعتر از استراتژی پیشرو برآورد می‌کند. اما استراتژی پیشرو، بدلیل محاسبهٔ مقادیر پیشرو پارامترهای انتقال از هر وضعیتی به وضعیت پایان ($a_{i,End}$) را سریعتر از استراتژی پیشرو برآورد می‌کند. زیرا نیازی به محاسبهٔ مقادیر t برای این انتقال ندارد و این انتقال تنها از طریق محاسبهٔ مقادیر پیشرو برآورد می‌شود.

در الگوریتم جدید پیشنهادی دیگر مقادیر $t_{Start,i}$ تعریف نمی‌شوند. در مرحلهٔ رفت مقادیر پیشرو بدست می‌آیند و با اجرای الگوریتم پیشرو برای مشاهدات از راست به چپ مقادیر انتقال از وضعیت‌های دیگر به وضعیت پایان را می‌شماریم. در این مرحله انتقال‌های دیگر برای مشاهدات از چپ به راست شمارش شده‌اند. در مرحلهٔ برگشت از طریق استراتژی پسرو مقادیر انتقال از راست به چپ را می‌شماریم و چون مقادیر پسرو بدست آمده‌اند، با اجرای الگوریتم پسرو برای مشاهدات از چپ به راست مقادیر انتقال از وضعیت شروع به دیگر وضعیت‌ها را دوبار می‌شماریم.

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

پس در الگوریتم جدید استراتژی پیشرو یا پسرو را دوبار اجرا نمی‌کنیم، بلکه برای شمارش فراوانی‌ها از چپ به راست از استراتژی پسرو و برای راست به چپ از استراتژی پیشرو استفاده می‌کنیم. در الگوریتم جدید پارامترها به صورت

$$\hat{a}_{ij} = \frac{\sum_{O\in S}\frac{t_{ij}^{END}}{p(O|\lambda)}+\sum_{O'\in S'}\frac{T_{ij}'^{END}}{p(O'|\lambda)}}{\sum_{j\in out(s_i)}\left(\sum_{O\in S}\frac{t_{ij}^{END}}{p(O|\lambda)}+\sum_{O'\in S'}\frac{T_{ij}'^{END}}{p(O'|\lambda)}\right)}$$

$$\hat{b}_j(\gamma) = \frac{\sum_{O\in S}\frac{e_j^{END}(\gamma)}{p(O|\lambda)}+\sum_{O'\in S'}\frac{E_j'^{END}(\gamma)}{p(O'|\lambda)}}{\sum_{\gamma=1}^{D}\left(\sum_{O\in S}\frac{e_j^{END}(\gamma)}{p(O|\lambda)}+\sum_{O'\in S'}\frac{E_j'^{END}(\gamma)}{p(O'|\lambda)}\right)}$$

$$\hat{a_{Start,i}} = \frac{\sum_{O\in S}\frac{\alpha_1(i)\beta_1(i)}{p(O|\lambda)}+\sum_{O'\in S'}\frac{\alpha_1'(i)\beta_1'(i)}{p(O'|\lambda)}}{\sum_i\left(\sum_{O\in S}\frac{\alpha_1(i)\beta_1(i)}{p(O|\lambda)}+\sum_{O'\in S'}\frac{\alpha_1'(i)\beta_1'(i)}{p(O'|\lambda)}\right)}$$

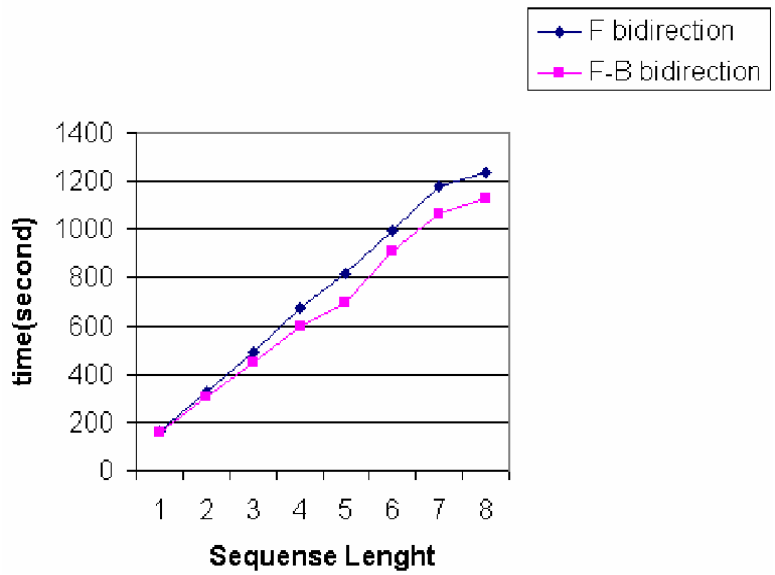
$$\hat{a_{i,end}} = \frac{\sum_{O\in S}\frac{\beta_T(i)a_{i,end}b_{o_T}(i)}{p(O|\lambda)}+\sum_{O'\in S'}\frac{\beta_T'(i)a_{i,end}b_{o_T}(i)}{p(O'|\lambda)}}{\sum_i\left(\sum_{O\in S}\frac{\beta_T(i)a_{i,end}b_{o_T}(i)}{p(O|\lambda)}+\sum_{O'\in S'}\frac{\beta_T'(i)a_{i,end}b_{o_T}(i)}{p(O'|\lambda)}\right)}$$

برآورد می‌شوند. با توجه به شکل ۴ زمان اجرای الگوریتم بر اساس استراتژی پیشرو–پسرو تا حدودی کمتر از الگوریتم بر اساس استراتژی پسرو دو طرفه است. به علاوه میزان حافظهٔ استراتژی پیشرو–پسرو کمتر از استراتژی پیشرو دوطرفه است، زیرا استراتژی پیشرو دو طرفه برای محاسبهٔ مقادیر $t_{Start,i}$ در زمان $t+1$ به $3m+3$ مقدار $t_{Start,i}$ در زمان t نیاز دارد و این مقادیر باید در حافظه ذخیره شوند.

بحث و نتیجه‌گیری

با توجه به کاربرد وسیع الگوی مارکف پنهان در مسائل گوناگون، در این مقاله الگویی ارائه شد که مسیرها را با دقت بیشتری پیش‌بینی می‌کند. به علاوه برای هر توالی با طول‌های متفاوت نیز از نظر حافظه قابل اجرا است. زمان اجرای الگوریتم‌ها یکی از مسائل مهم در تحلیل الگوریتم‌ها بشمار می‌رود، در این مقاله الگوریتمی ارائه کردیم که زمان اجرای آن نسبت به الگوریتم‌های موجود کمتر است و نتایج برای این الگوریتم ارائه شد.

[DOR: 20.1001.1.17358183.1387.2.2.1.7]



شکل ۴: زمان اجرای استراتژی پیشرو دوطرفه و استراتژی پیشرو – پسرو.

تقدیر و تشکر

مولفین از معاونت پژوهشی دانشگاه تهران تشکر می‌کنند. بخشی از این کار توسط گرانتی از پژوهشگاه دانش‌های بنیادی مورد حمایت قرار گرفته است (CS-1385-02).

مراجع

Bilmes, J. (1998), A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, *International Computer Science Institute*, **TR-97-021.**

Churbanov, A. and Winters-Hilt, S. (2008), Implementing EM and Viterbi Algorithms for Hidden Markov Models in Linear Memory, *BMC Bioinformatics*, **9**:224, DOI:10.1186/1471-2105-9-224.

[DOR: 20.1001.1.17358183.1387.2.2.1.7]

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, Series B, **39**, 1-38.

Durbin, R. and Eddy, S., Krogh, A. and Mitchison, G., (1998), *Biological Sequence Analysis*, Cambridge University Press.

Grice, J. A., Hughey, R., Alicia, J. Grice, R. H., and Speck, D., (1997), *Reduced Space Sequence Alignment*, CABIOS **13**, 45-53.

Miklos, I. and Meyer, I. (2005), A Linear Memory Algorithm for Baum-Welch Training, *BMC Bioinformatics* **6**:231.

Rabiner, L. (1989), A Tutorial on Hidden Markov Models and Seleted Applications in Speech Recognitionn., *Proceeding of IEEE*, **77**, 257-286.

Tarnas, C., Hughey, R. (1998),*Reduced Space Hidden Markov Model training.*, Bioinformatics **14**, 401-406.

[DOR: 20.1001.1.17358183.1387.2.2.1.7]