دانشکده ریاضی و علوم کامپیوتر
دانشگاه صنعتی شریف

# فیلتر کالمن (Kalman Filter)

**گزارش مطالعه و پیاده‌سازی فیلتر کالمن**

## پروژه درس بهینه‌سازی در علوم داده

دکتر مجتبی تفاق، ترم دوم سال تحصیلی ۱۴۰۱

**نگارندگان: محسن قدرتی، محمد صالح بهرامی**

# ۱ مقدمه

# ۲ تعاریف اولیه

## ۱.۲ مدل‌های اتوریگرسیو

## ۲.۲ فیلتر

# ۳ معرفی مساله داده‌کاوی

## ۱.۳ صورت مساله و داده‌ها

## ۲.۳ تخمین حافظه موثر

## ۳.۳ تخمین سایز مناسب برای یادگیری دینامیک مساله

## ۴.۳ تهیه تعدادی ویژگی (کنترل دینامیک)

# ۴ فیلتر کالمن

## ۱.۴ تعریف

## ۲.۴ الگوریتم

## ۳.۴ تحلیل زمانی

## ۴.۴ پیاده‌سازی

### ۱.۴.۴ مدل ۱: دینامیک خطی بدون کنترل

فرض می‌کنیم دینامیک تغییرات قیمت به شکل زیر باشد:

$$\tilde{r}_n = f_{0,n}\tilde{r}_{n-1} + \sum_{i=1}^{m-1} f_{i,n} r_{i,n-1} + f_{m,n} + w_n, \qquad w_n \sim \mathcal{N}(0,\ \sigma_n^2). \tag{1}$$

به عبارت دیگر، فرض می‌کنیم اگر تغییرات قیمت از $i = 2,\ \ldots,\ m-1$ کندل گذشته تا کندل گذشته را بدانیم و تغییرات قیمت از کندل گذشته تا کندل فعلی را نیز بدانیم، بازدهی پیش رو ترکیبی خطی از این مقادیر به علاوه یک نویز با توزیع نرمال است. در این صورت برای بردار $\mathbf{x^{(n)}}$ فرم معادل زیر را خواهیم داشت:

$$\mathbf{x^{(n)}} = F_n \mathbf{x^{(n-1)}} + \mathbf{w^{(n)}}, \tag{2}$$

که در آن

$$\mathbf{x^{(n)}} = \begin{bmatrix} \tilde{r}_n \\ r_{1,n} \\ \vdots \\ r_{m-1,n} \\ 1 \end{bmatrix}, \quad F_n = \begin{bmatrix} f_{0,n} & f_{1,n} & \cdots & f_{m,n} \\ 1+\tilde{r}_{n-1} & 0 & \ddots & 0 \\ 0 & 1+\tilde{r}_{n-1} & 0 & \vdots \\ 0 & \ddots & & 0 \\ \vdots & 0 & 1+\tilde{r}_{n-1} & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}, \quad \mathbf{w^{(n)}} = \begin{bmatrix} w_n \\ -\tilde{r}_{n-1}^2 \\ \tilde{r}_{n-1} \\ \vdots \\ \tilde{r}_{n-1} \\ 0 \end{bmatrix}.$$

با توجه به این که در هر لحظه $n$، بردار $\mathbf{x^{(n-1)}}$ معلوم است و کواریانس بردار $\mathbf{w^{(n)}}$ تنها در درایه ۱, ۱ ناصفر و برابر با $\sigma_n^2$ است، به دنبال مقادیری از $f_{0,n},\ \ldots,\ f_{m-1,n}$ می‌گردیم که $\sigma_n^2$ را کمینه کند. (در واقع خطای مدل را در حد امکان کاهش دهد.) هم‌چنین توجه می‌کنیم که اضافه کردن عدد ۱ به انتهای بردار $\mathbf{x^{(n)}}$ باعث می‌شود بتوان عرض از مبدا مدل ۱ را به $f_{m,n}$ منتقل کرد و لذا فرض صفر بودن میانگین نویز $w_n$ برقرار است. اکنون سعی می‌کنیم با بازنویسی معادله بالا، فرم مجموع مربعاتی برای یافتن بهترین $f_{i,n}$ها ارایه کنیم.

اگر قرار دهیم $\theta_n = (f_{0,n},\ \ldots,\ f_{m,n})^T$، به دنبال حل مساله بهینه‌سازی زیر هستیم:

$$minimize \quad |\tilde{r}_n - \langle \theta_n,\ \mathbf{x^{(n-1)}} \rangle|$$

اما با توجه به اینکه مساله بالا احتمالا بی‌نهایت جواب دارد و این موضوع باعث تغییرات سریع دینامیک مدل در هر لحظه می‌شود که خلاف طبیعت مدل است، و مهم‌تر آنکه انتظار داریم دینامیک مدل در طول یک پنجره کوتاه مدت ($k$تایی) پایدار باشد (یعنی $\theta_n \simeq \theta_{n-1} \simeq \cdots \simeq \theta_{n-(k-1)}$) و بتوانیم از اطلاعات این پنجره برای تقریب یک دینامیک کم‌نوسان بهره ببریم، مساله بهینه‌سازی فوق را به مساله زیر تعمیم می‌دهیم:

$$minimize \quad |\tilde{r}_{n-(k-1)} - \langle \theta_n, \mathbf{x^{(n-1-(k-1))}} \rangle|^2 + \cdots + |\tilde{r}_n - \langle \theta_n, \mathbf{x^{(n-1)}} \rangle|^2$$

که با در نظر گرفتن ماتریس $X^{(n)}$ به صورت

$$X^{(n)} = \begin{bmatrix} \mathbf{x^{(n-(k-1))}}^T \\ \vdots \\ \mathbf{x^{(n)}}^T \end{bmatrix}, \quad \Longrightarrow \quad X^{(n)} := \begin{bmatrix} X_0^{(n)} & \dots & X_m^{(n)} \end{bmatrix}$$

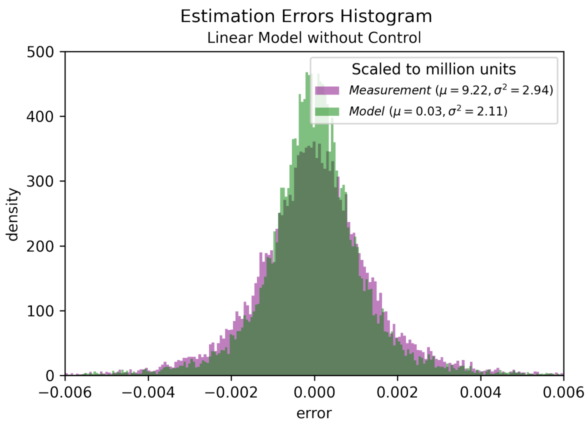
به فرم زیر قابل بیان است:

$$for\ given\ n: \quad minimize_{\theta_n}\ ||X^{(n-1)}\theta_n - X_0^{(n)}||_2^2 \tag{۳}$$

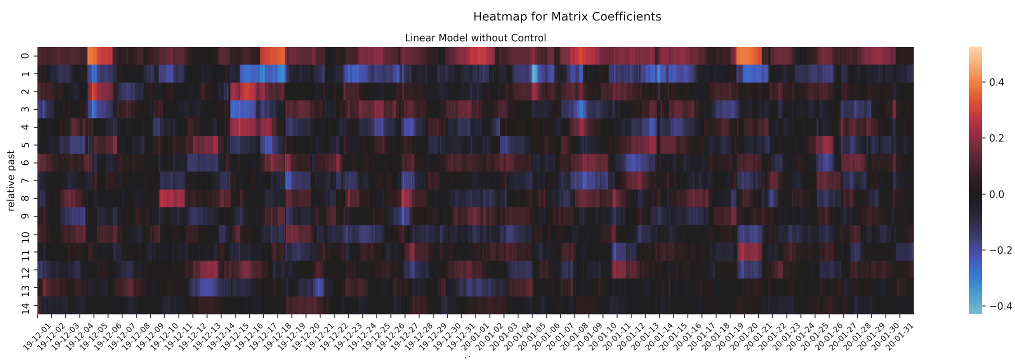
توجه به این نکته ضروری است که از بردار $X_0^{(n)}$، درایه اول مربوط به بازدهی آینده است و در دسترس نیست. لذا به جای حل مساله فوق برای $n$ آن را برای $n-1$ حل می‌کنیم.

در ادامه به بررسی نتایج حاصل از حل مساله بهینه‌سازی ۳ به کمک روش کمترین مجموع مربعات و پیاده‌سازی فیلتر کالمن بر روی مدل ۱ می‌پردازیم.

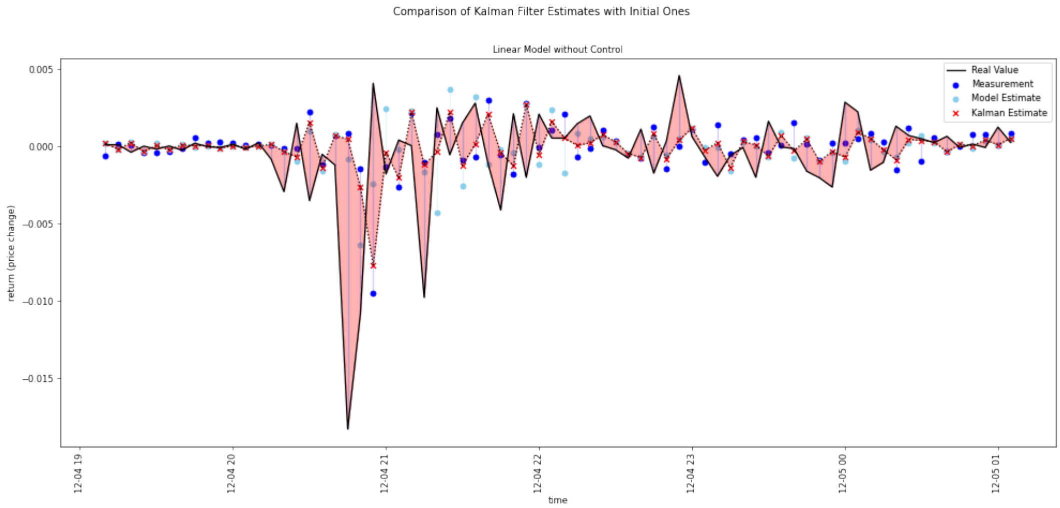
**مشاهده ۱.** مطابق با نمودار زیر می‌توان فرض نرمال بودن نویز در اندازه‌گیری و مدل را تا حدودی تایید کرد. همچنین توجه می‌کنیم که میانگین توزیع نویز، با دقت ۷ رقم اعشار صفر خواهد بود.




**مشاهده ۲.** همان‌طور که اشاره کردیم، فرض می‌کنیم دینامیک مدل دچار تغییرات سریع نمی‌شود. این نکته در نمودار زیر به صورت طیف‌های پیوسته قابل مشاهده است. همچنین در نمودار زیر این نکته قابل توجه وجود دارد که ضرایب بزرگ‌تر و تاثیرگذارتر در مدل خطی پیشنهاد شده، به لحظات نزدیک‌تر به لحظه حال مرتبط می‌شوند. این نیز مشاهده‌ای است که انتظارش را داشتیم.




**نتیجه ۳.** در شکل می‌توان خروجی فیلتر کالمن را بر روی مدل خطی در هر لحظه به ازای یک پنجره خاص زمانی مشاهده کرد.




#### ۲.۴.۴ مدل ۲: دینامیک خطی همراه با کنترل

#### ۳.۴.۴ مدل ۳: دینامیک خطی حساس به زمان

#### ۴.۴.۴ مدل ۴: دینامیک خطی با ماتریس چگال

# ۵ فیلتر کالمن تعمیم‌یافته

## ۱.۵ تعریف

## ۲.۵ پیاده‌سازی

#### ۱.۲.۵ مدل ۵: دینامیک درجه ۲ بدون کنترل

#### ۲.۲.۵ مدل ۶: دینامیک درجه ۲ همراه با کنترل خطی

#### ۳.۲.۵ مدل ۷: دینامیک شبکه عصبی

#### ۴.۲.۵ مدل ۸: دینامیک مشتق‌ناپذیر

# ۶ فیلتر کالمن سریع

# ۷ فیلتر کالمن با چند مدل

## ۱.۷ جمع‌بندی مدل‌های ارایه شده

## ۲.۷ تعمیم فیلتر کالمن به چند مدل موازی

#### ۱.۲.۷ تعمیم ضرایب کالمن

#### ۲.۲.۷ ترکیب چند فیلتر

# ۸ ارجاع و منابع

# ۹ گزاره‌های کاربردی

در این قسمت، مطالب و لم‌هایی را که در حل تمرینات مورد استفاده قرار می‌دهیم، بیان و اثبات می‌کنیم.

# ۱۰ تمرینات

در ادامه همگی تمرینات فصل دوم حل شده‌اند. اما تنها تمرینات 2.6 الی 2.9 تحویلی بوده‌اند که در کنار آن‌ها یک علامت * زده شده است.

**تمرین ۱.** توالی‌های $DNA$ $S_1$ و $S_2$، به ترتیب با طول‌های $n$ و $m$ داده شده‌اند. آیا می‌توانید یک الگوریتم کارا ارائه کنید که تعداد کل هم‌ترازی‌های بین $S_1$ و $S_2$ را برگرداند؟ پیچیدگی زمانی این الگوریتم پیشنهادی چیست؟

حل. جدول برنامه‌ریزی پویا برای الگوریتم Needleman-Wunsch را $M$ در نظر می‌گیریم. بنا به نحوه ساخت این جدول، می‌دانیم هر هم‌ترازی بین رشته‌های $S$ و $T$ یک مسیر از درایه ابتدایی سمت چپ و بالای جدول به درایه انتهایی سمت راست و پایین جدول است که تنها یال‌های مجاز در آن، حرکت به سمت راست یا پایین یا حرکت قطری راست_پایین است. برعکس هر چنین مسیری یک هم‌ترازی را معین می‌کند و لذا مجموعه هم‌ترازی‌ها(ی بدون ستون‌های بدیهی اضافه) با مسیرهای $M[0,0] \to M[n,m]$ در تناظر یک‌به‌یک است. در این بین، ادعا می‌کنیم که مسیرهای $P$ که کلیه یال‌های آن‌ها، یعنی مجموعه $\{v := M[i,j] \to M[i+b_1, j+b_2] | v \in P,\ b_1, b_2 = 0, 1\}$، بهینه باشند، تنها مسیرهای بهینه هستند. منظور از یال بهینه $e = M[i,j] \to M[i+b_1, j+b_2]$ آن است که $e$ در جدول برنامه‌ریزی پویا ظاهر شود. یا به عبارت دیگر، درایه $M[i+b_1, j+b_2]$ بتواند مقدار بهینه خود را با صرف هزینه عملیات نظیر لازم، از درایه $M[i,j]$ اخذ کند. اثبات این ادعا با برهان خلف کار دشواری نیست. کافیست فرض کنیم یالی از $P'$ مانند $e'$ موجود است که در جدول $M$ نیست و این یال، آخرین یال با خاصیت مذکور باشد. مثلا $e' = M[i',j'] \to M[i'+b_1, j'+b_2]$. اکنون به استقرای ریاضی روی مکان این یال، می‌توان به سادگی نشان داد که مقدار درایه $M[i'+b_1, j'+b_2]$ در جدول، از هزینه هم‌ترازی متناظر با $P'$ تا انتهای یال $e'$ اکیدا کمتر است و لذا $P'$ نمی‌تواند یک هم‌ترازی بهینه باشد.

پس تعداد مسیرهای جهت‌دار را می‌خواهیم که از $M[0,0]$ شروع شده و به $M[n,m]$ ختم می‌شوند. برای شمارش این مسیرها، جدول $V$ را از ابعاد مساوی با $M$ می‌سازیم و همه مقادیر آن را در ابتدا برابر با صفر تعیین می‌کنیم. به کمک رابطه بازگشتی زیر و با یک بار پیمایش جدول $V$، مقدار خانه $V[0,0]$ را به عنوان پاسخ باز می‌گردانیم. (در رابطه زیر، $\delta$ همان تابع دلتای دیراک است و $M[i,j] \to M[i',j'] \in M$ یعنی یال فوق، آنطور که تعریف کردیم، بهینه باشد.

$$\begin{cases} V[n,m] = 1 \\ V[i,j] = \Sigma_{b_1, b_2 \in \{0,1\}} V[i+b_1, j+b_2]\delta_{M[i,j] \to M[i+b_1, j+b_2] \in M} \end{cases}$$

اثبات اینکه رابطه بازگشتی فوق، به درستی تعداد مسیرهای $M[i,j]$ به $M[n,m]$ را باز می‌گرداند، سرراست است. در این خصوص توجه می‌کنیم که تعداد مسیرهای با راس شروع $M[i,j]$ که به $M[n,m]$ ختم می‌شوند، طبق اصل جمع برابر است با جمع تعداد مسیرهای از $M[i+b_1, j+b_2]$ به $M[n,m]$ که یال $M[i,j] \to M[i+b_1, j+b_2]$ موجود باشد.

واضح است که پیچیدگی این الگوریتم، برابر با پیچیدگی الگوریتم Needleman-Wunsch خواهد بود به علاوه $O(mn)$ پیچیدگی زمانی برای پیمایش جدول $V$، و $O(mn) + O(\min(m,n)) = O(mn)$ پیچیدگی حافظه برای نگه‌داری جدول $M$ و آپدیت ستونی یا سطری $V$. $\square$

**تمرین ۲.** هم‌ترازی زیر را در نظر بگیرید،

$$\begin{matrix} A & C & - & G & G & T & T & A & T & - \\ - & C & T & G & G & - & - & A & T & C \end{matrix}$$

به همراه ماتریس امتیاز زیر.

| | $A$ | $C$ | $G$ | $T$ |
|---|---|---|---|---|
| $A$ | ۱ | | | |
| $C$ | −۱ | ۲ | | |
| $G$ | −۱ | −۲ | ۱ | |
| $T$ | −۲ | −۱ | −۲ | ۱ |

۱. فرض کنید برای یک گپ از سایز $g$ پنالتی $g - 5 -$ را در نظر بگیریم. امتیاز هم‌ترازی فوق را بر اساس این پنالتی و ماتریس امتیاز داده شده بیابید.

حل. چهار گپ داریم که به ترتیب در مکان‌های $i = [۱], [۳], [۶, ۷], [۱۰]$ هستند. لذا پنالتی یا همان امتیاز منفی تحمیلی این گپ‌ها به ترتیب برابر با $-۶, -۷, -۶, -۶$ خواهد بود. امتیاز سایر بخش‌های هم‌ترازی از روی ماتریس امتیاز قابل محاسبه خواهد بود و مجموع امتیاز برابر است با،

$$-۶ + ۲ - ۶ + ۱ + ۱ - ۷ + ۱ + ۱ - ۶$$

یا $-۱۹$. □

۲. آیا هم‌ترازی فوق بهینه است؟ اگر پاسخ منفیست، امتیاز هم‌ترازی بهینه و یک هم‌ترازی نظیر با آن را ارائه دهید.

حل. خیر. هم‌ترازی‌های زیر امتیازی برابر با $-۸$ دارند و نتیجتا هم‌ترازی سوال بهینه نیست.

$$\begin{array}{cccccccc} A & C & G & G & T & T & A & T \\ C & T & G & G & A & T & C & - \end{array} \qquad \begin{array}{cccccccc} A & C & G & G & T & T & A & T \\ C & T & G & G & A & T & - & C \end{array}$$

اکنون به کمک رابطه بازگشتی زیر ثابت می‌کنیم هم‌ترازی‌های ارائه شده بهینه هستند (که در آن رابطه $A \to B$ به این معنی است که خانه $B$ جدول، مقدار بهینه خود را از خانه $A$ دریافت کرده است).

$$M[i+۱, j+۱] = \max \begin{cases} M[i+۱, j] - ۶ & M[i+۱, j-۱] \not\to M[i+۱, j] \\ M[i+۱, j] - ۱ & M[i+۱, j-۱] \to M[i+۱, j] \\ M[i, j+۱] - ۶ & M[i-۱, j+۱] \not\to M[i, j+۱] \\ M[i, j+۱] - ۱ & M[i-۱, j+۱] \to M[i, j+۱] \\ M[i, j] + w(S[i+۱], T[j+۱]) & \end{cases}$$

جدول بهترین هم‌ترازی توالی‌های $S[۱, \ldots, i]$ و $T[۱, \ldots, j]$ در ادامه آمده است:

| | - | A | C | G | G | T | T | A | T |
|---|---|---|---|---|---|---|---|---|---|
| - | ۰ | −۶ | −۷ | −۸ | −۹ | −۱۰ | −۱۱ | −۱۲ | −۱۳ |
| C | −۶ | −۱ | −۴ | −۹ | −۱۰ | −۱۰ | −۱۱ | −۱۲ | −۱۳ |
| T | −۷ | −۷ | −۲ | −۶ | −۱۱ | −۹ | −۹ | −۱۳ | −۱۱ |
| G | −۸ | −۸ | −۸ | −۱ | −۵ | −۱۱ | −۱۱ | −۱۰ | −۱۵ |
| G | −۹ | −۹ | −۹ | −۷ | ۰ | −۶ | −۷ | −۸ | −۹ |
| A | −۱۰ | −۸ | −۱۰ | −۸ | −۶ | −۲ | −۸ | −۶ | −۱۰ |
| T | −۱۱ | −۱۲ | −۹ | −۹ | −۷ | −۵ | −۱ | −۷ | −۵ |
| C | −۱۲ | −۱۲ | −۱۰ | −۱۰ | −۸ | −۸ | −۶ | −۲ | −۸ |

می‌توان از روی جدول مشاهده کرد که تنها همین دو هم‌ترازی بهینه متمایز را داریم. $\square$

۳. یک ماتریس امتیاز و یک تابع پنالتی گپ ارائه کنید به قسمی که هم‌ترازی داده شده در بالا، یک هم‌ترازی بهینه شود.

حل. ادعا می‌کنیم ماتریس امتیاز زیر به همراه پنالتی ساده $-g$ برای گپ از سایز $g$، هم‌ترازی مورد نظر را یک هم‌ترازی بهینه می‌شناسد.

| | $A$ | $C$ | $G$ | $T$ |
|---|---|---|---|---|
| $A$ | ۱۰ | | | |
| $C$ | $-۱$ | ۱ | | |
| $G$ | $-۱$ | $-۱$ | ۱۰۰ | |
| $T$ | $-۱$ | $-۱$ | $-۱$ | ۱ |

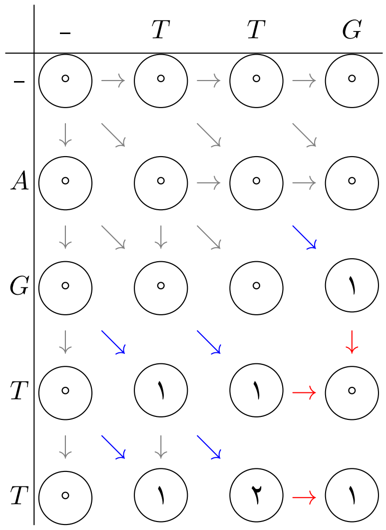
با توجه به امتیاز هم‌خوانی $G$ واضح است که هر هم‌ترازی بهینه برای این ماتریس امتیاز، باید حتما $G$ها را مقابل هم قرار دهد. اکنون برای قسمت‌های $S[۱,۲]$ و $T[۱,۲]$ نیز واضح است که قرار دادن $C$های این دو زیررشته در مقابل هم، امتیاز هم‌ترازی را افزایش می‌دهد. برای زیررشته‌های طرف دیگر $S[۴,\dots,۸]$ و $T[۴,\dots,۷]$ نیز با توجه به امتیاز هم‌خوانی $A$ می‌دانیم حتما باید تک عنصر $A$ موجود در این دو زیررشته هم‌خوان باشد و لذا نشان دادیم هم‌ترازی داده شده یک هم‌ترازی بهینه برای ماتریس امتیاز و پنالتی مفروض ماست. $\square$

**تمرین ۳.** الگوی $P$ از طول $m$ و رشته $T$ از طول $n$ داده شده‌اند. فرض کنید $P^n$ توالی از طول $mn$ باشد که از کنار هم قرار دادن $n$ کپی از $P$ ساخته شده است. هم‌چنین فرض کنید امتیاز هم‌خوانی برابر با ۱ و امتیاز ناهم‌خوانی، حذف، و اضافه برابر با $-۱$ باشد. آیا می‌توانید الگوریتمی با پیچیدگی زمانی $O(nm)$ ارائه کنید که قادر به محاسبه هم‌ترازی موضعی بهینه بین $P^n$ و $T$ باشد؟

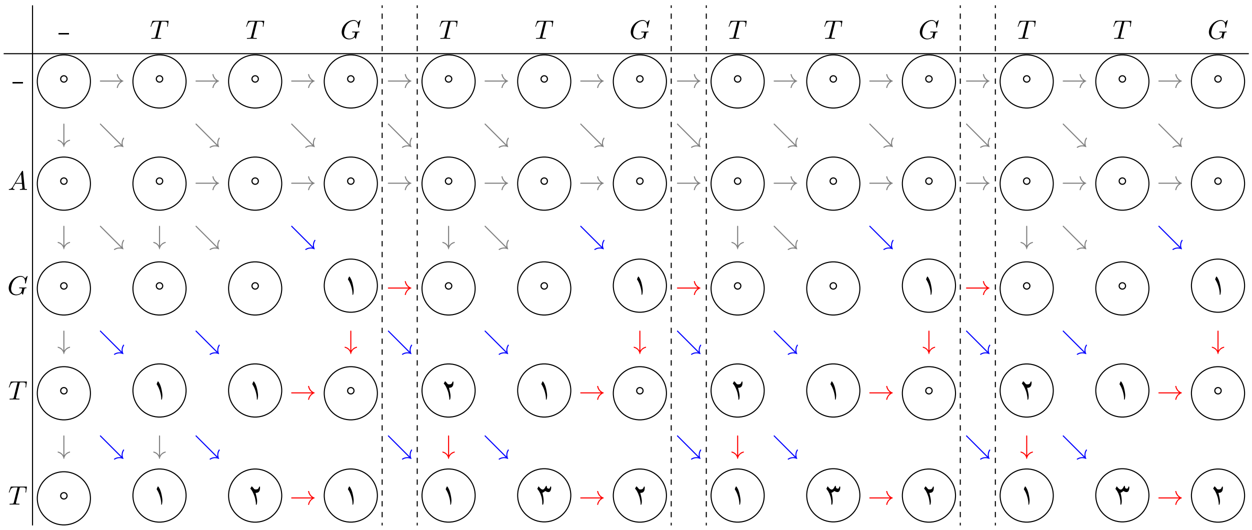
حل. فرض کنیم جدول برنامه‌ریزی پویای هم‌ترازی موضعی رشته‌های $P$ و $T$ را داریم و آن را $M$ بنامیم (موقتا فرض کنیم سطر و ستون - در $M$ ظاهر نشده باشند). توجه می‌کنیم که اگر گراف جهت‌دار مربوط به این جدول را $n$ مرتبه کپی کنیم و کنار هم بگذاریم، و یال‌های جهت‌دار بین درایه‌های ستون آخر جدول $M$ و ستون اول جدول $M$ بعدی را نیز محاسبه و ترسیم کنیم، گراف جهت‌دار متناظر با جدول برنامه‌ریزی پویای مساله بهترین هم‌ترازی موضعی $P^n$ و $T$ حاصل می‌شود. توجه می‌کنیم که هر هم‌ترازی موضعی معادل با مسیری در این گراف جهت‌دار است. اما به دلیل ساختار این گراف جهت‌دار، می‌توانیم به جای بررسی آن، یال‌های مربوط به ستون آخر جدول $M$ را مستقیما به ستون اول $M$ وصل کنیم و به دنبال یک مسیر مناسب در گراف جهت‌دار حاصل $D$ بگردیم. در واقع به کمک آنچه در مسایل قبل بیان و ثابت کردیم، می‌توانیم فرض کنیم هر هم‌ترازی بهینه موضعی توالی‌های $P^n$ و $T$ معادل با یک مسیر در گراف $D$ است.

یال‌های $D$ را بر اساس اینکه یک هم‌خوانی را بیان می‌کنند یا خیر، به ترتیب به دو رنگ آبی و قرمز رنگ‌آمیزی کنیم. امتیازی که هر یال آبی به هم‌ترازی موضعی متناظر با مسیرهای شامل خودش اضافه می‌کند، $+۱$ خواهد بود و امتیاز هر یال قرمز $-۱$. پس در گراف $D$ به دنبال مسیری هستیم که تفاضل تعداد یال‌های آبی از قرمزش بیشینه باشد. به عنوان یک نکته توجه می‌کنیم که طول $P^n$ از طول $T$ کمتر نیست و لذا هر مسیر در $D$، چون حداکثر $n-۱$ بار به سمت پایین یال دارد، حداکثر $n-۱$ بار ستون آخر $M$ را رد می‌کند و لذا هم‌ترازی معادلی در $P^n$ دارد.

برای مثال به جدول زیر که برای توالی‌های $P := TTC$ و $T := AGTT$ ترسیم شده است، (پیکان‌های خاکستری مربوط به جدول هم‌ترازی سراسری هستند. پیکان‌های قرمز و آبی، مطلوب ماست.)




و جدول زیر که مربوط به توالی‌های $P^۴ = TTGTTGTTGTTG$ و $T = AGTT$ است، توجه می‌کنیم.




اکنون الگوریتمی ارائه می‌کنیم که در زمان $O(mn)$ بتواند بهترین امتیاز در بین همه مسیرهای شروع شونده از هر درایه $M[i,j]$ را به صورت پویا محاسبه کند. ماتریس امتیازات $V$ را هم‌ابعاد با $M$ در نظر میگیریم و مقادیر اولیه درایه‌هایش را برابر با ۰ قرار می‌دهیم و سپس سطرهای $M$ را از پایین به بالا پیمایش می‌کنیم. برای سطر $n$ام قرار می‌دهیم $V[n,j] = ۰$. (چون هم‌ترازی‌های موضعی را بررسی می‌کنیم، مقادیر منفی با ۰ جایگزین می‌شوند.) می‌توانیم فرض کنیم $V[i,j]$ بیانگر امتیاز هم‌ترازی بهینه موضعی توالی‌های $P^n[j+۱,\ldots,mn]$ و $T[i+۱,\ldots,n]$ است (واضح است که هر هم‌ترازی بهینه موضعی را می‌توان فرض کرد از $P$ یا در واقع از اندیسی بین ۱ و $m$ شروع می‌شود). برای سطرهای بالاتر $V$ الگوریتم زیر را داریم:

**به ازای** $i \leftarrow n-۱, \ldots, ۰$ :
  **به ازای** $j \leftarrow ۰, \ldots, m$ :
    $V[i,j] \leftarrow ۰$
    **اگر** $v_{i,j} \to v_{i+۱,j} \in D$ **آنگاه:**
      $V[i,j] \leftarrow \max\{V[i+۱,j]-۱,\ V[i,j]\}$
    **اگر** $v_{i,j} \to v_{i+۱,j+۱} \in D$ و یال قرمز باشد **آنگاه:**
      $V[i,j] \leftarrow \max\{V[i+۱,j+۱]-۱,\ V[i,j]\}$
    **اگر** $v_{i,j} \to v_{i+۱,j+۱} \in D$ و یال آبی باشد **آنگاه:**
      $V[i,j] \leftarrow \max\{V[i+۱,j+۱]+۱,\ V[i,j]\}$

۱۰: $j_0 \leftarrow \arg\max_j \{V[i,j]\}$
۱۱: **به ازای** $k \leftarrow 0, \ldots, m$:
۱۲: $j \leftarrow j_k \mod (m+1)$
۱۳: $j_+ \leftarrow j_k + 1 \mod (m+1)$
۱۴: **اگر** $V[i,j] \to V[i,j_+] \in D$ **آنگاه:**
۱۵: $V[i,j_+] \leftarrow \max\{V[i,j]-1, V[i,j_+]\}$

واضح است که الگوریتم بالا از مرتبه $O(mn)$ است و همینطور برای یافتن $D$ به $O(mn)$ زمان احتیاج داریم. پس در مجموع به زمان و حافظه $O(mn)$ نیاز خواهد بود. خطوط ۱۰ تا ۱۵ الگوریتم ارائه شده فوق، در واقع سعی دارند از اطلاعات یال‌های افقی در $D$ استفاده کنند و از آنجا که این یال‌ها همگی قرمز هستند، درایه ماکسیمم در یک سطر، هیچگاه مقدارش تغییری نمی‌کند. پس با شروع از چنین درایه‌ای می‌توان با یک گردش $O(m)$ کل مقادیر سطر را آپدیت کرد. □

**تمرین ۴.** دو توالی $DNA$ $S_1$ و $S_2$، به ترتیب با طول‌های $n$ و $m$ داده شده‌اند. قصد داریم کم‌ترین تعداد عملیات لازم برای تبدیل $S_1$ به $S_2$ را محاسبه کنیم، وقتی عملیات مجاز شامل (۱) اضافه کردن یک عنصر، (۲) حذف یک عنصر، (۳) جایگزینی یک عنصر، و (۴) برعکس کردن یک زیر رشته از $DNA$ باشد. به علاوه، در مورد عملیات (۴) فرض می‌کنیم که هر گاه بر روی یک بخش از $DNA$ اعمال شود، عناصر آن بخش دیگر امکان تبدیل شدن به چیز دیگری تحت هیچ یک از چهار عملیات ذکر شده را نخواهند داشت. آیا می‌توانید یک الگوریتم کارا ارائه کنید که کمترین تعداد عملیات لازم برای تبدیل $S_1$ به $S_2$ را باز گرداند؟ پیچیدگی زمانی این الگوریتم چه خواهد بود؟

حل. ماتریس $I$ را در نظر می‌گیریم که در آن، $I[i_1,i_2;j_1,j_2]$ امتیاز بهینه هم‌ترازی توالی‌های $S_1[i_1,\ldots,i_2]^R$ و $S_2[j_1,\ldots,j_2]$ را تنها با عملیات (۱)، (۲) و (۳) نشان می‌دهد. رابطه بازگشتی زیر را برای محاسبه $I$ در $O(m^2n^2)$ داریم.

$$\begin{cases} I[i+1,i;j_1,j_2] = (j_2-j_1+1)w_{indel} \\ I[i_1,i_2;j+1,j] = (i_2-i_1+1)w_{indel} \\ I[i_1,i_2;j_1,j_2] = \max \begin{cases} I[i_1+1,i_2;j_1,j_2-1] + w(S_1[i_1],S_2[j_2]) \\ I[i_1+1,i_2;j_1,j_2] + w_{indel} \\ I[i_1,i_2;j_1,j_2-1] + w_{indel} \end{cases} \end{cases}$$

حال $D[i,j]$ را برابر با امتیاز هم‌ترازی بهینه بین رشته‌های $S_1[1\ldots i]$ و $S_2[1\ldots j]$ تعریف کنیم (که در آنها هر چهار عملیات مجاز است). توجه می‌کنیم که شرط ثابت ماندن هر زیررشته بعد از اعمال عملیات (۴) معادل با این است که پس از برعکس کردن یک زیررشته، دیگر فقط امکان اعمال مجدد عملیات (۴) برای عناصر این زیررشته وجود ندارد. به عبارت دیگر، می‌توانیم ابتدا عملیات‌های (۱)، (۲) و (۳) را انجام دهیم. سپس به سراغ عملیات (۴) برویم.

اگر فرض کنیم آخرین مکان اعمال عملیات (۴) در جفت زیررشته $S_1[1\ldots i]$ و $S_2[1\ldots j]$، مکان $(i',j')$ باشد، رابطه بازگشتی زیر را خواهیم داشت:

$$\begin{cases} D[0,j] = j \times w_{indel} \\ D[i,0] = i \times w_{indel} \\ D[i,j] = \max \begin{cases} D[i-1,j-1] + w(S_1[i],S_2[j]) \\ D[i-1,j] + w_{indel} \\ D[i,j-1] + w_{indel} \\ \max_{1\le i'\le i+1,\ 1\le j'\le j+1}\{D[i'-1,j'-1] + I[i',i;j',j]\} - w_{reverse} \end{cases} \end{cases}$$

پیچیدگی زمانی و حافظه برنامه‌ریزی پویای با ضابطه بالا، $O(m^2n^2)$ است. □

# ۱۱ ارجاع و منابع

[?]