

#187 メンバー / TakumiOtagaki / 研究進捗 /

wip jax-capr

Updated at 2025-11-11 22:17

はじめに

実装力がないと言われてクッソーと思ったので Max Ward らに任せることではなく、彼らの jax-rnafold を利用して jax-capr を実装してやろうじゃないか。と考えています。

jax-rnafold の dp table

実装で利用されているのは

- $\xi(i)$: 実質 $Z(i, n)$
- $P(i, j)$: 実質 $Z^b(i, j)$
- $M(p, i, j)$: Z^m と Z^{m1} を併せ持っている (ここがちょっとややこしい)
 - i, j がマルチループの中にあって、その中に少なくとも p 個の塩基対が存在するような全ての部分構造 $\sigma[i; j]$ のボルツマンファクターの和
 - i, j による塩基対は含めない。

また、outside algorithm が未実装であるという課題があるが、これは落ち着いてやればできそう。

forward の漸化式から半自動的に outside を導出できる。

forward 漸化式 in jax-rnafold

$$\begin{aligned}\xi(i) &= \xi(i+1) \cdot \exp(-\beta \cdot 0) \\ &\quad + \sum_{\substack{j \\ j < i}} \xi(j) P(i, j) \\ P(i, j) &= B(f_1(i, j)) + \sum_{\substack{i < h < l < j}} P(h, l) B(f_2(h, l)) \\ &\quad + M(2, i+1, j-1) B(M_c) B(M_p) \\ M(p, i, j) &= M(p, i+1, j) B(M_u) \\ &\quad + \sum_{\substack{i \leq k < j}} P(i, k) B(M_p) M(\max(0, p-1), k+1, j)\end{aligned}$$

ここで $\bar{\xi}$ のようにバーをつけることで backward を表すことになると

$$\begin{aligned}\bar{\xi}(i) &= \bar{\xi}(i-1) + \sum_{j < i-1} \bar{\xi}(j) P(i-1, j) \\ \bar{P}(h, l) &:= \bar{\xi}(h) \xi'(l+1) + \sum_{\substack{i, j \\ i < h < l < j \\ h-i+1+j-l-1 \leq 30}} B(f_2(i, j, h, l)) \bar{P}(i, j) \\ &\quad + \sum_{\substack{l < j}} B(\text{multi_branch}(b_h, b_l)) \begin{bmatrix} M'(1, l+1, j) \bar{M}'(2, h, j) \\ + M'(0, l+1, j) \bar{M}'(1, h, j) \\ + M'(0, l+1, j) \bar{M}'(0, h, j) \end{bmatrix}\end{aligned}$$

のようにしてかける。

また、懸念として $B(f_2)$ の計算で internal outer mismatch を考慮する際に、少し散らかった式が出てくるだろうということが挙げられる。これは実装の際にもう少し詳しくみてみることにする。

これをベクトル化するためには同じ配列長さ d となる任意の h, l つまり

$$\forall (h, l), (h', l'), \text{ s.t. } l - h - 1 = l' - h' - 1 (= d), (h, l) \neq (h', l')$$

に対して $\bar{P}(h, l)$ と $\bar{P}(h', l')$ の計算は互いに依存しないため、それぞれを vectorization することができる。

これらのベクトル化をかませば、対角線のサイズ $O(n)$ だけの計算時間で計算を終わらせることができる（メモリが無限にある限り）。

続いて、 \mathbf{M} についても outside algorithm を走らせる必要があるって、

$$\begin{aligned} M(2, i, j) &= M(2, i+1, j)B(M_u) + \sum_{i < k < j} P(i, k)B(M_p)M(1, k+1, j) \\ M(1, i, j) &= M(1, i+1, j)B(M_u) + \sum_{i < k < j} P(i, k)B(M_p)M(0, k+1, j) \\ M(0, i, j) &= M(0, i+1, j)B(M_u) + \sum_{i < k < j} P(i, k)B(M_p)M(0, k+1, j) \end{aligned}$$

であって $M(2, i, j)$ は P の漸化式でも出てくることから、以下のように書き下すことができる。

$$\begin{aligned} \bar{M}(2, h, l) &= \bar{M}(2, h-1, l)B(M_u) + \bar{P}(h-1, l+1)B(M_c) \\ \bar{M}(1, h, l) &= \bar{M}(1, h-1, l)B(M_u) + \sum_{i < h-1} P(i, h-1)B(M_p)\bar{M}(2, i, l) \\ \bar{M}(0, h, l) &= \bar{M}(0, h-1, l)B(M_u) + \sum_{i < h-1} P(i, h-1)B(M_p)(\bar{M}(0, i, l) + \bar{M}(1, i, l)) \end{aligned}$$

のように書き下すことができる。

これにて、 ξ, P, M それぞれの outside algorithm が仕上がった。あとは実装。

ここまで丁寧な情報整理ができていればすぐに実装できる気がする。

1102 update

内側アルゴリズムと外側アルゴリズムを整理する。

内側アルゴリズム:

$$\begin{aligned} \xi(i) &= \xi(i+1) \cdot \exp(-\beta \cdot 0) \\ &\quad + \sum_{\substack{j \\ j < i}} \xi(j)P(i, j)B(\text{ext_branch}(b_i, b_j)) \\ P(i, j) &= B(f_1(i, j)) + \sum_{\substack{i < h < l \\ i < h < j}} P(h, l)B(f_2(h, l)) \\ &\quad + M(2, i+1, j-1)B(\text{multi_closing}(b_i, b_j)) \\ M(p, i, j) &= M(p, i+1, j)B(\text{multi_unpaired}) \\ &\quad + \sum_{\substack{i \leq k < j}} P(i, k)B(\text{multi_branch}(b_i, b_k))M(\max(0, p-1), k+1, j) \end{aligned}$$

外側アルゴリズム:

$$\begin{aligned} \bar{\xi}(i) &= \bar{\xi}(i-1) + \sum_{\substack{j \\ j < i-1}} \bar{\xi}(j)P(i-1, j)B(\text{ext_branch}(b_{i-1}, b_j)) \\ \bar{P}(h, l) &:= \bar{\xi}(h)\xi'(l+1) + \sum_{\substack{i, j \\ i < h < l \\ i < h < j \\ h-i-1+j-l-1 \leq 30}} B(f_2(i, j, h, l))\bar{P}(i, j) \\ &\quad + \sum_{\substack{j \\ l < j}} B(\text{multi_branch}(b_h, b_l)) \left[\begin{array}{l} M'(1, l+1, j)\bar{M}'(2, h, j) \\ + M'(0, l+1, j)\bar{M}'(1, h, j) \\ + M'(0, l+1, j)\bar{M}'(0, h, j) \end{array} \right] \\ \bar{M}(2, h, l) &= \bar{M}(2, h-1, l)B(M_u) + \bar{P}(h-1, l+1)B(M_c) \\ \bar{M}(1, h, l) &= \bar{M}(1, h-1, l)B(M_u) + \sum_{i < h-1} P(i, h-1)B(M_p)\bar{M}(2, i, l) \\ \bar{M}(0, h, l) &= \bar{M}(0, h-1, l)B(M_u) + \sum_{i < h-1} P(i, h-1)B(M_p)(\bar{M}(0, i, l) + \bar{M}(1, i, l)) \end{aligned}$$

1104 update

アルゴリズムの走査の順番について考えた。

気づいたこととしては

- $\bar{\xi}$ は ξ の計算を左からやるか右からやるかみたいな差しかない (i, j じゃなくて i だけの左右片方しか動かないから) ため、両者の計算には共通して $P(i, j)$ が利用される。forward が終わっていると P は計算済みとなっているため、そのままほぼ独立に $\bar{\xi}$ を計算することができる。
- dp をこの順番にした理由は
 - まず $\bar{\xi}$ が計算済みになるため、 ξ から繋がる table を優先的に計算する必要があるが、それは \bar{P} だけ。だからまずは \bar{P}

- それから \bar{M} を計算する。
- また chatGPT に添削させると、 ξ の更新式が間違っていると指摘される。これはどうやら自動微分に則って更新式を構築すると summation の範囲が $i < j$ となるみたいなことを言っていて、けど二次構造を意識して更新式を作ると絶対に $j < i - 1$ となるのが正しいみたいな、奇妙なことが起こる。これについては考察をしなくてはならないし、McCaskill の自動微分が、outside に一致するかどうかみたいな話に関わってきそうなので、feature work として検討する必要がありそう。ただ、一つは、上述の通り、 x_i は左右の片方 (i) しか情報を持っていなくてそれがなんか悪さをしてるのかなーとか（そんなわけない）

1104 update [Scaling]

以下のような方針で scaling を行う。

scaling factor $s()$

$$s(i) := \exp(-ki)$$

ただし、 k は定数。

具体的には以下のようにする↓

$$\begin{aligned} \xi'(i) &:= s(n-1-i+1)\xi(i) \\ P'(i,j) &:= s(j-i+1)P(i,j) \\ M'(i,j) &:= s(j-i+1)M(p,i,j) \\ \bar{\xi}'(i) &:= s(i-1)\bar{\xi}(i) \\ \bar{P}'(i,j) &:= s(i-1+n-1-j)\bar{P}(i,j) \\ \bar{M}'(p,i,j) &:= s(i-1+n-1-j)\bar{M}(p,i,j) \end{aligned}$$

実際に、上述の漸化式に対して $\bar{P}(i,j) = \bar{P}'(i,j)/s(i-1+n-j)$ のような代入を行うことで、scaled table だけを用いて漸化式を書き直すことができる。

書き直したものは以下になる↓

scaled 外側アルゴリズム:

$$\begin{aligned} \bar{\xi}'(i) &= s(1)\bar{\xi}'(i-1) + \sum_{j < i-1} \bar{\xi}'(j)P'(j,i-1) \\ \bar{P}'(h,l) &:= \bar{\xi}'(h)\xi'(l+1) + \sum_{\substack{i,j \\ i < h, l < j \\ h-i-1+j-l-1 \leq 30}} B(f_2(i,j,h,l,OMM))\bar{P}'(i,j)s(h-i+j-l) \\ &\quad + \sum_{i < h} \left[s(1)M'(1,i+1,h-1)\bar{P}'_{m1}(i,l) \right. \\ &\quad \left. + \{s(1)M'(1,i+1,h-1) + s(h-i)B(M_u(h-i-1))\}\bar{P}'_m(i,l) \right] \\ \bar{M}'(2,h,l) &:= s(1)\bar{M}'(2,h-1,l)B(M_u) + s(2)\bar{P}(h-1,l+1)B(M_c(b_{h-1},b_{l+1})) \\ \bar{M}'(1,h,l) &:= s(1)\bar{M}'(1,h-1,l)B(M_u) + \sum_{i < h-1} P'(i,h-1)B(M_p(b_i,b_{h-1}))\bar{M}'(2,i,l) \\ \bar{M}'(0,h,l) &:= s(1)\bar{M}'(0,h-1,l)B(M_u) + \sum_{i < h-1} P'(i,h-1)B(M_p(b_i,b_{h-1}))(\bar{M}'(0,i,l) + \bar{M}'(1,i,l)) \end{aligned}$$

このように、各更新式で $s(1)$ などが登場する。これらが scaling つき outside algorithm になっている。

forward でも同様の scaling がなされるべきなのだが、これは実は jax-rnafold ではすでに 実装されていて、default で同様の scaled forward algorithm が計算される。

というわけで現時点での outside algorithm の書き下しが完了した。

おそらく次は 要件定義 & 実装、と進めるのが良い。

各個撃破に向けて

v.s. Fill bar E

py

```

for i in range(1, n + 1): # i = 1, 2, ..., n (0-origin)
    for j in range(0, i - 1): # j = 0, 1, ..., i - 2
        for bp_idx in range(NBPS): # AU, UA, GU, UG, CG, GC などのペア
            bp = bp_bases[bp_idx]
            bj = bp[0]
            bim1 = bp[1]
            base_en = bar_E[j] * bar_P[bp_idx, bj, bim1]
            bar_E[i, bim1] += s_table[1] * bar_E[i-1] + sum([
                bar_E[j] * P[j, i-1] for j in range(0, i-1) #j = 0, 1, ..., i - 2
            ])
            bar_E[i, :] += s_table[1] * bar_E[i-1, :]

```

これを jax で書くのが仕事。

v.s. bar P

jax-rnafold をよく読んでいるんだけれど、どうやら multiloop unpaired energy を全く参照していないさうだし、
Multiloop の table M の計算で必要とされている $M^* B(Multi_unpaired * 1)$ みたいなものも見当たらない。これは実装ミスか？ どうも納得がいかない。
以下のように jax-rnafold の source を勝手に修正しておいた (submodules/jax-rnafold)

```

426
427     nb_j_terms = get_nb_j_terms(jnp.arange(3), jnp.arange(seq_len+1))
428     # 1104 update by Takumi Otagaki
429     # for each i, p, ML[:, i, :] += s_table[1] * ML[:, i - 1, :] * em.en_multi_unpaired()
430     ML = ML.at[:, i, :].set(nb_j_terms)
431     ML += s_table[1] * ML[:, i - 1, :] * em.en_multi_unpaired()
432     return ML
433

```

Issues を明日くらいに立てようと思う。

--> done.

OMMについての話

OMM は general internal loop (1N, N1, 22, 23, 32 以外) の塩基対 (i, j) と (h, l) がある時について
 $i < h < l < j$ となっていることを想定する。この時 internal loop の内側の mismatch $(h - 1, l + 1)$ にはエネルギーが与えられていて、その計算のために使われている。
OMM は internal loop の f2 energy を計算する時に使われる。また inner mismatch mm_{ij} といって $(i + 1, j - 1)$ による寄与もある。
まとめると

$$\begin{aligned}
&\# \text{内側アルゴリズム} \\
mm_{i,j} &= \sum_{b_{i+1}, b_{j-1}} B(\text{inner_match}(b_i, b_j, b_{i+1}, b_{j-1})) x_{i+1, b_{i+1}} x_{j-1, b_{j-1}} \\
P(i, j) &= \sum_{1 < h_{\text{off}} \leq 30} \sum_{1 < l_{\text{off}} \leq 30} B(\text{internal_init}(h_{\text{off}} + 1 + l_{\text{off}} + 1)) \\
&\quad \times B(\text{internal_asym}(h_{\text{off}} + 1, l_{\text{off}} + 1)) \\
&\quad \times OMM(h, l) \times mm_{i,j} \times s_table[h_{\text{off}} + 1 + l_{\text{off}} + 1 + 2] \\
&\# \text{外側アルゴリズム} \\
\overline{OMM}(h, l) &= \sum_{1 < i_{\text{off}} \leq 30} \sum_{1 < j_{\text{off}} \leq 30} B(\text{internal_init}(i_{\text{off}} + 1 + j_{\text{off}} + 1)) \\
&\quad \times B(\text{internal_asym}(h_{\text{off}} + 1, l_{\text{off}} + 1)) \\
&\quad \times mm_{i,j} \times \overline{P}(i, j) \\
\overline{P}(h, l)_{\text{general il}} &= \overline{OMM}(h, l)
\end{aligned}$$

このように general internal では \overline{OMM} が \overline{P} の一つのケースとなっていることが判明した。

1105 update

- 大体実装は完成した。

1107 update

outside P の計算のマルチループのところがめちゃくちゃ間違っていた、自動微分を考えたら普通にわかるのに、なんか囚われていた。

scaled 外側アルゴリズム with x_i, x_j terms:

$$\begin{aligned}\xi'(i) &= s(1)\xi'(i-1) + \sum_{\substack{j \\ j < i-1}} \sum_{\text{bp_idx}_{j,i-1} = (b_j, b_{i-1})} \xi'(j)P'(\text{bp_idx}_{j,i-1}, j, i-1) x_{j,b_j} x_{i,b_{i-1}} \\ \bar{P}'(\text{bp_idx}_{h,l}, h, l) &:= \xi'(h)\xi'(l+1) + \sum_{\substack{i,j \\ i < h, j < l \\ h-i-1+j-l-1 \leq 30}} \sum_{\text{bp_idx}_{i,j} = (b_i, b_j)} B(f_2(i, j, h, l)) \bar{P}'(\text{bp_idx}_{i,j}, i, j) s(h-i+j-l) x_{i,b_i} x_{j,b_j} \\ &\quad + \sum_{\substack{j \\ i < j}} x_{h,b_h} x_{i,b_i} B(M_p(b_h, b_l)) \left[\begin{array}{l} M'(1, l+1, j) \bar{M}'(2, h, j) \\ + M'(0, l+1, j) \bar{M}'(1, h, j) \\ + M'(0, l+1, j) \bar{M}'(0, h, j) \end{array} \right] \\ \bar{M}'(2, h, l) &:= s(1)\bar{M}'(2, h-1, l)B(M_u) + s(2) \sum_{\text{bp_idx}_{h-1,l+1} = (b_{h-1}, b_{l+1})} \bar{P}(h-1, l+1)B(M_c(b_{h-1}, b_{l+1})) x_{h-1,b_{h-1}} x_{l+1,b_{l+1}} \\ \bar{M}'(1, h, l) &:= s(1)\bar{M}'(1, h-1, l)B(M_u) + \sum_{i < h-1} \bar{M}'(2, i, l) \sum_{\text{bp_idx}_{i,h-1} = (b_i, b_{h-1})} P'(\text{bp_idx}_{i,h-1}, i, h-1)B(M_p(b_i, b_{h-1})) x_{i,b_i} x_{h-1,b_{h-1}} \\ \bar{M}'(0, h, l) &:= s(1)\bar{M}'(0, h-1, l)B(M_u) + \sum_{i < h-1} (\bar{M}'(0, i, l) + \bar{M}'(1, i, l)) \sum_{\text{bp_idx}_{i,h-1} = (b_i, b_{h-1})} P'(\text{bp_idx}_{i,h-1}, i, h-1)B(M_p(b_i, b_{h-1})) x_{i,b_i}\end{aligned}$$

これが正しい。

1110 update

outside の実装が被ってるということだけを話すのが良い。

エネルギーパラメータの話をやるのはいい

Max Ward らのチームへ各メールに向けて、情報整理

物事には順序があって…

いきなり全部報告するんじゃなくて、outside をやっています..

ある程度やってるけど、統合できる…?みたいなことを話す…

完成ではないしデバッグ中だから、こんなのは作っている.

メリット：デバッグ終わらないうちに公開されたらもっと元気なくなる

最近の論文について、いいねとだけ言っておく、すごく informative だって伝える

長いメールを書いても仕方がない。

CapR でエネルギーパラメータの予測みたいな話はまだ出さなくていい。

彼らの論文を見て考えるべきこともあって

- probing を使って予測するけど…
- エネルギーパラメータを決めるといけないけど
- 塩基対確率でやる
- エネルギーパラメータが決まったとして二次構造予測はどうするのか
 - やっぱり機械学習とエネルギーパラメータを組み合わせた MXFold2 みたいなもの？
 - probing のデータを使って pseudoenergy を使ってやるのも良い
 - 専門家が少ないし、いくらでもアイデアはある。
- まだ一緒にやろうという必要はない
- いずれは、RNA 同情みたいな関係になれば良い
 - タンパク質立体構造のやつはボストンとか MIT とかで定期的に mtg している
- PI 目線
 - 共同研究をやると誰が 1st になるのかとか、考えることが多くなって単純じゃない。
 - こっちから持ち込む話はその学生が 1st でいいんだけど見たいな
- overthinking, negative thinking
 - <https://amzn.asia/d/4ljDX2v>