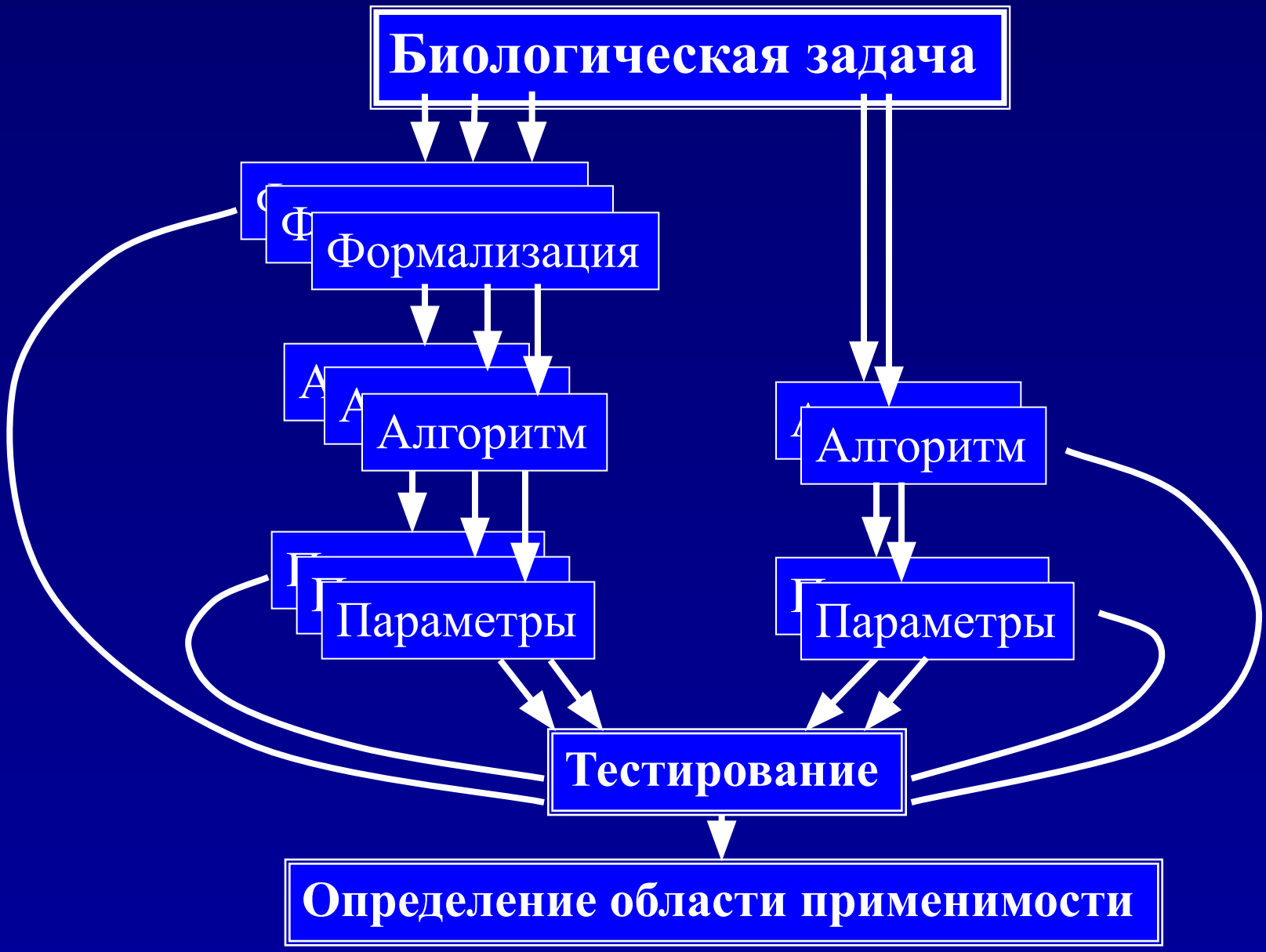
Алгоритмы биоинформатики

ФББ

2013 г., весенний семестр, 3-й курс.

Миронов Андрей Александрович
Спирин Сергей Александрович

Информатика и Биоинформатика



Пример: сравнение последовательностей

- ***Тестирование:*** **алгоритм должен распознавать последовательности, для которых известно, что они биологически (структурно и/или функционально) сходны**

Сравнение последовательностей

- *Формализация1:* глобальное выравнивание
- *Алгоритм1:* Граф выравнивания, динамическое программирование
- *Алгоритм1а:* Граф выравнивания, динамическое программирование, линейная память
- *Параметры:* Матрица сходства, штраф за делецию

Сравнение последовательностей

- *Формализация2:* локальное выравнивание
- *Алгоритм2:* Граф локального выравнивания, динамическое программирование
- *Параметры:* Матрица сходства, штраф за делецию

Сравнение последовательностей

- *Формализация3:* локальное выравнивание с аффинными штрафами
- *Алгоритм3:* Расширенный граф локального выравнивания, динамическое программирование
- *Параметры:* Матрица сходства, штраф за открытие делеции, штраф за расширение делеции

Сравнение последовательностей

- *Алгоритм4:* BLAST. Формальная задача плохо определена.
- *Параметры:* Размер якоря, матрица сходства, штраф за делецию

Выравнивания

Редакционное расстояние

- ***Элементарное преобразование последовательности:*** замена буквы или удаление буквы или вставка буквы.
- ***Редакционное расстояние:*** минимальное количество элементарных преобразований, переводящих одну последовательность в другую.
- Формализация задачи сравнения последовательностей: найти редакционное расстояние и набор преобразований, его реализующий

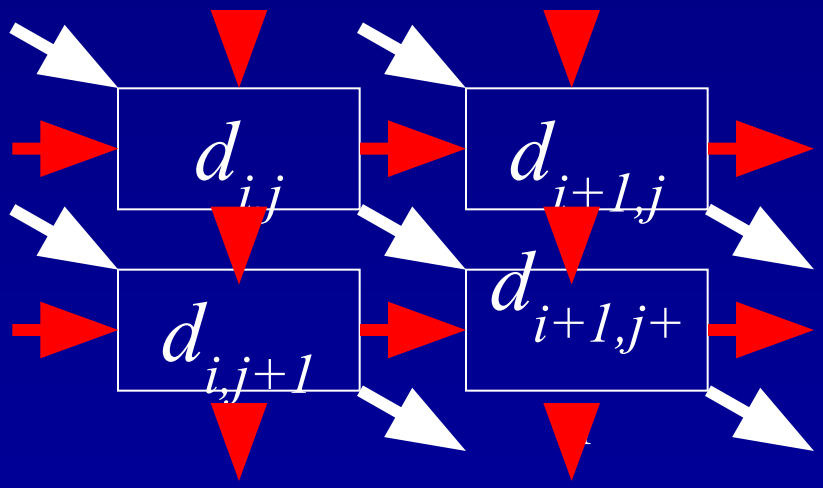
Сколько существует выравниваний?

- Дано: две последовательности S_1 и S_2 длиной m и n. Сколько есть способов написать одну последовательность под другой (со вставками)?
- Построим выборочную последовательность S длиной $m+n$ следующим образом: возьмем несколько символов из последовательности S_1, потом несколько символов из последовательности S_2 потом опять несколько символов из S_1, потом опять несколько из S_2.
  - Каждой выборочной последовательности S соответствует выравнивание и по каждому выравниванию можно построить выборочную последовательность. ***(Доказать!)***
  - Количество выборочных последовательностей равно $N_{sel} = C_{n+m}^{m} = (m+n)! / (m! \cdot n!)$ ***(Доказать!)***

$$N_{align} = \frac{(2n)!}{n!^2} = 2\sqrt{\frac{\pi}{n}}2^{2n}$$

Динамическое программирование для редакционного расстояния

- Граф редакционного расстояния для последовательностей S^1,S^2: вершина $v_{i,j}$ соответствует префиксам последовательностей $\{S^1_{1..i}\}$, $\{S^2_{1..j}\}$. На вершине записано редакционное расстояние между префиксами.
(*красные стрелки соответствуют вставкам и удалениям*)



$$d_{i+1,j+1} = \mathbf{min}\{\, d_{i+1,j}+1,\ d_{i,j+1}+1,\ d_{i,j}+e_{i,+1,j+1}\}$$

$$e_{i,j} = \{\ 0,\ S^1_i = S^2_j\ ;\ 1,\ S^1_i \neq S^2_j\ \}$$

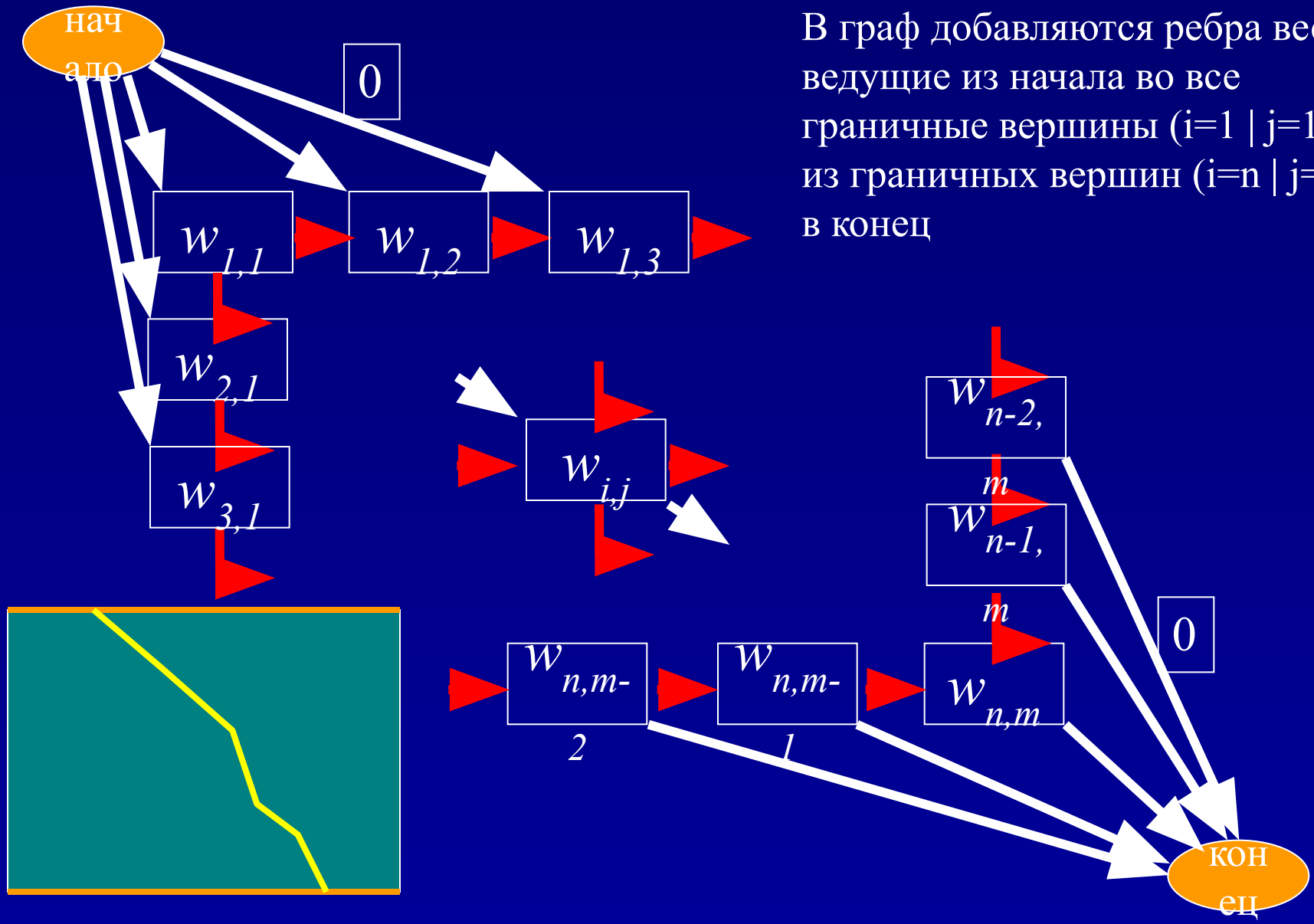
Подмена задачи и обобщение

- Заменим расстояния $d_{i,j}$ на $-d_{i,j}$. Тогда операцию **min** надо заменить на **max.**
- Прибавим к $-d_{i,j}$ ½ ($w_{i,j} = ½ - d_{i,j}$), тогда получим функцию сходства: совпадение = ½, замена = –½, делеция = –1.
- Функцию сходства W легко обобщить, варьируя штрафы за замену и делеции.
- Новая задача: написать одну последовательность под другой так, чтобы максимизировать сходство
- Алгоритм Нидлмана – Вунша решает эту задачу, используя динамическое программирование.

Граничные условия

При таких граничных условиях начальные и концевые делеции штрафуются

Как не штрафовать за концевые делеции



В граф добавляются ребра веса 0, ведущие из начала во все граничные вершины (i=1 | j=1) и из граничных вершин (i=n | j=m) в конец

Алгортим Нидлмана – Вунша: оценка времени работы и необходимой памяти

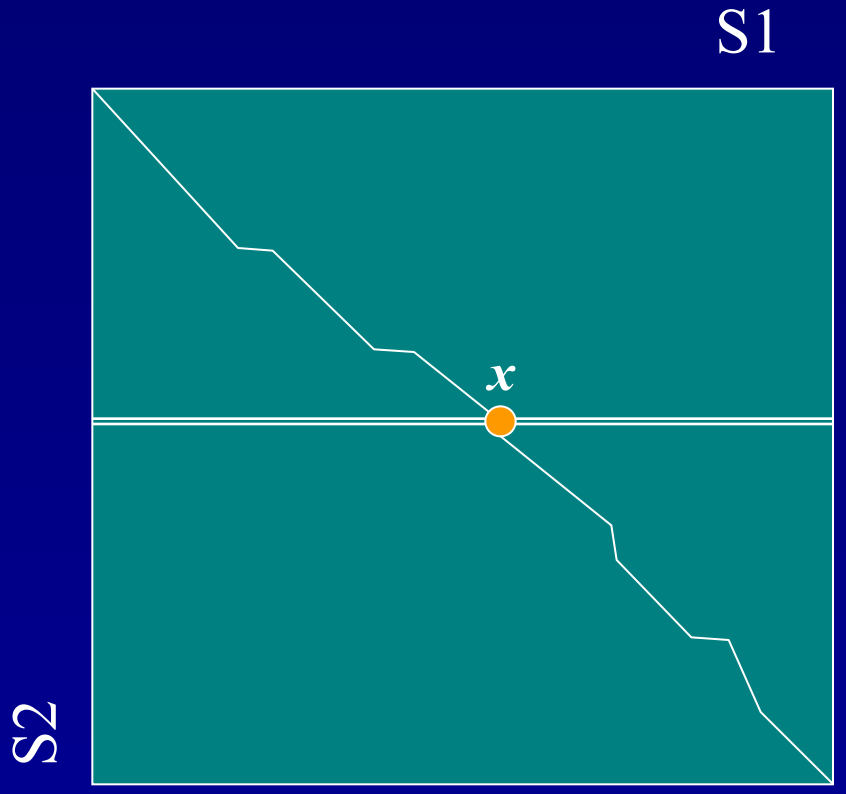
- Алгоритм просматривает все вершины графа
- В каждой вершине делается 3 сравнения
- Количество необходимых операций (время работы алгоритма): $T=O(n \cdot m)$. Говорят, что алгоритм выравнивания квадратичен по времени работы.
- Для запоминания весов и восстановления оптимального выравнивания надо в каждой вершине запомнить ее вес и направление перехода. Таким образом, алгоритм квадратичен по памяти.

Где можно сэкономить?

- Во-первых не обязательно запоминать веса во всех вершинах. При просмотре матрицы выравнивания (графа выравнивания) можно идти по строкам. При этом нам необходима только предыдущая строка.

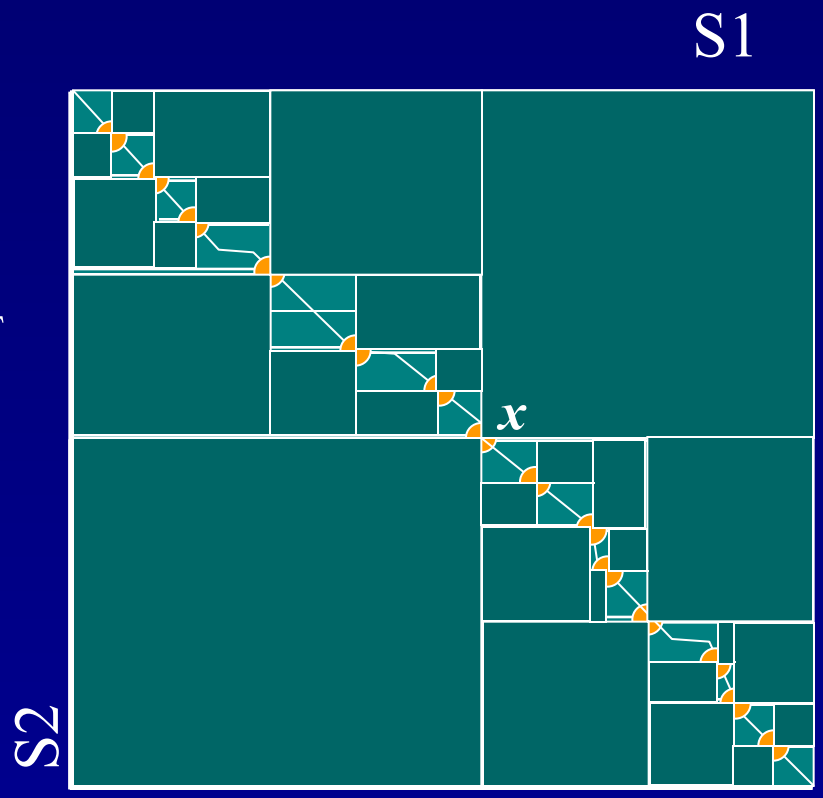
Линейный по памяти алгоритм Миллера – Маерса

- Разбиваем одну из последовательностей на две равные части
- Для каждой точки x линии раздела находим веса оптимальных выравниваний из начала в x и из конца в x:
  $W^+(x)$, $W^-(x)$
- Вес оптимального выравнивания, проходящего через точку x равен
  $W(x)=W^+(x) + W^-(x)$
- Вес оптимального выравнивания равен
  $W = \mathbf{max}_x\ (W(x))$
- Таким образом, за время $T=C\cdot n^2$ найдена одна точка, через которую проходит оптимальное выравнивание



Алгоритм Миллера – Маерса

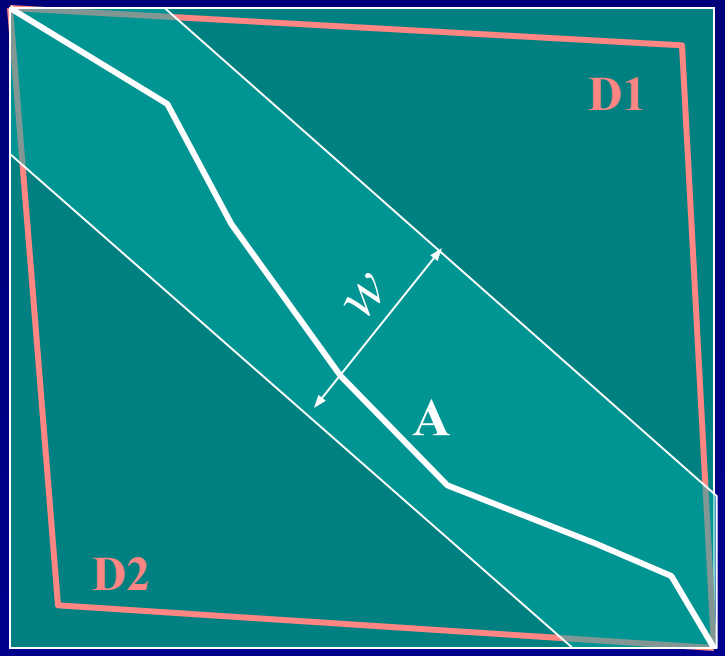
- Найденная точка *x* разбивает матрицу выравнивания на четыре квадранта, два из которых заведомо не содержат оптимального выравнивания
- Для двух квадрантов, содержащих оптимальный путь можно применить тот же прием, и запомнить точки *x'* и *x''*.
- Просмотр оставшихся квадрантов требует времени $T=Cn^2/2$ (почему?)
- Продолжая процедуру деления пополам найдем все точки, через которые проходит оптимальный путь.
- Время работы алгоритма
  $T=Cn^2+Cn^2/2+Cn^2/4+\ldots=$
  $= Cn^2(1+1/2+1/4+1/8+\ldots);$
  $\boldsymbol{T=2C\cdot n^2}$



Важно, что при просмотре мы не запоминали обратных переходов!

Еще один способ сэкономить время и память

- Ясно, что выравнивания D1 и D2 не представляют интереса, поскольку содержат в основном делеции
- Разумные выравнивания (A) лежат в полосе
- Алгоритм: задаемся шириной полосы w и просматриваем только те вершины графа, что лежат в указанной полосе.

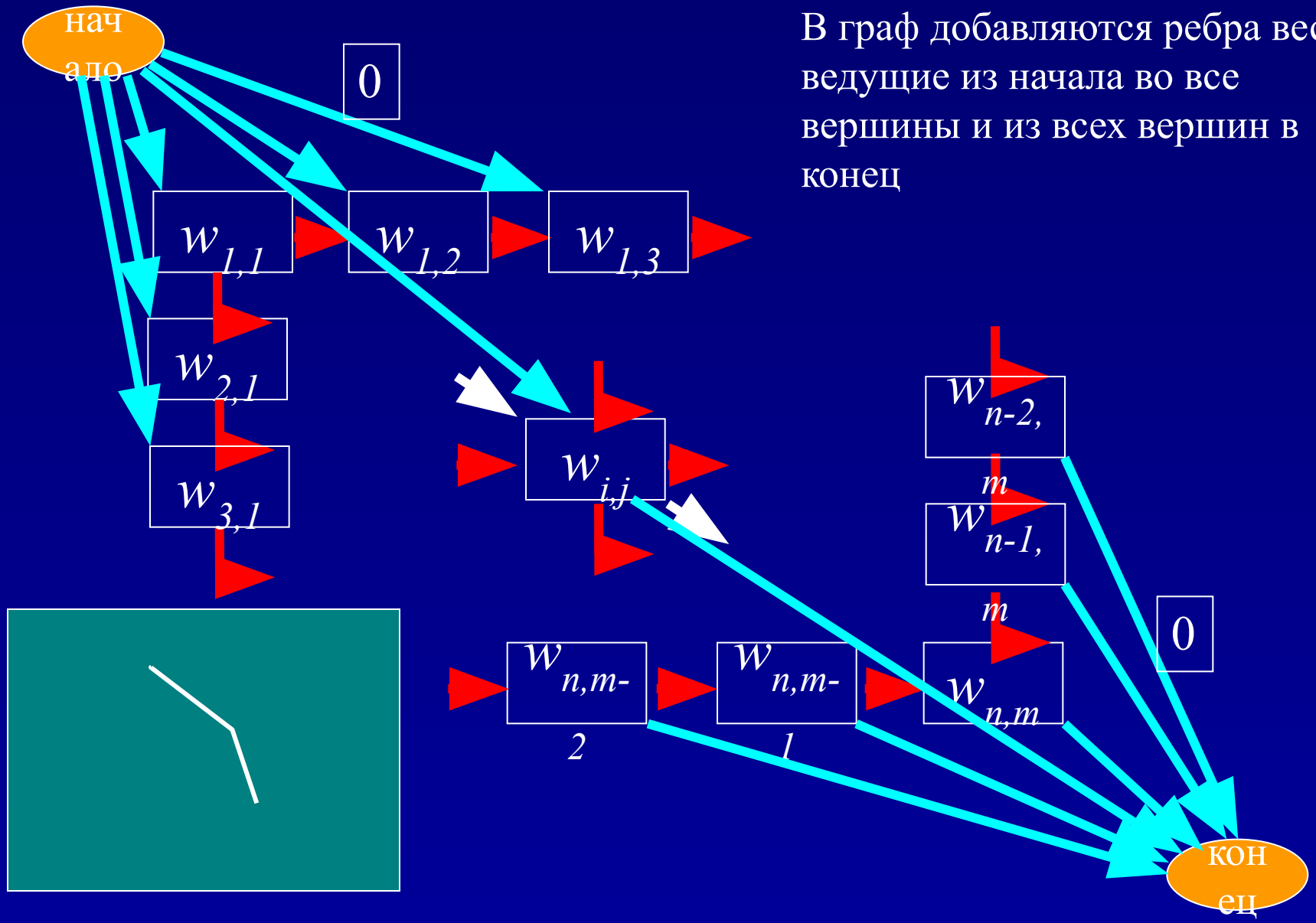


Локальное выравнивание

- Локальным оптимальным выравниванием называется такое оптимальное выравнивание фрагментов последовательностей, при котором любое удлинение или укорочение фрагментов приводит только к уменьшению веса
- Локальному оптимальному выравниванию отвечает путь с наибольшим весом, независимо от того, где он начинается и где кончается
- Локальное оптимальное выравнивание может иметь бóльший биологический смысл, чем глобальное, но только если математическое ожидание веса сравнения букв, случайно взятых из последовательностей, отрицательно (почему?)

  Например, для алфавита из 4 букв, встречающихся с одинаковой частотой, годятся параметры 1 за совпадение, –1 за замену или 5 за совпадение, –2 за замену, но не имеет смысла использовать 5 за совпадение и –1 за замену.

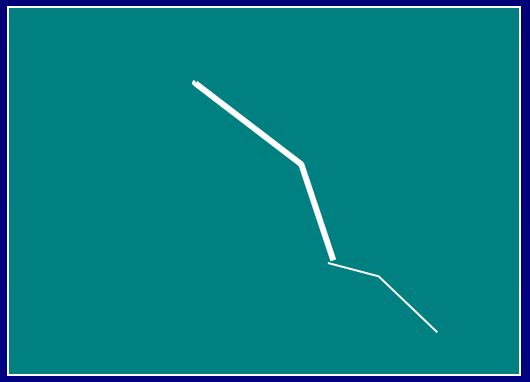
Алгоритм Смита – Ватермана

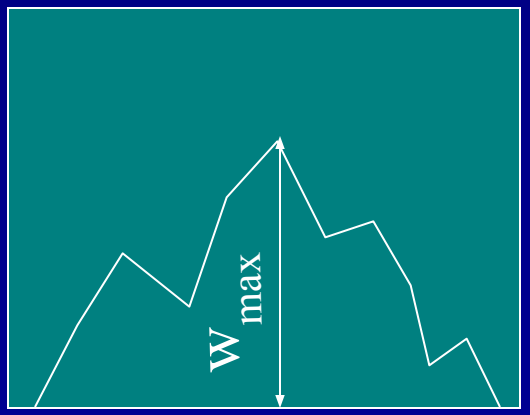


В граф добавляются ребра веса 0, ведущие из начала во все вершины и из всех вершин в конец

Алгоритм Смита – Ватермана

- Пусть есть какой-то путь с неотрицательными весами
- Построим график веса вдоль пути
- Абсолютный максимум на этом графике определит точку окончания пути





Алгоритм Смита – Ватермана

$$w_{i,j} = \mathbf{max} \{ w_{i\text{-}i,j\text{-}1} + e_{i,j},\ i > 1, j > 1$$
$$w_{i\text{-}1,j} - d\ ,\ i > 1$$
$$w_{i,j\text{-}1} - d\ ,\ j > 1$$
$$\mathbf{0} \}$$

- Точка конца пути (от нее начинаем обратный просмотр и восстановление пути) определяется так:

$$(i_{max}, j_{max}) = \mathbf{argmax}\ (w_{i,j})$$

Пусть (при одинаковых параметрах) мы получили вес глобального выравнивания S_{glob} и вес локального выравнивания S_{loc}. Какая величина больше?

Более общая зависимость штрафа за делецию от величины делеции

- Простейшая модель делеции: элементарное событие – удаление ***одного*** символа. Протяженная делеция – несколько независимых событий удаления одного символа. Работает плохо.
- Более реалистичная модель: делеция нескольких символов происходит за одно элементарное событие, а размер делеции является некоторой случайной величиной. Поэтому в качестве штрафа хорошо бы взять что-нибудь вроде

  $\Delta (l) = a \log(l + 1)$, где l – длина делеции

  В любом случае функция $\Delta (l)$ должна быть выпуклой – должно выполняться неравенство треугольника:

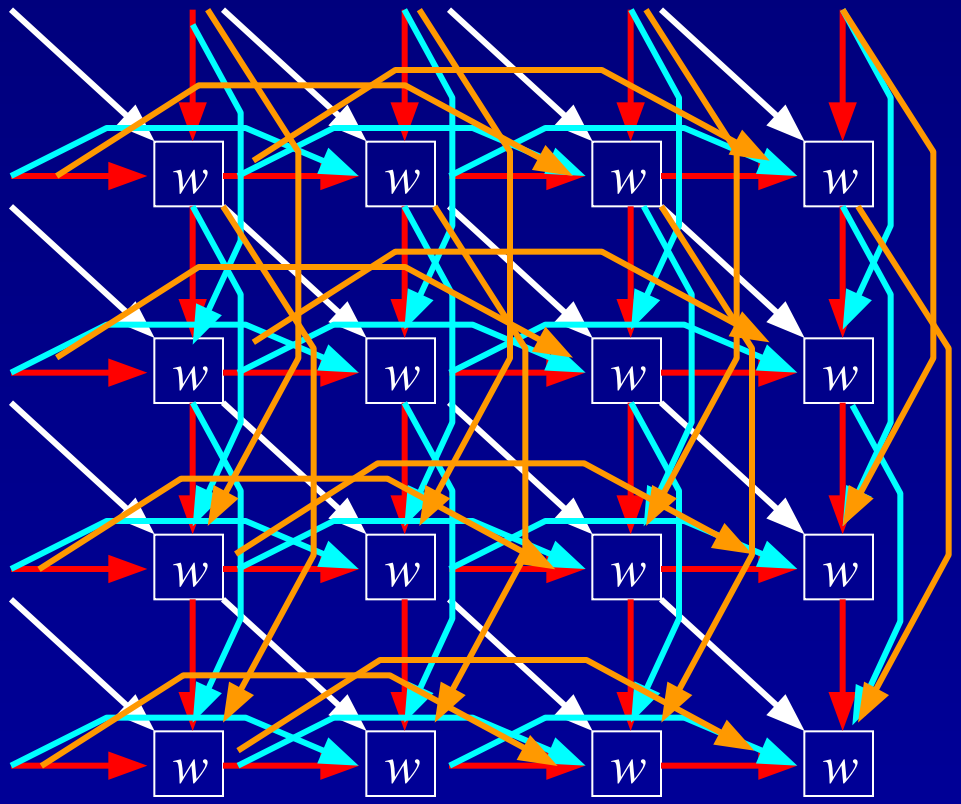
  $$\Delta (l_1 + l_2) \le \Delta (l_1) + \Delta (l_2)$$

Более общая зависимость штрафа за делецию от величины делеции. Алгоритм.

Теперь надо просматривать все возможные варианты делеций. Поэтому в каждую вершину входит не 3 ребра, а примерно $(n+m)/2$ ребер, где n, m – длины последовательностей

Поэтому время работы алгоритма становится кубичным:

$$T = O\,(\,nm\,(n+m)\,)$$

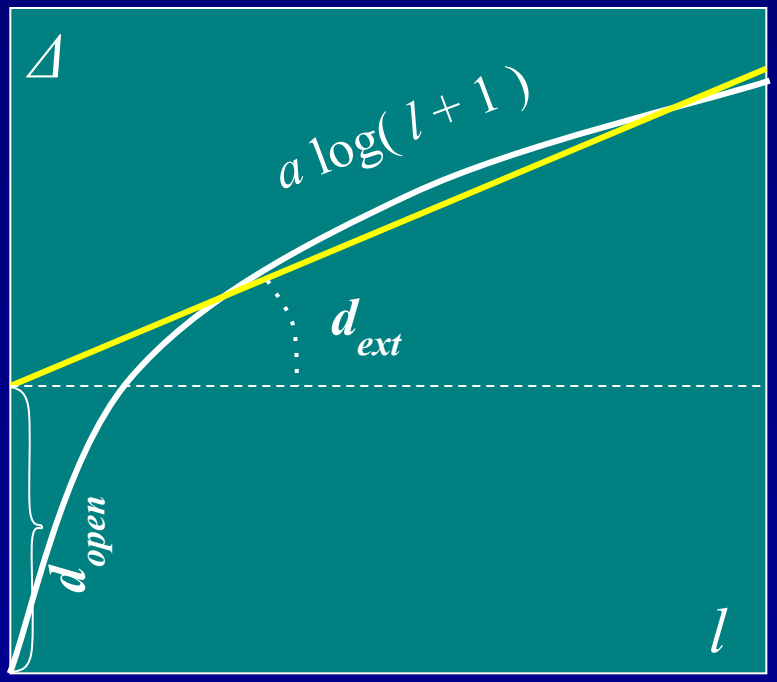


Аффинные штрафы за делецию

- Вместо логарифмической зависимости используют зависимость вида:

$$\Delta(l) = d_{open} + l\, d_{ext}$$

- d_{open} – штраф за открытие делеции
- d_{ext} – штраф за длину делеции



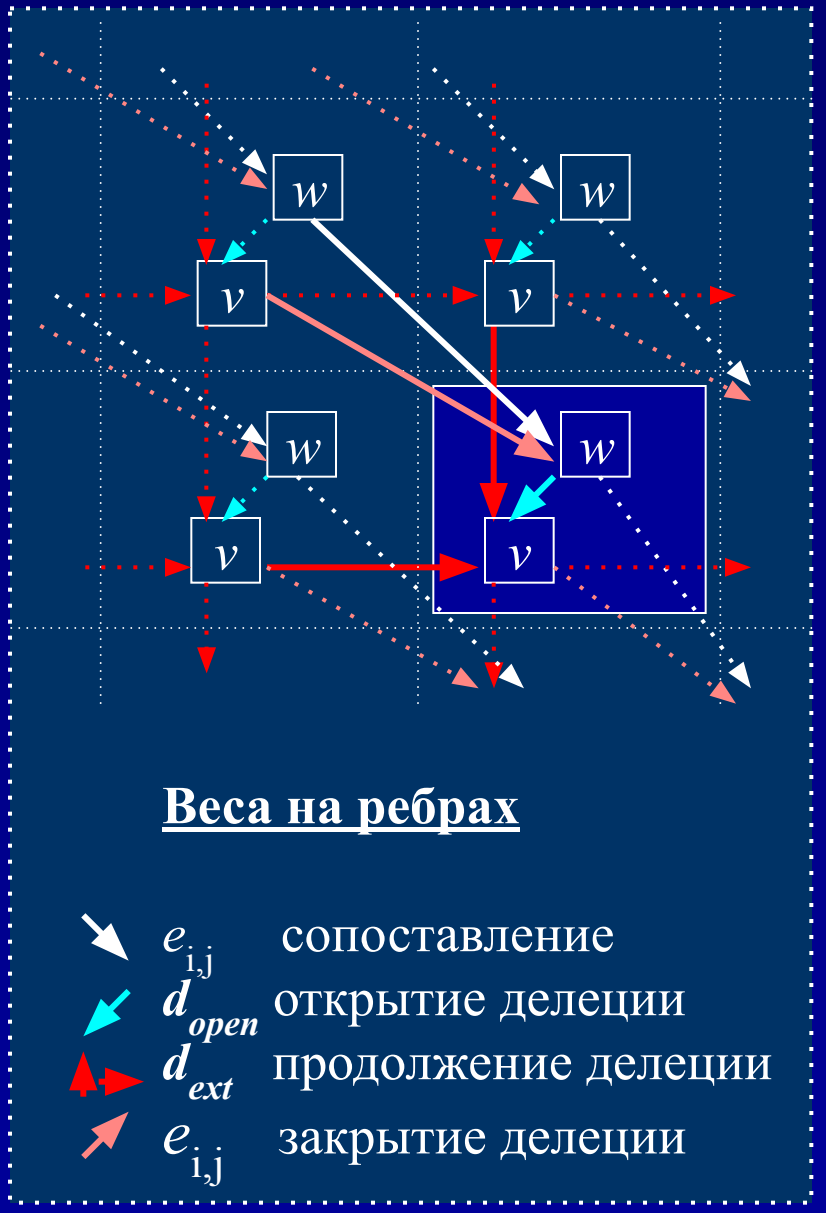
Алгоритм для аффинных штрафов

Модификация стандартного графа:

1. В каждой ячейке вводится дополнительная вершина (v), отвечающая делеционному пути
2. Вводятся делеционные ребра для открытия и закрытия делеции (из вершин типа w в вершины типа v и обратно)
3. Ребра, отвечающие продолжению делеции переносятся на новые вершины

Число вершин графа равно $2mn$
число ребер равно $5mn$

Трудоемкость алгоритма равна:
$T = O(mn)$



Рекурсия для аффинных штрафов

- $w_{i,j} = \mathbf{max}\,(\, w_{i-1,j-1} + e_{ij},\ v_{i-1,j-1} + e_{ij},\ 0\,)$;
- $v_{i,j} = \mathbf{max}\,(\, w_{i,j} - d_{\text{open}},\ v_{i-1,j} - d_{\text{ext}},\ v_{i,j-1} - d_{\text{ext}}\,)$;
- $(i_{max}, j_{max}) = \mathbf{argmax}\,(w_{i,j})$

Статистика выравниваний

Параметры выравнивания

- В простейшем случае есть три параметра:
  - премия за совпадение (*match*)
  - штраф за несовпадение (*mism*)
  - штраф за делецию (*indel*)
- Если все параметры умножить на одну и ту же положительную величину, то само оптимальное выравнивание не изменится, а вес выравнивания умножится на ту же величину. Поэтому можно положить ***match*=1.**
- Если $mism > 2 \cdot indel$, то оптимальное выравнивание не будет иметь замен (почему?)

Статистика выравниваний

- Допустим мы выровняли две последовательности длиной 100 и получили вес 20. Что это значит? Может быть при выравнивании двух случайных последовательностей будет тот же вес?
- А что такое случайные последовательности?

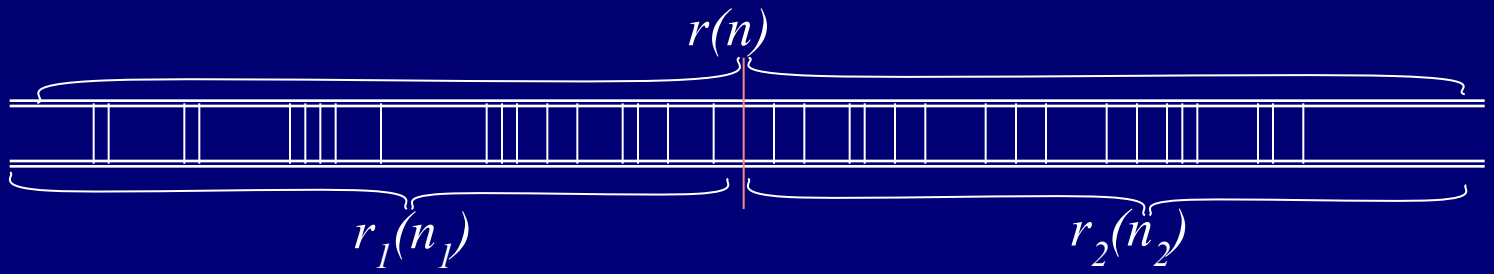
Модели случайных последовательностей

- Базовая (вообще говоря неправильная) модель – бернуллиевские последовательности: символы генерируются независимо друг от друга с заданной вероятностью. Для этой модели математика проще и проще получить оценки.
- Уточненная модель (лучше, но тоже неправильная) – марковская цепь (вероятность появления следующего символа зависит от нескольких предыдущих символов). Математика значительно сложнее. Почти ничего не известно.

Частные случаи локального выравнивания

- $mism = 0$, $indel = 0$ – максимальная общая подпоследовательность
- $mism = \infty$, $indel = \infty$ – максимальное общее подслово

Наибольшая общая подпоследовательность



- Длина общей подпоследовательности есть случайная величина $r(n)$, зависящая от длины последовательностей.
- Пусть две последовательности длиной n разбиты каждая на два фрагмента длиной n_1 и n_2 ($n_1 + n_2 = n$)
- Ясно, что общая подпоследовательность будет не короче, чем объединение общих подпоследовательностей для фрагментов:
  $r(n) \geq r_1(n_1) + r_2(n_2)$ *(попробуйте понять смысл неравенства)*
- Отсюда следует, что математическое ожидание
  $E(r(n)) \geq E(r(n_1)) + E(r(n_2))$, или $E(r(n)) \geq c \cdot n$
- Можно показать, что
  $$E(r(n)) - \left(E(r(n_1)) + E(r(n_2))\right) \to 0$$
- Поэтому:

$$\boldsymbol{E(r(n)) \approx c \cdot n \ (n \to \infty)}$$

Наибольшее общее слово

- Наложим одну последовательность на другую. Будем идти вдоль пары последовательностей и, если буквы совпали, то будем считать успехом, иначе – неудача. Имеем классическую схему испытаний Бернулли. Наибольшему общему слову при таких испытаниях будет соответствовать максимальная серия успехов. Известно, что средняя величина максимальной серии успехов равна:

  $E(l) = \log_{1/p}(n)$

- Возможных наложений много (порядка длины последовательности). Максимальное общее слово есть максимум от максимальных серий успехов при всех возможных наложениях. Показано (*Waterman*), что:

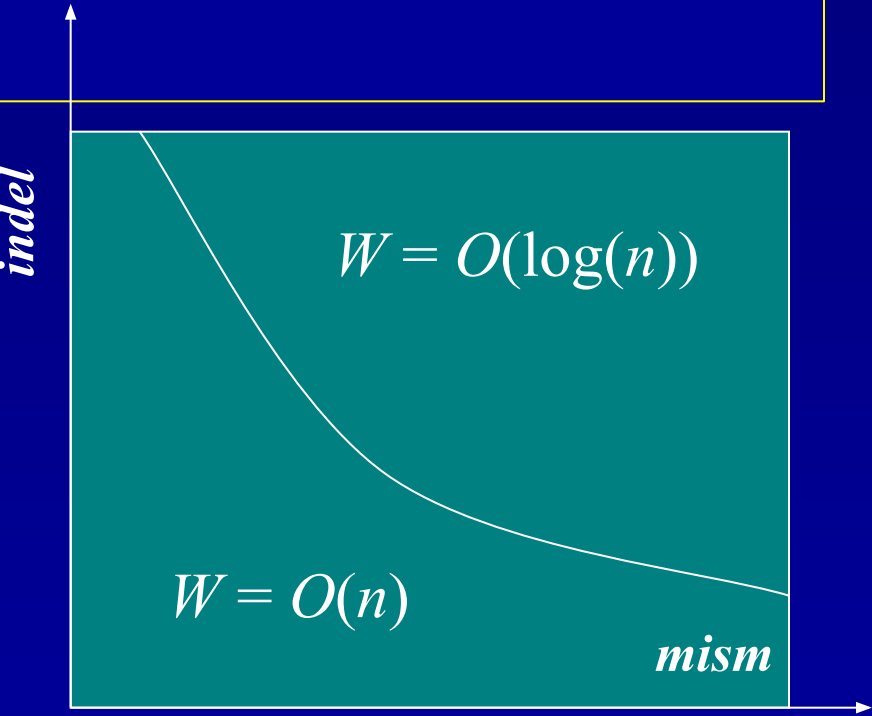
$$\boldsymbol{E(l) \approx \log_{1/p}(nm) + \log_{1/p}(1\text{-}p) + \gamma \cdot \log_{1/p}(e) - \tfrac{1}{2} = \log_{1/p}(Knm),} \qquad (m, n \to \infty,\ \gamma \approx 0{,}577)$$

$$\boldsymbol{\sigma(l) \approx [\ \pi \log_{1/p}(e)\ ]^2 / 6 + \tfrac{1}{2},} \quad \textit{(не зависит от n !)}$$

Зависимость от параметров

- Показано, что зависимость ***ожидаемого*** веса выравнивания от длины последовательности может быть либо логарифмической, либо линейной в зависимости от параметров. Все пространство параметров разбивается некой поверхностью на две области поведения.

При безделеционном выравнивании поведение логарифмическое, если мат. ожидание ве́са сравнения двух случайных сегментов отрицательно.



Матрицы замен

Откуда берутся параметры для выравнивания?

- Пусть у нас есть выравнивание. Если последовательности случайные и независимые (модель R), то вероятность увидеть букву α против β

$$p(\alpha, \beta \mid R) = p(\alpha)\, p(\beta)$$

а вероятность выравнивания (x,y) будет равна

$$p(x,y \mid \mathrm{R}) = \prod p(x_i) \prod p(y_i)$$

Если выравнивание не случайно (модель M), то

$$p(x,y \mid \mathrm{M}) = \prod p(x_i, y_i)$$

Отношение правдоподобия:

$$\frac{p(x,y \mid \mathrm{M})}{p(x,y \mid \mathrm{R})} = \frac{\prod p(x_i, y_i)}{\prod p(x_i) \prod p(y_i)}$$

Логарифмируя, получаем

$$\log\left(p(x,y|\mathrm{M})/p(x,y|\mathrm{R}) \right) = \sum s(x_i, y_i);$$

Матрица замен: $\boldsymbol{s(\alpha, \beta) = \log(p_{\alpha\beta} / p_\alpha p_\beta)}$

Серия матриц BLOSUM

- База данных BLOCKS (*Henikoff & Henikoff*) – безделеционные фрагменты множественных выравниваний (выравнивания получены специальной программой).
- В каждом блоке отбираем подмножество последовательностей такое, что для каждой пары в нём процент идентичных аминокислот не больше заданного значения ID.
- В урезанном блоке в каждой колонке подсчитываем число пар аминокислот

$$n^{bl}_{col}(\alpha, \beta)$$

- Усредняем по всем колонкам и по всем блокам:

$$f(\alpha, \beta) = \sum n^{bl}_{col}(\alpha, \beta) / N_{col}$$

- Элемент матрицы $BLOSUM_{ID}$:

$$\mathbf{BLOSUM}_{\mathbf{ID}}(\alpha, \beta) = \log(f(\alpha, \beta) / f(\alpha) f(\beta))$$

Серия матриц PAM

- **P**oint **A**ccepted **M**utation – эволюционное расстояние, при котором произошла одна замена на 100 остатков.
- Эволюционный процесс можно представить как марковский процесс. Если в начальный момент времени $t = 0$ в некоторой позиции был остаток α, то через время Δt в этой позиции с некоторой вероятностью будет остаток β:

$$\mathbf{p(\beta|\ \alpha, \Delta t) = M_{\Delta t}(\beta, \alpha)}$$

$M_{\Delta t}$ – эволюционная матрица

Через время $2 \cdot \Delta t$

$$\mathbf{p(\beta|\ \alpha, 2 \cdot \Delta t) = \sum_{\gamma} M_{\Delta t}(\beta, \gamma) \cdot M_{\Delta t}(\gamma, \alpha) = M_{\Delta t}^{2}(\beta, \alpha)}$$

Через время $N \cdot \Delta t$

$$\mathbf{p(\beta|\ \alpha, N \cdot \Delta t) = M_{\Delta t}^{N}(\beta, \alpha)}$$

Серия матриц PAM

Находим выравнивания, отвечающие расстоянию PAM1

Находим частоты пар и вычисляем частоты пар. Поскольку расстояние мало, то один из символов соответствует предку:

$$p(\alpha\beta) = p(\alpha \rightarrow \beta)\, p(\alpha) + p(\beta \rightarrow \alpha)\, p(\beta)$$

полагая эволюцию равновесной:

$$p(\alpha \rightarrow \beta)\, p(\alpha) = p(\beta \rightarrow \alpha)\, p(\beta)$$

получаем

$$p(\alpha \rightarrow \beta) = 2p(\alpha\beta) / p(\alpha)$$

$$p(\alpha \rightarrow \alpha) = 1 - \sum\nolimits_{\beta \neq \alpha} p(\alpha \rightarrow \beta)$$

Марковский процесс:

$$p_N(\alpha\beta) = p^N(\alpha \rightarrow \beta)\, p(\alpha)$$

$$\mathbf{PAM}_N(\alpha\beta) = \log\left(p^N(\alpha \rightarrow \beta) / p_\beta\right)$$

Распределение экстремальных значений

- Пусть вес выравнивания x (случайная величина) имеет распределение

$$G(S) = P(x < S)$$

- Тогда при N независимых испытаниях распределение максимального значения будет

$$G_N(x) = G^N(x);$$

- Можно показать, что для нормально распределенного $G(x)$ при больших N

$$\boldsymbol{G_N(x) \approx \exp(-KN\, e^{-\lambda x})}$$

E-value и P-value

- Для бернуллиевских последовательностей длин m и n математическое ожидание количества независимых локальных выравниваний с весом $>S$ описывается формулой (*Karlin &Altschul*) :

$$E(S) = Kmn\, e^{-\lambda S}$$

где λ – положительный корень уравнения $\sum_{\alpha\beta} \boldsymbol{p_\alpha p_\beta}\, e^{\lambda s(\alpha\beta)} = 1$
$\boldsymbol{s(\alpha\beta)}$ – матрица замен (с отрицательным матожиданием сравнения случайных букв: $\sum_{\alpha\beta} \boldsymbol{p_\alpha p_\beta}\, \boldsymbol{s(\alpha\beta)} < 0$)
K – константа, зависящая от $\boldsymbol{p_\alpha}$ и $\boldsymbol{s(\alpha\beta)}$
Для поиска по банку n – суммарная длина банка.

- E-value: $E(S)$
  ожидаемое количество выравниваний с таким или большим весом
- P-value: $p(x > S) = 1 - e^{-E(S)}$
  вероятность встретить (хоть одно) выравнивание с таким или большим весом

Проблема малой сложности

- В формулах для E-value и P-value присутствуют вероятности букв p_α
- Эти формулы работают удовлетворительно, если частоты букв в последовательностях примерно равны вероятностям букв в модели
- Но в биологических последовательностях встречаются участки, в которых частоты букв сильно “сдвинуты” (например, богатые пролином петли в белках). Выравнивание таких участков при использовании стандартной матрицы $s(\alpha\beta)$ получит высокий вес даже при отсутствии реальной гомологии.
- Близкая проблема: участки вида

  … atgcatgcatgcatgcatgcatgcatgc …

Проблема малой сложности: подходы к решению

- «Маскировка» участков малой сложности:
  - применялась в BLAST до 2005 года как единственный вариант; сейчас применяется в основном для нуклеотидных последовательностей;
  - для белковых последовательностей применяется программа **seg**, маскирующая участки с частотами букв, сильно отличающимися от «базовых»;
  - для нуклеотидных последовательностей применяется программа **dust**, которая маскирует участки с сильно «сдвинутым» составом триплетов (подслов длины 3);
  - основной недостаток — «всё или ничего»: можно замаскировать биологически осмысленное выравнивание, с другой стороны участок чуть выше «порога сложности» может дать много бессмысленных выравниваний с последовательностями банка.
- Корректировка матрицы замен $s(\alpha\beta)$ в соответствии с частотами букв в последовательностях:
  - предложена в статьях Yu (Юй) и Альтшуля 2003–2005;
  - с 2005 включена по умолчанию в BLAST для белков.

Алгоритм dust

На входе: нуклеотидная последовательность (в алфавите A, T, G, C).
На выходе: маскированная последовательности, отличающаяся от исходной тем, что участки малой сложности заменены буквами N.

Для каждой подпоследовательности a и для каждого триплета (слова длины 3) t определяется число $c_t(a)$, равное числу подслов в a, совпадающих с t.

Вес $S(a)$ определяется как $(n-3)^{-1} \cdot \sum_t c_t(a) \cdot (c_t(a) - 1)/2$, где n – длина a.
При более или менее равных частотах триплетов вес мал, а если какие-то триплеты встречаются сильно чаще других, велик (почему?).

Алгоритм:
– рассматриваются все подпоследовательности a длины W=64
– для каждого a находится префикс a' максимального веса $S(a')$
– если $S(a') > T = 2$, то в a' находится суффикс a'' максимального веса $S(a'')$
– все найденные так подпоследовательности a'' маскируются.

Алгоритм seg

Вход: белковая или нуклеотидная последовательность.
Выход: маскированная последовательность (нуклеотиды в участках малой сложности заменяются на “N”, аминокислоты — на “X”).

В алгоритме SEG используется две меры сложности последовательности:

- $K_1 = 1/L \cdot \log_N \Omega$

здесь N – размер алфавита (4 или 20), L – длина последовательности, а Ω равно числу различных последовательностей, имеющих в точности те же частоты букв, что и данная.
Чем больше частоты букв различаются между собой, тем меньше K_1 (почему?)

- $K_2 = \sum_\alpha f(\alpha) \log_2 f(\alpha)$, где f – частоты букв.

K_2 представляет собой приближение K_1 : $K_2 \to K_1$ при $L \to \infty$

Алгоритм seg (окончание)

- Проходим по последовательности окном длины W (по умолчанию $W = 12$) и определяем сложность (K_2) для каждого окна.
- Помечаем все буквы, попавшие в окна, чья сложность ниже первого порога, по умолчанию равного 2,2.
- Помечаем все буквы, попавшие в окна, пересекающиеся с уже помеченными участками и чья сложность ниже второго порога, по умолчанию равного 2,5.
- Для каждого непрерывного отрезка из помеченных букв уточняем границы участка малой сложности, используя меру K_1 .

Корректировка матрицы замен

Матрица замен (BLOSUM или PAM) может быть представлена в виде:
$s(\alpha\beta) = \lambda^{-1} \cdot \log_2(f(\alpha, \beta) / f(\alpha) f(\beta))$,
где $f(\alpha, \beta)$ – частоты замен в эталонных выравниваниях, а $f(\alpha)$ – частоты букв $f(\alpha) = p_\alpha$, вероятностям бернуллиевской модели в формуле для E-value

Имеется очевидное равенство: $\sum_\alpha f(\alpha, \beta) = f(\beta)$ для любой буквы β

Пусть теперь имеется пара последовательностей, в которых частоты букв $f'(\alpha) \neq f(\alpha)$.

Задача: подобрать $f'(\alpha, \beta)$ так, чтобы:
а) $\sum_\alpha f'(\alpha, \beta) = f'(\beta)$, т.е. частоты замен соответствовали частотам букв;
б) матрица $f'(\alpha, \beta)$ была (в каком-нибудь смысле) «самой близкой» к матрице $f(\alpha, \beta)$ изо всех матриц, удовлетворяющих условию (а).

Когда $f'(\alpha, \beta)$ подобраны, для выравнивания используется матрица замен
$s'(\alpha\beta) = \lambda^{-1} \cdot \log_2(f'(\alpha, \beta) / f'(\alpha) f'(\beta))$

Какая матрица самая близкая?

Используется один из двух вариантов.

Вариант 1. Самая близкая матрица $f'(\alpha, \beta)$ та, для которой минимальна величина

$$D = \sum_{\alpha\beta} f'(\alpha, \beta) \log(f'(\alpha, \beta) / f(\alpha, \beta))$$

Вариант 2. Самая близкая матрица та, для которой, во первых, выполнено равенство:

$$\sum_{\alpha\beta} f'(\alpha, \beta) \log(f'(\alpha, \beta) / f'(\alpha) f'(\beta)) = \sum_{\alpha\beta} f(\alpha, \beta) \log(f(\alpha, \beta) / f(\alpha) f(\beta))$$

а во-вторых, величина D минимальна среди всех матриц, для которых выполнено это равенство.

Именно второй вариант реализован в программе BLASTP.

Смысл равенства в том, что величина в правой и левой его части («энтропия матрицы») характеризует отличие частот пар в выравниваниях от произведений частот букв.

Для матриц PAM энтропия уменьшается с ростом номера (для PAM10 она больше, чем для PAM100), для матриц BLOSUM — наоборот (почему?)

Поиск по банку

Поиск по банку. Постановка задачи

- На входе: последовательность-запрос (query) и банк из большого количества последовательностей.
- Нужно найти в банке кандидатов в гомологи запроса.
- Кандидатами в гомологи считаются последовательности с достаточно высоким весом локального выравнивания с запросом.
- Достаточно ли высок вес, оценивается по P-value или E-value.
- Главная проблема – время: алгоритм Смита – Ватермана слишком медленный. Нужны приёмы быстрого выравнивания.
- Нахождение всех достаточно хороших выравниваний не гарантируется (эвристические алгоритмы).

Поиск по банку. Хеширование.

- Подготовка банка – построение хэш-таблицы. Хэш-функция – номер слова заданного размера (***l-tuple***, ***l-грамма***).
- В хэш-таблице хранятся списки ссылок на последовательности и на позиции в последовательностях, где встречается соответствующая *l*-грамма.
- При поиске запроса (query) в последовательности запроса последовательно находятся *l*-граммы, далее, по хэш-таблице для них находятся соответствующие документы и позиции.
- Пара совпадающих *l*-грамм в запросе и в банке называется ***затравкой***, ***якорем***, ***seed***.

Поиск по банку. BLAST1.

- Ищем якоря с помощью хэш-таблицы. Длина якоря: $l = 3$ или 4 для белков, $l \geq 7$ для н.к.
- Каждый якорь расширяем с тем, чтобы получить сегмент совпадения наибольшего веса (HSP – high scoring pair).
- Оцениваем его статистическую значимость (E-value), и, если она больше порога, то выдаём

*(**Altschul** , Gish, Miller, Myers, Lipman, 1990)*

Поиск по банку. BLAST2.

- Расширяются не одиночные якоря, а пары якорей, которые находятся недалеко и на близких диагоналях
- При расширении пары якорей допускаются делеции
- *T*-соседней *l*-граммой $\boldsymbol{L}^{\boldsymbol{T}}$ для *l*-граммы ***L*** называется такая *l*-грамма, что вес ее сравнения с ***L*** не меньше заданного ***T***:

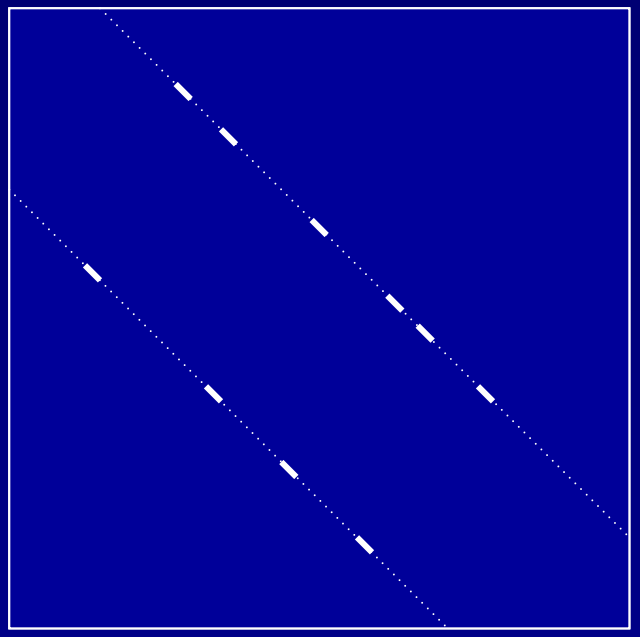
  $$\sum s(L_i, L_i^T) \geq T$$

  Для аминокислотных последовательностей при просмотре запроса формируем не только те *l*-граммы, которые встретились в нем, но также все *T*-соседние *l*-граммы.
- Используются $l = 2$ или 3 для белков, $l \geq 4$ для н.к.

(*Altschul et al., 1997)*

Поиск по банку. FASTA.

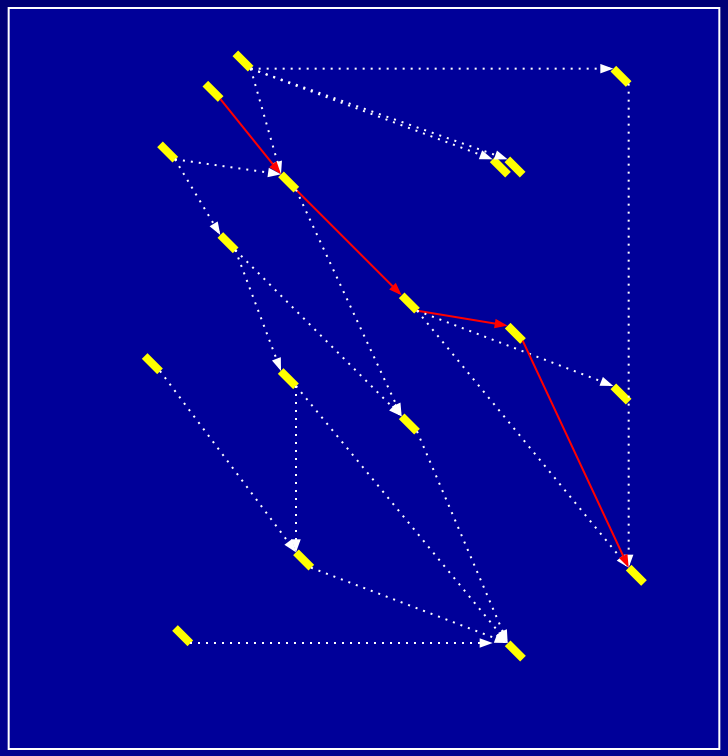
- Используются якоря длины $l = 1$ или 2 (для белков); $l = 3, \ldots, 6$ (для нк).
- Два якоря (i_1,j_1), (i_2,j_2) принадлежат одной диагонали, если
  $i_1 - j_1 = i_2 - j_2$
- Мощностью диагонали называется количество якорей, принадлежащих диагонали. Иногда в мощность диагонали включают мощности соседних диагоналей (чтобы учесть возможность небольших делеций)
- Отбираем n (например $n = 10$) самых мощных диагоналей и для каждой строим локальное выравнивание в полосе заданной ширины вокруг диагонали



(Wilbur, Lipman, Pearson)

Ещё один алгоритм быстрого выравнивания

- Ищем якоря
- Якорь (i_1,j_1) предшествует якорю (i_2,j_2), если
  $i_1 < i_2$ & $j_1 < j_2$
  & $i_2 - i_1 < d$ & $j_2 - j_1 < d$
- Получаем ориентированный граф с небольшим количеством вершин и ребер
- Можно найти оптимальную цепочку якорей методом динамического программирования



Введение в байесову статистику

и некоторые дополнительные сведения из математики

δ-функция

- **Определение:**

$$\delta(x) = 0,\ x \neq 0;$$

$$\int_{-\infty}^{+\infty} \delta(x)\, dx = 1$$

- **Свойство:**

$$\int_{-\infty}^{+\infty} \delta(x - a)\, f(x)\, dx = f(a)$$

- **Символ Кронекера. Определение:**

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

Г-функция

- **Определение:**

$$\Gamma(x) = \int_0^{+\infty} e^{-t}\, t^{x-1}\, dt$$

- **Свойства:**

  Г(0) =1;

  Г(1) =1;

  Г(*x*+1) =*x* Г(*x*)

- **Следствие:**

  Г(n+1) = n!

- **Формула Стирлинга**

$$\Gamma(x+1) \approx \sqrt{2\pi x}\, x^x\, e^{-x}, \quad x \to +\infty$$

Модели последовательностей

- Подсчитаем частоту встречаемости букв в русском языке.
- Будем генерировать символы с подсчитанными частотами
- Получим ли мы что-либо похожее на слова русского языка?

- Способ генерации случайных слов (последовательностей) называется (статистической) моделью языка.

Марковские цепи

- Очевидно, что ничего хорошего не получится.
- Есть наблюдение, что вероятность появления буквы зависит от предыдущей буквы. Например, 'ь' никогда не появляется после гласной буквы; две гласные подряд встречаются редко, и т.п.
- Символы в последовательности ***НЕ НЕЗАВИСИМЫ***.
- Марковская модель первого порядка:

$$\mathbf{p}(S_i) = \sum_{\beta} \mathbf{p}(\beta \rightarrow S_i)\, \delta\,(\beta, S_{i-1})$$

- $p(\beta \rightarrow \alpha)$ – матрица переходных вероятностей:

$$\mathbf{p}(\beta \rightarrow \alpha) = \mathbf{P}(\alpha \mid \beta)$$

вероятность появления символа α при условии, что предыдущий символ – β.

Матрица переходных вероятностей

- Размер матрицы – $\Sigma \times \Sigma$, где Σ – число исходов (например, размер алфавита). Вообще говоря, количество исходов может быть бесконечным.
- Сумма значений в строке :

$$\sum_{\beta} \mathbf{p}(\beta \to \alpha) = 1$$

- Пусть первый символ был a. Тогда распределение вероятностей для второго символа будет

$$\mathbf{p}_2(\alpha) = \mathbf{p}\,(a \to \alpha) = \sum_{\beta} \mathbf{p}(\beta \to \alpha)\,\delta(\beta, a);\ \mathbf{p}_2 = \mathbf{P} \cdot \delta(\beta, a)$$

Здесь P означает матрицу переходных вероятностей, $\delta(\beta,a)$ – одномерный вектор (столбец) из нулей и единицы.

- Распределение вероятностей для n-го символа в генерированной последовательности:

$$\mathbf{p}_n(\alpha) = \sum_{\beta} \mathbf{p}(\beta \to \alpha)\,\mathbf{p}_{n-1}(\beta);$$

$$\mathbf{p}_n = \mathbf{P}\,\mathbf{p}_{n-1} = \mathbf{P}^2\mathbf{p}_{n-2} = \ldots = \mathbf{P}^{n-2}\,\mathbf{p}_2 = \mathbf{P}^{n-1}\delta(\beta, a)$$

Марковские цепи и эволюция

- Пусть происходит эволюция некоторого белка. Ясно, что некоторые замены (например, I→V) фиксируются часто, а некоторые – редко.
- Если в некоторой позиции в предке была аминокислота G, то можно построить распределение частот аминокислоты через некоторое время.
- Марковский процесс: изменение аминокислотного остатка в данной позиции.

Марковские цепи и эволюция

- Матрица P переходных вероятностей имеет вид:

$\mathbf{P} =$

	A	G	V	I	…
A	$1-S_A\,dt$	$P_{G\to A}\,dt$	$P_{V\to A}\,dt$	$P_{I\to A}\,dt$	…
G	$P_{A\to G}\,dt$	$1-S_G\,dt$	$P_{V\to G}\,dt$	$P_{I\to G}\,dt$	…
V	$P_{A\to V}\,dt$	$P_{G\to V}\,dt$	$1-S_V\,dt$	$P_{I\to V}\,dt$	…
I	$P_{A\to I}\,dt$	$P_{G\to V}\,dt$	$P_{V\to I}\,dt$	$1-S_I\,dt$	…
…	…	…	…	$S_\alpha = \sum_{\beta\neq\alpha} P(\beta\to\alpha)$	

$= \mathbf{I} + \mathbf{Q}\,dt$

Распределение вероятностей через время t:

$$p(\alpha, t) = \exp(t \cdot Q)\, p(\beta, t)$$

Загадочный объект – экспонента от матрицы. Это матрица, которая определяется через ряд Тейлора, только вместо степени числа пишется соответствующая степень матрицы.

Марковские цепи высших порядков

- Вероятность появления очередного символа зависит не от одного, а от нескольких предыдущих символов:

$$p(S_i) = \sum_{\alpha 1\, \alpha 2\, \ldots\, \alpha m} \delta(S_{i-k}, \alpha_1)\, \delta(S_{i-k+1}, \alpha_2) \ldots\, \delta(S_{i-1}, \alpha_m)$$
$$P(S_i \mid \alpha_1 \alpha_2 \ldots \alpha_m)$$

Оценка порядка марковской цепи в модели последовательностей

- Оценка переходной вероятности:

$$p^*(\alpha_1,\ldots,\alpha_k;\alpha_{k+1}) = n(\alpha_1,\ldots,\alpha_k,\alpha_{k+1}) / n(\alpha_1,\ldots,\alpha_k,*);$$

$$n(\alpha_1,\ldots,\alpha_k,*) = \sum_{\beta} n(\alpha_1,\ldots,\alpha_k,\beta)$$

- Информационный критерий Байеса: лог-правдоподобие:

$$L_k = \sum_{\text{все слова}} n(\alpha_1,\ldots,\alpha_k,\alpha_{k+1}) \ln p^*(\alpha_1,\ldots,\alpha_k;\alpha_{k+1})$$

$$\mathbf{BIC}(k) = -2L_k + a \ln(N)$$

$a = A^k(A-1)$ – число независимых параметров цепи (A – размер алфавита), N – число последовательностей в обучении.

истинный порядок цепи

$$k^* = \mathbf{argmin}_k (\mathbf{BIC}(k))$$

Задача

Муж		Жен	
лек	Кнтр	лек	кнтр
50	4	10	80
90	12	12	120

К чему бы это?

- Испытания лекарства:

М: 50/90=5/9 > 4/12=3/9

Ж: 10/12=5/6 > 80/120=4/6

- Вместе:
- Л: 60/102=20/34
- К: 84/132=7/11=21/33
- 20/34 < 21/33 !!!!

Введение в байесову статистику

- Задача. Мы 3 раза бросили монету и 3 раза выпал орел. Какова вероятность выпадения орла у этой монеты?
  - Если мы уверены, что монета не кривая, то $p = ½$
  - Допустим, что мы взяли монету из мешка, а в мешке монеты разной кривизны. Но при этом мы знаем как распределена кривизна монет $P_a(p)$ (априорное распределение).
  - Мы хотим на основе наблюдения 3о и априорного распределения ***распределений вероятностей*** оценить вероятность выпадения орла у ***данной*** монеты.

Введение в байесову статистику

- $P(3o \mid p) = p^3$;
- $P(3o, p) = P(3o \mid p)\, P_a(p) = P(p \mid 3o)\, P(3o)$;
- $P(p \mid 3o) = \{P(3o \mid p)\, P_a(p)\} / P(3o)$;
- Загадочный объект $P(3o)$ – безусловная вероятность трех орлов. Определяется из условия нормировки: $\int P(p \mid 3o) = 1$;
- Окончательно, распределение вероятностей вероятности орла будет:
- $P(p \mid 3o) = p^3\, P_a(p) / \int p^3\, P_a(p)$;

Введение в байесову статистику

- $P(p \mid 3o) = p^3 P_a(p) / \int p^3 P_a(p)\, dp$;
- В качестве оценки для искомой вероятности удобно иметь число, а не распределение:
  - Максимальное значение $p^{ML} = \text{argmax}_p (P(3o \mid p))$ – максимальное правдоподобие (max likelihood, ML)
  - Среднее значение $p^E = E(P(p \mid 3o)) = \int p\, P(p \mid 3o)\, dp$;

Введение в байесову статистику

- ML оценка (максимальное правдоподобие):

  $$p^{ML} = \text{argmax}\,(p^3) = 1;$$

- E оценка (матожидание апостериорной вероятности)

  $$p^E = \int p^4 P_a(p)\,dp \;/\; \int p^3 P_a(p)\,dp;$$

  – Если мы уверены, что монета правильная, то $P_a(p) = \delta(p - \frac{1}{2})$; $p^E = \frac{1}{2}$;

  – Если мы ничего не знаем о распределении $P_a(p)$, то положим $P_a(p) = \text{const}$. Тогда

  $$p^E = \int p^4 P_a(p)\,dp \,/\, \int p^3 P_a(p)\,dp = (1/5) \,/\, (1/4) = 4/5\ ;$$

  В более общем случае

  $$p^E(no) = (n+1)/(n+2);$$

- MAP оценка (максимум апостериорной вероятности)

  $$p^{MAP} = \text{argmax}\,\{ P(p \mid 3o)\};$$

Определения

- Пусть у нас есть несколько источников Y событий X (например, несколько монет). Тогда :

  $P(X \mid Y)$ – ***условная вероятность***

  $P(X,Y) = P(X \mid Y)\, P(Y)$ – ***совместная вероятность***

  $P(X) = \sum_Y P(X,Y) = \sum_Y P(X \mid Y)\, P(Y)$ – ***полная вероятность***

  $P(Y \mid X)$ – ***апостериорная вероятность выбора источника (правдоподобие гипотезы)***

  $P(Y)$ – ***априорная вероятность выбора источника***

- **Теорема Байеса:**

  $$P(X \mid Y) = P(Y \mid X)\, P(X) / P(Y)$$

Пример 1

- Пусть есть две кости – правильная и кривая (с вероятностью выпадения шестёрки, равной ½ вместо 1/6). Пусть нам подсовывают кривую кость с вероятностью 1%. Мы бросили кость 3 раза и 3 раза получили 6. Какова вероятность того, что нам дали кривую кость?
- $$\mathbf{P}(\text{кривая кость} \mid \text{3 шестерки}) = \frac{\mathbf{P}(\text{3 шестерки} \mid \text{кривая кость}) \cdot \mathbf{P}(\text{кривая кость})}{\mathbf{P}(\text{3 шестерки})}$$

$$\mathbf{P}(\text{3 шестерки}) = \mathbf{P}(\text{3 шестерки} \mid \text{кривая кость}) \cdot \mathbf{P}(\text{кривая кость}) + \mathbf{P}(\text{3 шестерки} \mid \text{правильная кость}) \cdot \mathbf{P}(\text{правильная кость}) = 0.5^3 \cdot 0.01 + (1/6)^3 \cdot 0.99 = 0.00125 + 0.0046 = 0.00585$$

$$\mathbf{P}(\text{кривая кость} \mid \text{3 шестерки}) = 0.00125/0.00585 = 0.21$$

- **Вывод – кость скорее правильная!**
- **Сколько шестерок подряд надо, чтобы мы поняли, что нас обманывают?**

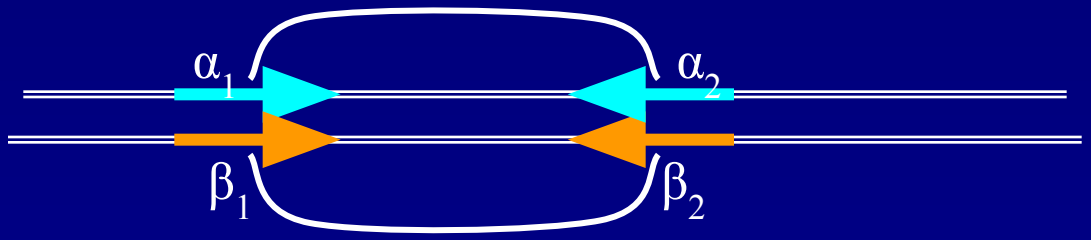
Пример 2

- Есть редкая болезнь, $P(б.)=10^{-6}$
- Имеется тест со свойствами: если больны, то вероятность ошибки теста $P(–|б.) = 0$, если здоровы, $P(+|з.)=10^{-4}$
- Стоит ли проходить тест?

$$P(б.|+)=P(+|б.)\cdot P(б.) / P(+);$$
$$P(+)= P(+|б.)\cdot P(б.) + P(+|з.)P(з.)\approx 10^{-4}$$
$$P(б.|+)=10^{-6}/10^{-4}=10^{-2}$$
$$P(з.|+)=1 – P(б.|+)=0,99$$

Пример 3

- В последовательности A нашли взаимно-комплементарную структуру.
- Последовательность B имеет степень сходства id.
- В выроненных участках последовательности B нашли аналогичную шпильку (буквы в ней не обязательно такие же, важно, что шпилька)
- Какова значимость этого наблюдения?



Символы α_1 и взаимно-комплементарны; Символы β_1 и β_2 также взаимно-комплементарны; Найдем вероятность такого события.

Пример 3 (продолжение)

$$P(\beta_1 \leftrightarrow \beta_2 \mid \alpha_1 \leftrightarrow \alpha_2) =$$

$$P(\beta_1 = 'a', \beta_2 = 't' \mid \alpha_1 \leftrightarrow \alpha_2) + P(\beta_1 = 't', \beta_2 = 'a' \mid \alpha_1 \leftrightarrow \alpha_2) + \ldots$$

Первый член :

$$P(\beta_1 = 'a', \beta_2 = 't' \mid \alpha_1 \leftrightarrow \alpha_2) =$$

$$P(\beta_1 = 'a', \beta_2 = 't' \mid \alpha_1 = 'a', \alpha_2 = 't') \cdot P(\alpha_1 = 'a') +$$

$$P(\beta_1 = 'a', \beta_2 = 't' \mid \alpha_1 = 'g', \alpha_2 = 'c') \cdot P(\alpha_1 = 'g') + \ldots =$$

$$\left(id^2 + 3 \cdot \left(\frac{1 - id}{3} \right)^2 \right) \cdot \frac{1}{4} = \frac{1}{4} \cdot \left(id^2 + \frac{(1 - id)^2}{3} \right)$$

Таких членов 4. Итого

$$P(\beta_1 \leftrightarrow \beta_2 \mid \alpha_1 \leftrightarrow \alpha_2) = \left(id^2 + \frac{(1 - id)^2}{3} \right)$$

Пример 4

- Пусть ORF начинается всегда с ATG и кончается стоп-кодоном. Найти распределение длин ORF.

$P(\text{ORF длины L} \mid \text{с поз. i}) = P(\text{start}) \cdot P^{L-2}(\text{codon}) P(\text{stop})$

$P(\text{ORF длины L}) =$

$P(\text{ORF длины L} \mid \text{с поз. i}) / P(\text{ORF с поз. i})$

Оценка параметров по результатам

- Пусть у нас есть наблюдение ***D*** и некоторый набор параметров распределения θ, которые мы хотим оценить (см. пример про 3 орла). Кроме того, у нас есть представление о том, как эти параметры распределены (***prior***)
- Апостериорное распределение вероятностей параметров получаем из теоремы Байеса:

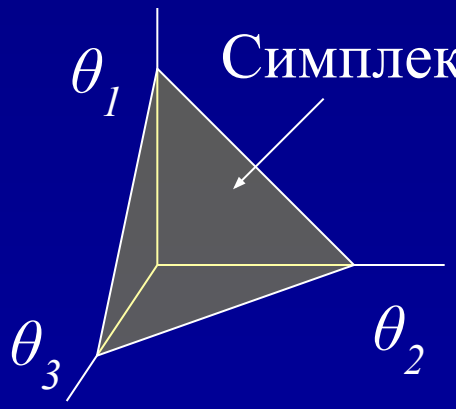
$$\boldsymbol{P}(\theta \mid \boldsymbol{D}) = \frac{\boldsymbol{P}(\theta)\,\boldsymbol{P}(D \mid \theta)}{\int_{\theta} \boldsymbol{P}(\theta')\,\boldsymbol{P}(D \mid \theta')}$$

Распределение Дирихле

- Определение:

$$D(\theta|\alpha)=Z^{-1}\prod \theta_i^{\alpha i}\,\delta(\sum \theta_i - 1);$$

  - Z – нормировочный множитель
  - α_i – параметры распределения
  - $\theta_i \geq 0$ – область определения распределения
  - δ – дельта-функция ($\delta(x)=0$, $x\neq 0$; $\int \delta(x)dx=1$;)



Задача: найти объем симплекса в n-мерном пространстве

Оценка по максимуму апостериорной вероятности (MAP)

- Пусть есть модель с L исходами.
- Пусть есть наблюдения $n_1, n_2, \ldots, n_L$.
- Пусть априорное распределение – распределение Дирихле с параметрами $\alpha_1, \alpha_2, \ldots, \alpha_L$:

$$f(\theta_1 \ldots \theta_L) = Z^{-1} \prod_i \theta^{\alpha_i}$$

- Найдем максимальную апостериорную вероятность

$$P(\theta \mid D) = \frac{P(D \mid \theta) P(\theta)}{\int_{simplex} P(D \mid \theta) P(\theta) d\theta} = \frac{\theta_1^{n_1} \cdot \theta_2^{n_2} \cdot \boxtimes \cdot \theta_L^{n_L} \cdot \theta_1^{\alpha_1} \cdot \theta_2^{\alpha_2} \cdot \boxtimes \cdot \theta_L^{\alpha_L}}{\int_{simplex} P(D \mid \theta) P(\theta) d\theta}$$

- Условие максимума при ограничении $\sum\theta=1$

$$\frac{\partial}{\partial \theta_i} \left(P(\theta \mid D) - \lambda \left(\sum_i \theta_i - 1 \right) \right) = 0;$$

MAP-оценка

$$\frac{\partial}{\partial \theta_i}\left(P(\theta \mid D) - \lambda\left(\sum_k \theta_k - 1 \right) \right) = 0;$$

$$(n_i + \alpha_i) \cdot \theta_i^{n_i + \alpha_i - 1} \cdot \prod_{k \neq i} \theta_k^{n_k + \alpha_k} - \lambda = 0;$$

$$(n_i + \alpha_i) \cdot \prod_k \theta_k^{n_k + \alpha_k} - \lambda \cdot \theta_i = 0;$$

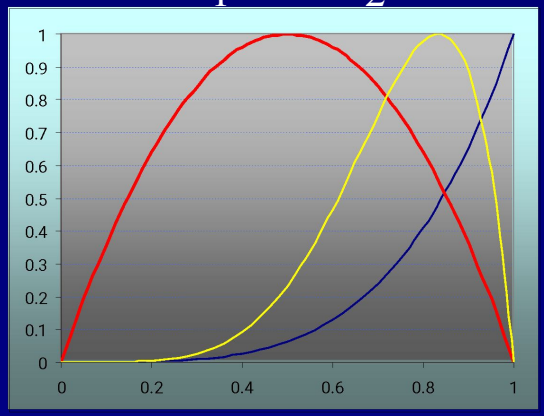
$$\theta_i = (n_i + \alpha_i) \cdot \frac{\prod_k \theta_k^{n_k + \alpha_k}}{\lambda} = (n_i + \alpha_i) \cdot \xi;$$

Из условия $\sum_k \theta_k = 1$ получаем : $\theta_i = \frac{n_i + \alpha_i}{\sum_k (n_k + \alpha_k)}$

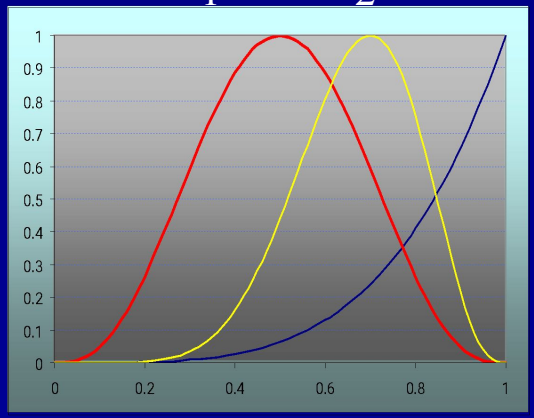
prior = распределение Дирихле

- Часто в качестве prior используют распределение Дирихле. Параметры этого распределения α_i называют ***псевдо-отсчетами (pseudo counts).*** Они определяют степень нашего доверия к результатам
- На графиках показаны распределения для случая 4-х орлов при 4-х бросаниях монеты. θ – вероятность орла
  - Синяя линия – $\mathbf{P}(D \mid \theta)$
  - Красная линия – распределение Дирихле $\mathbf{P}(\theta)$
  - Желтая линия – апостериорная вероятность выпадения орла $\mathbf{P}(\theta \mid D)$

$\alpha_1=1, \alpha_2=1$



$\alpha_1=3, \alpha_2=3$



Скрытые Марковские модели (HMM)

Пример

- Пусть некто имеет две монеты – правильную и кривую. Он бросает монету и сообщает нам серию результатов. С некоторой вероятностью он может подменить монету. Моменты подмены монеты нам неизвестны, но известно:
  - результаты бросков
  - вероятность с которой он заменяет монету
  - степень кривизны каждой монеты
- Задача: определить моменты смены монеты

Биологические примеры

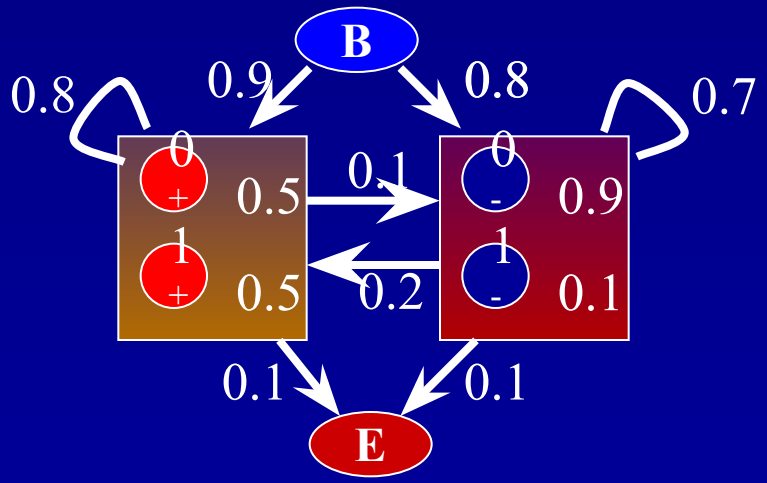
- Дана аминокислотная последовательность трансмембранного белка. Известно, что частоты встречаемости аминокислот в трансмембранных и в растворимых частях белка различаются (аналог разных монет). Определить по последовательности где находятся трансмембранные участки.
- Дана геномная последовательность. Статистические свойства кодирующих областей отличаются от свойств некодирующих областей. Найти кодирующие области.
- …
- …
- …

Описание HMM

- Пример с монетой можно представить в виде схемы ***конечного автомата***:
  - Прямоугольники означают ***состояния***
  - Кружки означают результат бросания (***эмиссии***)
  - Стрелки – возможные ***переходы*** между состояниями
  - Числа около кружков – вероятности эмиссии e_i
  - числа около стрелок – вероятности переходов между состояниями a_{ik}
  - Есть начальное и конечное состояния

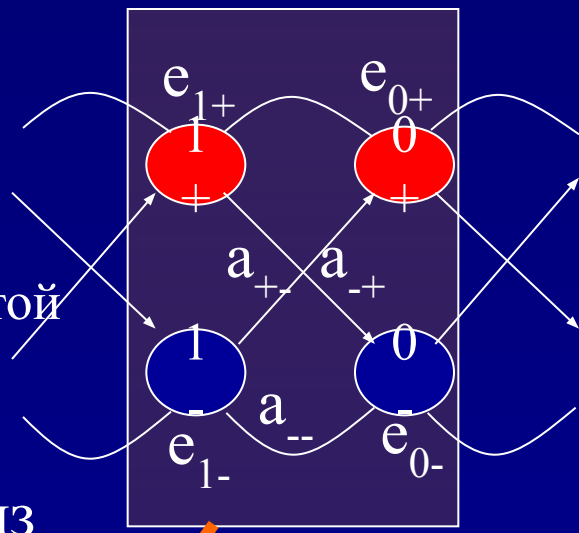
- Сумма весов исходящих стрелок равна 1
- Сумма весов эмиссии в каждом состоянии рана 1

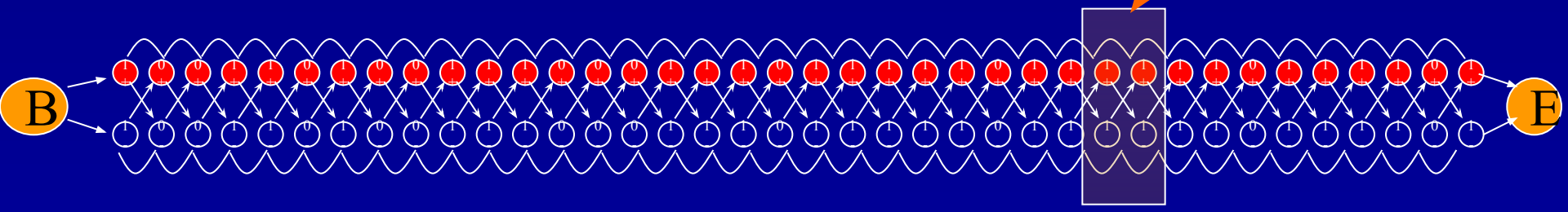
(Конечный автомат *Мура* – в алгоритмах был автомат *Мили*)



Решение задачи о монете

- Пусть нам известна серия бросков:
  `10011010011100011101111101111110111101`
- Этой серии можно поставить в соответствие граф переходов:
  - Красные вершины соответствуют эмиссии соответствующих значений правильной монетой
  - Синие вершины – эмиссия значений кривой монетой
  - на ребрах – вероятности переходов
  - на вершинах – вероятности эмиссии
- Каждому пути по графу соответствует одна из гипотез о порядке смены монеты





Решение задачи о монете

- Для любого пути можно подсчитать вероятность того, что наблюденная серия соответствует этому пути (порядку смены монет)

$$P = a_{0,1} \cdot \prod a_{i,i+1} \cdot e_{i+1}$$

- Найдем путь, отвечающий максимуму P. log является монотонной функцией, поэтому можно прологарифмировать формулу для вероятности. (почему?)

$$\pi^* = \operatorname{argmin} \{ -\log a_{01} - \sum_{\pi} (\log(a_{i,i+1}) + \log(e_{i+1})) \}$$

- Это задача поиска оптимального пути на графе. Решается динамическим программированием
- Алгоритм динамического программирования для поиска наиболее вероятного пути называется ***Viterbi***

Viterbi рекурсия

- ***Обозначения***
  - $v_k(i)$ – наилучшая вероятность пути, проходящего через позицию i в состоянии k.
  - $\pi_k(i)$ – наилучший переход из позиции i в состоянии k в предыдущую позицию (предыдущее состояние)
  - $\pi^*(i)$ – наилучшее состояние в позиции i
- ***Инициация***

  $v_k(0) = \delta(0,k)$; k – номер состояния
- ***Рекурсия***

  $v_k(i) = e_k(x_i) \max_m (v_m(i-1) a_{mk})$;

  $\pi(i,k) = \text{argmax}_m (v_m(i-1) a_{mk})$; обратный переход
- ***Завершение***

  $P(x,\pi^*) = \max_m (v_m(L) a_{m0})$;

  $\pi^*(L) = \text{argmax}_m (v_m (L) a_{m0})$;
- ***Оптимальный путь***

  $\pi^*(i-1) = \pi (i, \pi^* (i))$;

Другая постановка задачи

- Для каждого наблюденного значения определить вероятность того, что в этот момент монета была правильной.
- Для этого надо просуммировать по всем путям, проходящим через точку i_+ вероятности этих путей. Для решения этой задачи достаточно вспомнить динамическое программирование над полукольцом с использованием операции сложения и умножения.
- Нас интересует вероятность

  $\mathbf{P}(\pi_i=k \mid x) = \mathbf{P}(x, \pi_i=k) / P(x)$
- Оцениваем значение

  $\mathbf{P}(x, \pi_i=k) = \mathbf{P}(x_1 \ldots x_i, \pi_i=k) \cdot \mathbf{P}(x_{i+1} \ldots x_L \mid \pi_i=k)$
  - Первый сомножитель $f_k(i) = P(x_1 \ldots x_i, \pi_i=k)$ определяем просмотром вперед
  - Второй сомножитель $b_k(i+1) = P(x_{i+1} \ldots x_L \mid \pi_i=k)$ определяем просмотром назад

Алгоритм Forward / backward

- Forward: по определению

$f_k(i) = P(x_1 \ldots x_i, \pi_i = k)$

$f_0(0)=1,\ f_k(0)=0,\ k>0$

$f_l(i) = e_i(x_i) \sum_k f_k(i-1)\ a_{kl}$

$P(X) = \sum_k f_k(L) a_{k0}$

- Backward:

$b_k(i) = P(x_{i+1} \ldots x_L \mid \pi_i = k)$

$b_k(L) = a_{k0}$

$b_k(i) = \sum_l a_{kl}\, e_l(x_{i+1})\, b_l(i+1)$

$P(X) = \sum_l a_{0l}\, e_l(x_1)\, b_l(1)$

Оценка параметров HMM

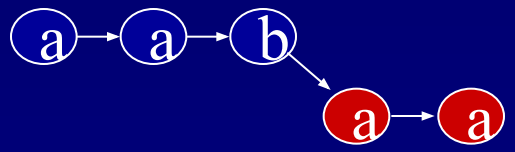
- Есть две постановки задачи.
  - Есть множество наблюдений с указанием, где происходит смена моделей ***(обучающая выборка, training set)***
  - Есть множество наблюдений, но смена моделей нам не дана
- В обоих случаях предполагается известными сами модели, т.е. конечные автоматы описаны, но неизвестны числа на стрелках и вероятности эмиссии.

Оценка параметров HMM при наличии обучающей выборки

- Здесь используется техника оценки параметров методом наибольшего правдоподобия.
- Пусть
  - x^n – набор независимых наблюдений
  - θ – набор параметров, которые надо оценить
- Тогда надо максимизировать

$$\theta^* = \text{argmax}_\theta \, l(x^1 \ldots x^n \mid \theta) = \text{argmax}_\theta \{\sum_j \log P(x^j \mid \theta)\}$$

Оценка параметров HMM при наличии обучающей выборки



$$P(x|\theta) = e_0(a)\cdot a_{00}\cdot e_0(a)\cdot a_{00}\cdot e_0(b)\cdot a_{01}\cdot e_1(a)\cdot \ldots = \prod_{k,\alpha} e_k^{E_k(\alpha)}(\alpha)\cdot \prod_{i,j} a_{ij}^{A_{ij}} \to \max$$

При условиях $\sum_{\alpha} e_k(\alpha) = 1; \qquad \sum_{j} a_{ij} = 1$

Метод неопределенных множителей Лагранжа

$$\Phi(\theta) = \sum_{k,\alpha} E_k(\alpha)\ln e_k(\alpha) + \sum_{i,j} A_{ij}\ln a_{ij} - \lambda_k\left(\sum_{\alpha} e_k(\alpha) - 1\right) - \mu_i\left(\sum_{j} a_{ij} - 1\right);$$

$$\frac{\partial\Phi}{\partial e_k(\alpha)} = \frac{E_k(\alpha)}{e_k(\alpha)} - \lambda_k = 0; \qquad \frac{\partial\Phi}{\partial a_{ij}} = \frac{A_{ij}}{a_{ij}} - \mu_k = 0; \qquad \sum_{\alpha} e_k(\alpha) = 1; \quad \sum_{j} a_{ij} = 1;$$

Оценка параметров HMM при наличии обучающей выборки

- Можно показать, что при большом количестве наблюдений справедливы оценки

  $\mathbf{a_{kl} = A_{kl} / \sum_{l'} A_{kl'}}$; $\mathbf{e_k(b) = E_k(b) / \sum_{b'} E_k(b')}$;

  - $\mathbf{A_{kl}}$ – наблюденное количество переходов между моделями
  - $\mathbf{E_k(b)}$ – количество порожденных символов в соответствующих моделях
- При малых размерах выборки используют технику псевдоотсчетов, добавляя к наблюденным значениям некоторое количество шума.

Если нет обучающей выборки

- **Итеративный алгоритм обучения Витерби.**
  1. Выберем некоторые наборы параметров HMM (обычно они генерируются случайно).
  2. Найдем для них оптимальные пути во всех представленных примерах
  3. По найденным оптимальным путям определим новые параметры, подсчитывая частоты эмиссии и переходов.
  4. Перейдем к шагу 2.
- **Итеративный алгоритм Баума-Велча** – то же самое, но параметры оцениваются с помощью Forward-Backward.
  - Показано, что алгоритм сходится (отношение правдоподобия растет на каждой итерации)
- Есть опасность нахождения локального, а не глобального экстремума.

Оценки параметров по Бауму – Велчу

- Имея заданные параметры модели можно определить вероятность перехода между состояниями:

$$P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x)}$$

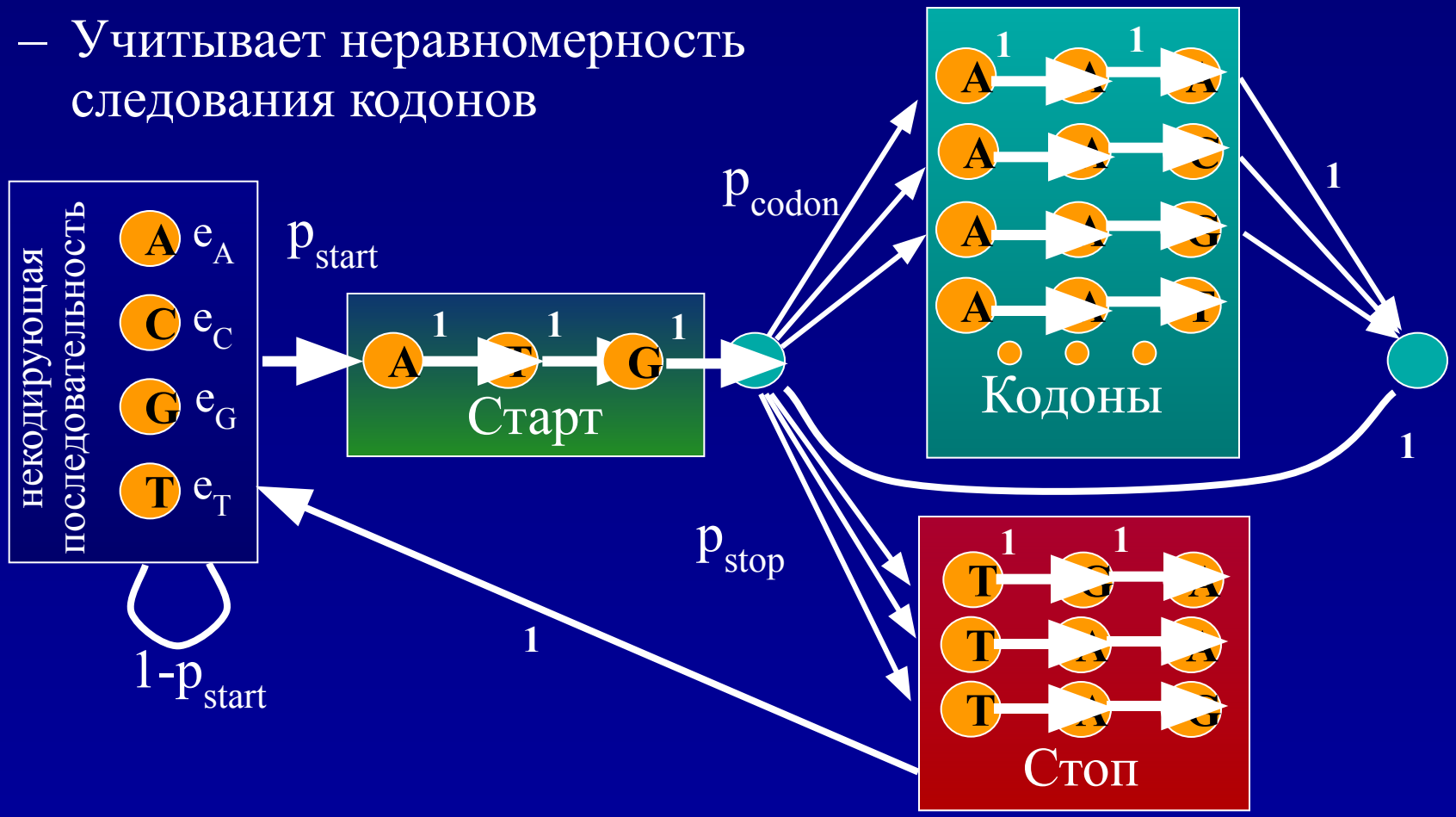
где $f_k(i) = P(x_1 \ldots x_i, \pi_i = k)$, $b_l(i+1) \cdot P(x_{i+1} \ldots x_L \mid \pi_{i+1} = l)$ – значения, полученные при прямом и обратном проходе. Тогда для переходных и эмиссионных вероятностей получим оценки для количества переходов и порожденных символов:

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1) \qquad E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i \mid x_i^j = b\}} f_k^j(i) b_k^j(i)$$

где x^j – j-последовательность в выборке,
$f^j{}_k$, $b^j{}_l$ – результаты прямого и обратного прохода по последовательности x^j

Предсказание кодирующих областей в прокариотах

- Реальная схема HMM для поиска кодирующих областей сложнее:
  - Включает в себя SD сайт
  - Учитывает неравномерность следования кодонов



Оценка качества обучения

- Выборку разбивают на два подмножества – обучающую и тестирующую
- На первой выборке подбирают параметры
- На второй – тестируют и определяют качество обучения:
  - ***TP*** – количество правильно определенных позитивных позиций (например, кодирующих)
  - ***TN*** – количество правильно определенных негативных позиций (например, некодирующих)
  - ***FP*** – количество неправильно определенных позитивных позиций (некодирующих, предсказанных как кодирующие)
  - ***FN*** – количество неправильно определенных негативных позиций (кодирующих некодирующих, предсказанных как некодирующие)

Оценка качества обучения

- Специфичность:

$$Sp = TP / (TP + FP)$$

- Чувствительность:

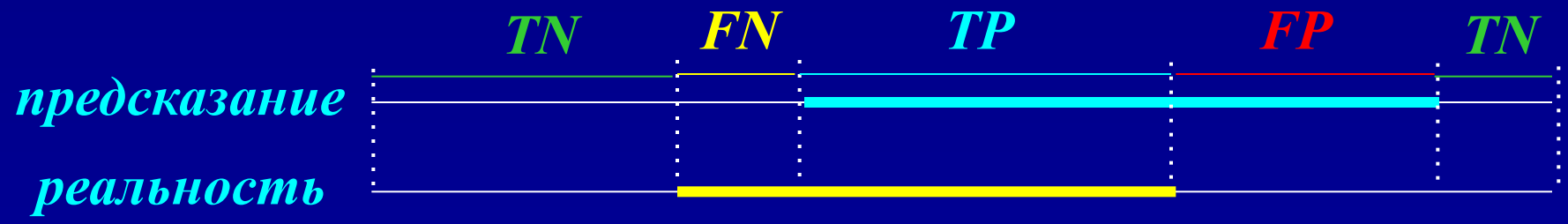
$$Sen = TP / (TP + FN)$$

- Качество (пересечение/объединение)

$$QQ = TP/(TP+FP+FN)$$

- Коэффициент корреляции

$$CC=(TP*TN–FP*FN) \; / \; \sqrt{((TP+FP)*(TN+FN)*(TP+FN)*(TN+FP))},$$



Казалось бы …

- Построим модель с миллионом параметров, включая учет притяжения Луны.
- Можно ожидать, что в этом случае мы получим очень точную модель, которая будет правильно все предсказывать.
- **НО…** При этом для оценки каждого параметра будет использовано примерно одно наблюдение. Поэтому хоть ***точность модели*** и велика, ***точность оценки параметров*** очень мала, и ее предсказательная сила будет также очень мала.

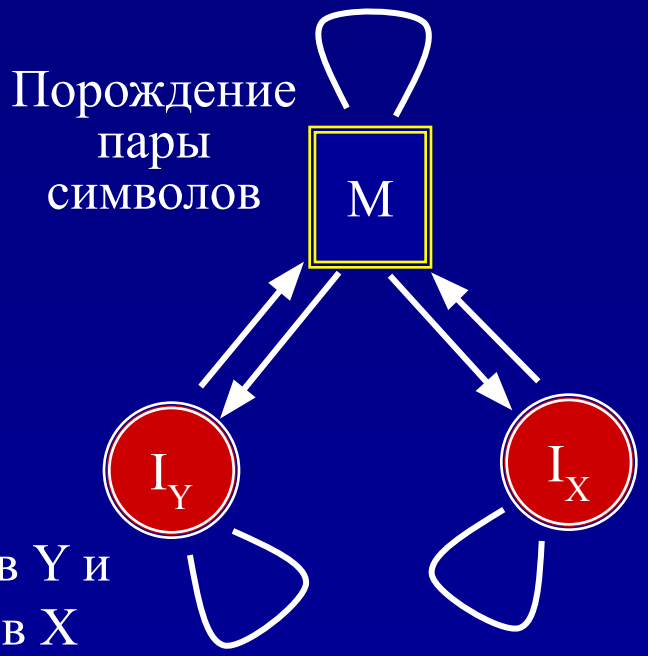
HMM
и
парное
выравнивание

Конечный автомат для парного выравнивания

$$V^M(i,j) = s(x_i, y_j) + \max\begin{cases} V^M(i-1, j-1), \\ V^X(i-1, j-1), \\ V^Y(i-1, j-1); \end{cases}$$

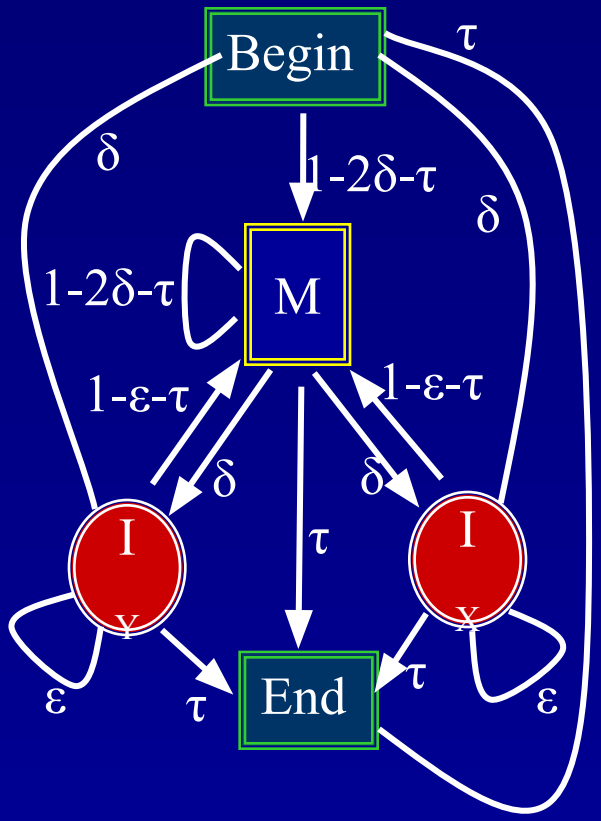
$$V^X(i,j) = \max\begin{cases} V^M(i-1, j) - d, \\ V^X(i-1, j) - e; \end{cases}$$

$$V^Y(i,j) = \max\begin{cases} V^M(i, j-1) - d, \\ V^Y(i, j-1) - e. \end{cases}$$



HMM для выравнивания

- ***Парная HMM***
- Состояния:
  - Начало
  - Сопоставление (генерация пары сопоставленных символов) $e_{ij}=s(x_i,y_j)$
  - Генерация символа в X и делеция в Y $e_i=q(x_i)$
  - Генерация символа в Y и делеция в X $e_i=q(y_i)$
  - Конец



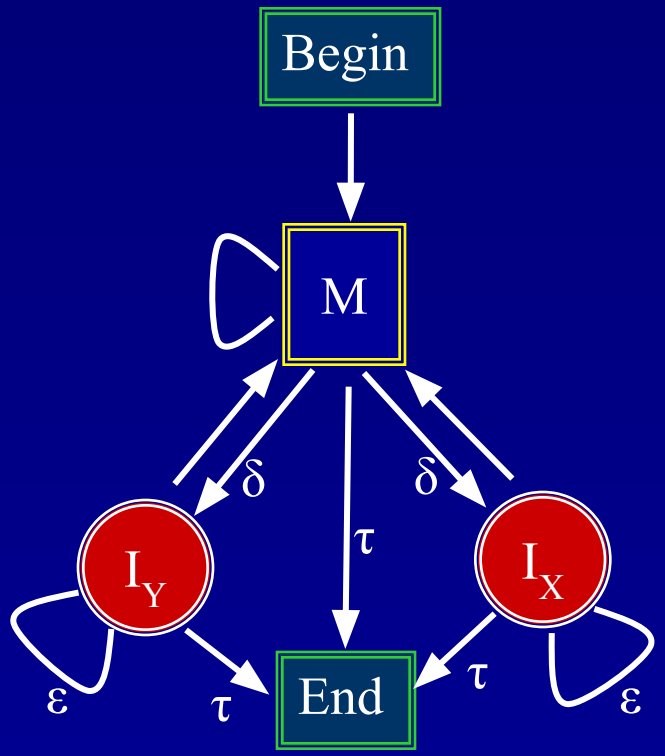
Viterbi для выравнивания

$$v^M(i,j) = p_{x_i y_j} \max\begin{cases}(1-2\delta-\tau)v^M(i-1,j-1),\\(1-\varepsilon-\tau)v^X(i-1,j-1),\\(1-\varepsilon-\tau)v^Y(i-1,j-1);\end{cases}$$

$$v^X(i,j) = q_{x_i} \max\begin{cases}\delta v^M(i-1,j),\\\varepsilon v^X(i-1,j);\end{cases}$$

$$v^Y(i,j) = q_{y_j} \max\begin{cases}\delta v^M(i,j-1),\\\varepsilon v^Y(i,j-1).\end{cases}$$

$$v^E = \tau \max(v^M(n,m), v^X(n,m), v^Y(n,m)).$$



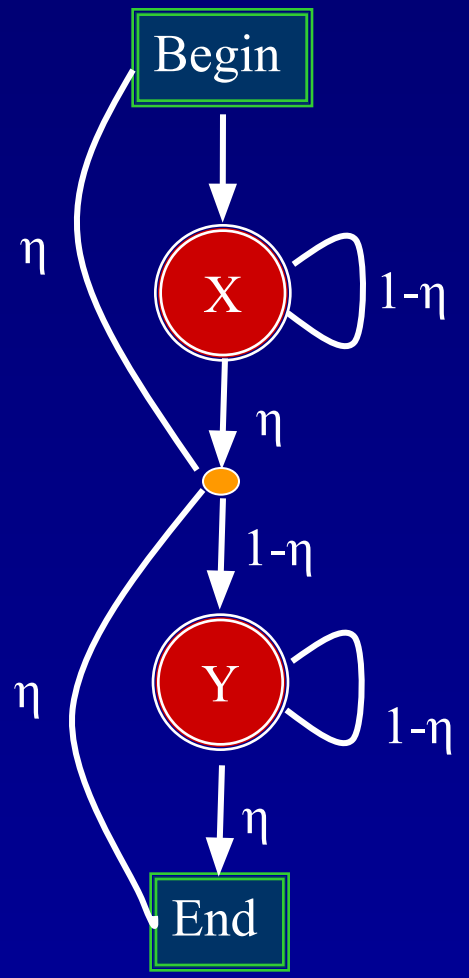
Случайная модель:
независимое порождение последовательностей

Отношение правдоподобия: $L = \frac{P(x, y \mid M)}{P(x, y \mid R)}$

Вероятность для случайного независимого порождения последовательностей

$$P(x, y|R) = \eta(1-\eta)^n \prod_{i=1}^{n} q_{x_i} \eta(1-\eta)^m \prod_{j=1}^{m} q_{x_j} =$$

$$= \eta^2 \prod_i \left(q_{x_i}(1-\eta)\right) \prod_j \left(q_{x_j}(1-\eta)\right)$$



Viterbi для **log** отношения правдоподобия

$$s(a,b) = \log\frac{p_{ab}}{q_a q_b} + \log\frac{(1-2\delta-\tau)}{(1-\eta)^2},$$

$$d = -\log\frac{\delta(1-\varepsilon-\tau)}{(1-\eta)(1-2\delta-\tau)},$$

$$e = -\log\frac{\varepsilon}{1-\tau},$$

$$V^M(i,j) = s(x_i, y_j) + \max\begin{cases} V^M(i-1,j-1), \\ V^X(i-1,j-1), \\ V^Y(i-1,j-1); \end{cases}$$

$$V^X(i,j) = \max\begin{cases} V^M(i-1,j) - d, \\ V^X(i-1,j) - e; \end{cases}$$

$$V^Y(i,j) = \max\begin{cases} V^M(i,j-1) - d, \\ V^Y(i,j-1) - e; \end{cases}$$

Завершение:

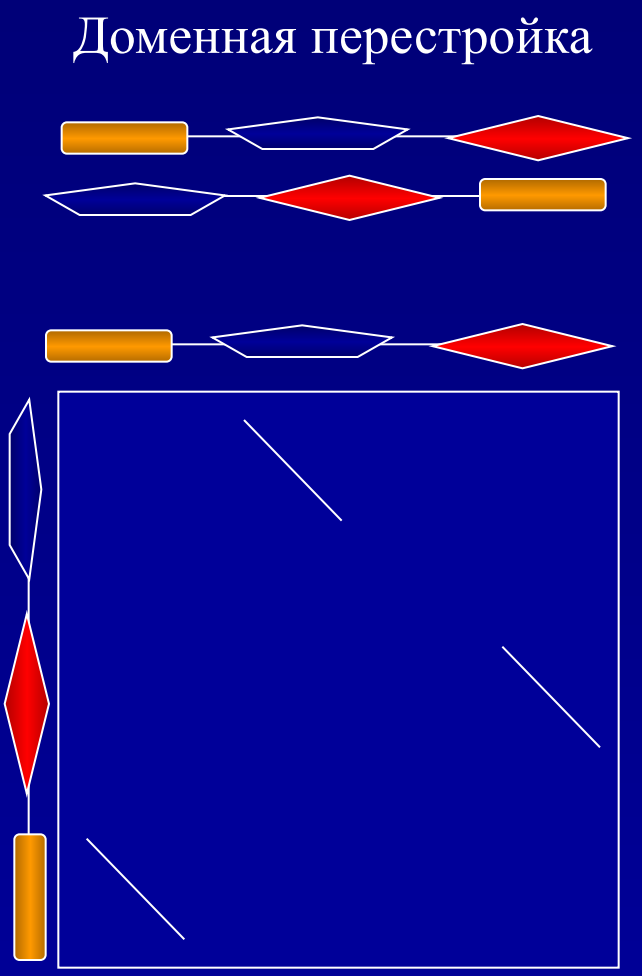
$$V = \max\begin{cases} V^M(i,j-1) - d, \\ V^Y(i,j-1) - e. \end{cases}$$

Если есть несколько слабых выравниваний

- Можно оценить полную вероятность

$$P(x,y) = \sum_{\text{выравнивания } \pi} P(x,y,\pi).$$

- Для этого можно использовать Forward - алгоритм вычисления полной вероятности



Forward

Инициация:

$$f^M(0,0) = 1. \qquad f^X(0,0) = f^Y(0,0) = 0.$$
$$f^{\bullet}(i,-1) = 0. \qquad f^{\bullet}(-1,j) = 0$$

Рекурсия:

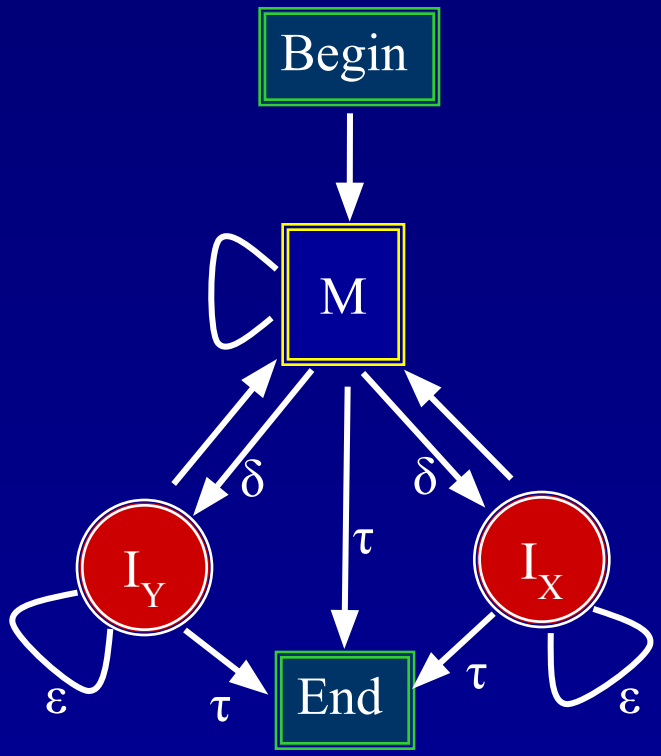
$$f^M(i,j) = p_{x_i y_j}[(1-2\delta-\tau)f^M(i-1,j-1) + (1-\varepsilon-\tau)(f^X(i-1,j-1) + f^Y(i-1,j-1))];$$

$$f^X(i,j) = q_{x_i}[\delta f^M(i-1,j) + \varepsilon f^X(i-1,j)];$$

$$f^Y(i,j) = q_{y_j}[\delta f^M(i,j-1) + \varepsilon f^Y(i,j-1)]$$

Завершение (полная вероятность):

$$f^E(n,m) = \tau[f^M(n,m) + f^X(n,m) + f^Y(n,m)].$$



Вероятностная генерация выравниваний

На обратном пути мы выбираем переходы не по максимуму, а с вероятностями:

$$P(M(i,j) \Rightarrow M(i-1,j-1)) = \frac{p_{x_i y_j}(1-2\delta-\tau) f^M(i-1,j-1)}{f^M(i,j)}$$

$$P(M(i,j) \Rightarrow X(i-1,j-1)) = \frac{p_{x_i y_j}(1-\varepsilon-\tau) f^X(i-1,j-1)}{f^M(i,j)}$$

$$P(M(i,j) \Rightarrow Y(i-1,j-1)) = \frac{p_{x_i y_j}(1-\varepsilon-\tau) f^Y(i-1,j-1)}{f^M(i,j)}.$$

$$P(X(i,j) \Rightarrow M(i-1,j)) = \frac{q_{x_i} \delta f^M(i-1,j)}{f^X(i,j)}$$

$$P(X(i,j) \Rightarrow X(i-1,j)) = \frac{q_{x_i} \varepsilon f^X(i-1,j)}{f^X(i,j)}$$

Вероятность того, что x_i и y_j выравнены

$y_j \leftrightarrow x_i$ y_j и x_i - выравнены

$$P(\boldsymbol{x}, \boldsymbol{y}, x_i \leftrightarrow y_j) =$$

$$P(\boldsymbol{x}_{1\ldots i}, \boldsymbol{y}_{1\ldots j}, x_i \leftrightarrow y_j) P(\boldsymbol{x}_{i+1\ldots n}, \boldsymbol{y}_{j+1\ldots m} \mid \boldsymbol{x}_{1\ldots i}, \boldsymbol{y}_{1\ldots j}, x_i \leftrightarrow y_j) =$$

$$P(\boldsymbol{x}_{1\ldots i}, \boldsymbol{y}_{1\ldots j}, x_i \leftrightarrow y_j) P(\boldsymbol{x}_{i+1\ldots n}, \boldsymbol{y}_{j+1\ldots m} \mid x_i \leftrightarrow y_j) =$$

$$Forward \cdot Backward =$$

$$f^M(i, j) \cdot b^M(i+1, j+1)$$

Backward

Инициализация

$$b^M(n,m) = b^X(n,m) = b^Y(n,m) = \tau.$$
$$b^{\bullet}(i,m+1) = b^{\bullet}(n+1,j) = 0.$$

Рекурсия

$$b^M(i,j) = (1-2\delta-\tau)\,p_{x_{i+1}y_{j+1}}\,b^M(i+1,j+1) + \delta[q_{x_{i+1}}\,b^X(i+1,j) + q_{y_{j+1}}\,b^Y(i,j+1)];$$

$$b^X(i,j) = (1-\varepsilon-\tau)\,p_{x_{i+1}y_{j+1}}\,b^M(i+1,j+1) + \varepsilon q_{x_{i+1}}\,b^X(i+1,j);$$

$$b^Y(i,j) = (1-\varepsilon-\tau)\,p_{x_{i+1}y_{j+1}}\,b^M(i+1,j+1) + \varepsilon q_{y_{i+1}}\,b^Y(i,j+1).$$

Искомая вероятность

$$P(x_i \text{ } y_j \mid x,y) = \frac{P(x,y,x_i \text{ } y_j)}{P(x,y)} = \frac{f^M(i,j)\,b^M(i+1,j+1)}{f^M(m,n)}$$

Информация и энтропия

Микро- и макросостояния

(кое-что из статистической физики)

- Пусть у нас есть p состояний. Числом заполнения n_i состояния i называется число частиц, находящихся в состоянии i .
- ***Микросостоянием*** системы из N частиц называется распределение (размещение) частиц по состояниям.
- ***Макросостоянием*** называется набор чисел заполнения.
- Одному Макросостоянию отвечает набор микросостояний
- ***Энтропией*** Макросостояния называется логарифм количества микросостояний, отвечающих данному макросостянию.

Энтропия

- По определению:

$$S(N) = \log(N! / (n_1! \, n_2! \, \ldots \, n_p!));$$

- используем приближение $n! = n^n e^{-n}$.

$$S(N) = N \log N - n_1 \log n_1 - n_2 \log n_2 - \ldots n_p \log n_p +$$
$$(- N + n_1 + n_2 + \ldots + n_p) =$$
$$(n_1 + n_2 + \ldots + n_p) \log N - n_1 \log n_1 - n_2 \log n_2 - \ldots n_p \log n_p;$$

- окончательно получаем:

$$\boldsymbol{S(N) = - \; N \sum_i f_i \log f_i \; ;}$$

Энтропия и информация

- Для источника символов энтропия равна:

$$H_{источника} = -\sum_{\alpha} P(\alpha) \log_2 P(\alpha)$$

если P(x) = 0 , то вклад этого члена равен 0. P(α) – вероятность генерации символа

- Энтропия – степень неопределенности при генерации символов
- Энтропия аддтивна: энтропия неопределенной последовательности X равна сумме энтропий позиций:

$$H(X) = \sum_i H_i = N H_i$$

- Энтропия максимальна, если все символы равновероятны
- При генерации последовательности неопределенность становится определенностью.
- Полное Информационное содержание – потеря энтропии:

$$I(X) = H_{before} - H_{after} = -\sum_i \sum_{\alpha} P(\alpha) \log_2 P(\alpha)$$

Информация

- Информация при генерации очередного символа:

$$I = \sum_{\alpha} P(\alpha) \log_2 P(\alpha) = \sum_{\alpha} P(\alpha)\, I(\alpha)$$

- $I(\alpha)$ – частная информация
- Частная информация (информационное содержание) последовательности:

$$I(X) = \sum_i I(x_i) = -\sum_i \log_2 P(x_i)$$

Информация выравнивания (bit-score)

```
S1 AFGILVQRSTASGNMFLC
   A|G|  Q||TA|GN F|C
S2 AYGVLVQKTTATGNWYIC
```

- **Информационное содержание выравнивания**

$$bit\text{-}score = -\sum \log_2 p(s^1_i, s^2_i);$$

Взаимная энтропия

- Вероятность макросостояния:

$$P(n_1,\ldots,n_N)=\frac{N!}{n_1!\ldots n_N!}p_1^{n_1}\cdot\ldots\cdot p_N^{n_N}$$

- Взаимная энтропия:

$$\log(P(n_1,\ldots,n_N))=\log\frac{N!}{n_1!\ldots n_N!}+n_1\log p_1+\ldots+n_N\log p_N=$$

$$-N\sum f_i\log f_i+N\sum f_i\log p_i=-N\sum f_i\log\frac{f_i}{p_i}$$

Взаимная информация

- Для двух распределений взаимная информация (расстояние Кульбака):

$$I(f \mid p) = \sum f_i \log \frac{f_i}{p_i}$$

- Свойство: если $f_i \neq p_i$, то $I(f \mid p) > 0$.
- Простое доказательство:

$$f_i = p_i + \varepsilon_i, \quad \sum \varepsilon_i = 0;$$

$$I(f \mid p) = \sum (p_i + \varepsilon_i) \cdot \log\left(1 + \frac{\varepsilon_i}{p_i}\right) \approx$$

$$\approx \sum (p_i + \varepsilon_i) \cdot \frac{\varepsilon_i}{p_i} = \sum \varepsilon_i + \sum \frac{\varepsilon_i^2}{p_i} > 0$$

Профили

Способы описания множественного выравнивания

- **Дано:** множественное выравнивание.
- **Задача:** определить принадлежит ли некая последовательность данному семейству.
- Простейший способ описания множественного выравнивания – консенсус – все просто и ясно – пишется наиболее часто встречающаяся буква
- Регулярное выражение (используется в Pro-Site): **L[ST]XX…**
- Матрица частот встречаемости аминокислот в колонке

```
 LSPADKTNVKAAWGKV
 LTPEEKSAVTALWGKV
 LSEGEWQLVLHVWAKV
 LSADQISTVQASFDKV
 LSAAEKTKIRSAWAPV
 LTESQAALVKSSWEEF
 LSAAQRQVIAATWKDI
 Ls......v.a.W.kv
L7..............
S.5.1...........
T.2.............
P..2............
E..213..........
A..33...........
G...1...........
D...11..........
Q....3..........
```

Энтропия колонки

- Пусть колонка содержит n_α букв типа α. Тогда вероятность появления такой колонки при случайных независимых последовательностях будет определяться мультиномиальным распределением:

$$P_{column} = \frac{N!}{\prod_\alpha n_\alpha!} \prod_\alpha p_\alpha^{n\alpha}$$; p_α – вероятность появления α

- Логарифм этой величины равен:

$$\log(P_{column}) = \log N! + \sum_\alpha (n_\alpha \log p_\alpha - \log n_\alpha!)$$

Заменим n на $N f_\alpha$ (f_α – частота) и применим оценку для факториала $n! \approx (n/e)^n$. Получим полную энтропию колонки

$$H_{column} = \log(P_{column}) = N \sum_\alpha f_\alpha (\log p_\alpha - \log f_\alpha);$$ ***доказать!***

Величина

$$\mathbf{I = -\sum_\alpha f_\alpha (\log p_\alpha - \log f_\alpha)}$$

называется ***информационным содержанием*** колонки

HMM профиль

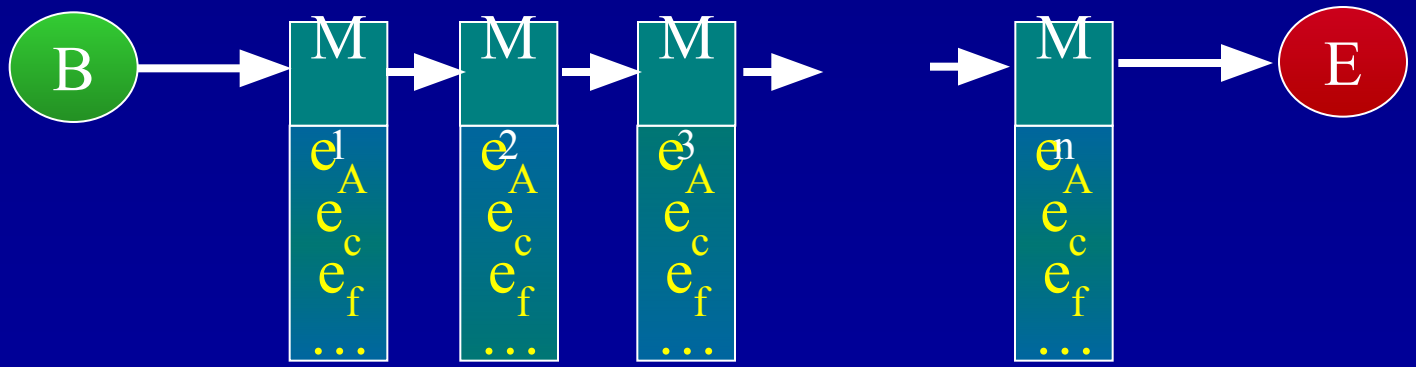
- Модель: каждая последовательность множественного выравнивания является серией скрытой Марковской модели.
- Профиль – описание Марковской модели. Каждой позиции соответствует свое состояние. Вероятности переходов между соседними состояниями равны 1.
- Вероятность того, что некоторая последовательность x соответствует профилю M:

$$P(x \mid M) = \prod e_i(x_i);$$

- Значимость определяется отношением правдоподобия: сравнением с P(x | R) – вероятностью, что последовательность сгенерирована случайной моделью R:

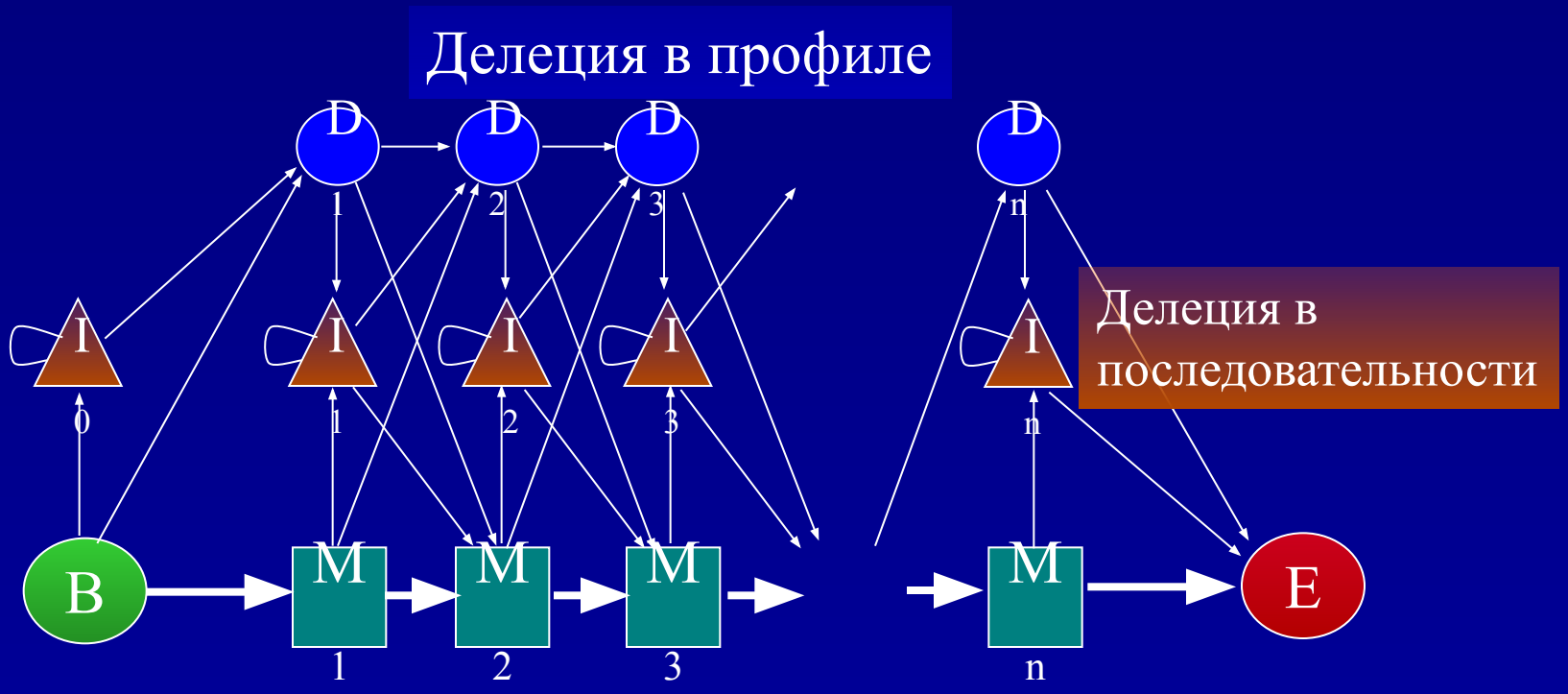
$$S = \log (P(x \mid M) / P(x \mid R)) = \sum \log \{e_i(x_i) / q(x_i)\};$$

- Величины $w_i(\alpha) = \log \{e_i(\alpha) / q(\alpha)\}$ называют позиционной весовой матрицей (PSSM, PWM)



HMM с учетом возможности вставок

- Делеция в профиле и в последовательности могут идти подряд (в отличие от парного выравнивания)
- Делеционные состояния – молчащие (не имеют эмиссии)
- Вероятность перехода в делеционное состояние зависит от позиции



Определение параметров модели

- Для начала надо определиться с длиной модели. В случае, если обучающее множественное выравнивание не имеет вставок/делеций это тривиально. Наличие же вставок/делеций требует различать вставки и делеции. Простейшее правило если колонка содержит больше половины вставок, то она не включатся в модель, а события вставок трактуются как вставки в последовательность с соответствующими эмиссионными вероятностями.
- Если выравнивание толстое, то для параметров можно использовать обычные оценки:

$$a_{kl} = A_{kl} / \sum_{l'} A_{kl'}\ ;\ \ e_k(a) = E_k / \sum_{a'} E_k(a');$$

Для тонких выравниваний

- Простейшие варианты псевдоотсчетов:
  - Правило Лапласа: к каждому счетчику прибавить 1:

$$e_k(a) = (E_k(a) + 1) / (\sum_{a'} E_k(a') + N_\alpha);$$

где N_α – размер алфавита (20)

  - Добавлять псевдоотсчеты, пропорционально фоновым частотам:

$$e_k(a) = (E_k(a) + Aq_a) / (\sum_{a'} E_k(a') + A);\ A \approx N_\alpha;$$

Такие псевдоотсчеты соответствуют Байесовой оценке

$$P(\theta \mid D) = P(D \mid \theta)\, P(\theta) / P(D);$$

при априорном распределении $P(\theta)$ – распределение Дирихле с параметром $\alpha_a = Aq_a$.

Смеси Дирихле

- Представим себе, что на распределение вероятностей влияют несколько источников – частота встречаемости символа в белках вообще, частота встречаемости символа в петлях, частота встречаемости символа в трансмембранных сегментах и т.п. Каждое такое распределение дает свои псевдоотсчеты α^k. Тогда для вероятности эмиссии можно написать:

$$e_k(a) = \sum_d P(d|E_k)\,(E_k(a) + \alpha^d_a) \,/\, (\sum_{a'} E_k(a') + \alpha^d_{a'});$$

где $P(d|E_k)$ – вероятность выбора распределения d при условии наблюдаемых частот:

$$P(d|E_k) = P(E_k|d)\,P(d) \,/\, \sum_{d'} P(E_k|d')\,P(d');$$

- Для оценки $P(E_k|d)$ используют простую формулу:

$$P(E_k|d) = \frac{(\sum_a E_k(a))!\;\Gamma(\sum_a (E_k(a) + \alpha^d_a))\;\Gamma(\sum \alpha^d_a)}{\prod_a E_k(a)!\;\prod_a \Gamma(E_k(a) + \alpha^d_a)\;\prod_a \Gamma(\alpha^d_a)}$$

Использование матрицы замен

- Еще один способ введения псевдоотсчетов. У нас есть матрица замен аминокислотных остатков (например, PAM120). Матрица замен может трактоваться как то, что каждая аминокислота является немножко другой аминокислотой. Поэтому в качестве псевдоотсчетов используют величину

$$\alpha_{ia} = A\sum_{b} f_{ib}\, P(a \mid b),$$

где f_{ib} – частота встречаемости в колонке буквы b, $P(a \mid b)$ – вероятности замены буквы b на a

Использование предка

- Все последовательности x^k в выравнивании произошли от общего предка y.

$$\mathbf{P}(y_j=a \mid \text{alignment}) = q_a \prod_k \mathbf{P}(x^k_j \mid a) \,/\, \sum_{a'} q_a \prod_k \mathbf{P}(x^k_j \mid a)$$

- Тогда для оценки эмиссионной вероятности

$$e_j(a) = \sum_{a'} \mathbf{P}_j(a \mid a')\, \mathbf{P}(y_j=a' \mid \text{alignment})$$

где $\mathbf{P}_j(a \mid a')$ – матрица замен. Матрица замен зависит от скорости эволюции соответствующей колонки. Для выбора матрицы можно использовать принцип максимального правдоподобия:

$$\mathbf{P}(x^1_j, x^2_j, \ldots, x^N_j) = \sum_{a'} q_a \prod_k \mathbf{P}(x^k_j \mid a, t) \rightarrow \max ;$$

- Для матрицы замен можно использовать выражение:

$$\mathbf{P}(a \mid b, t) = \exp(t\, \mathbf{P}(a \mid b, 1))$$

А чему же равно A?

- Для компенсации малости выборок используют псевдоотсчеты.
- Разные подходы дают разные распределения псвдоотсчетов α_i, но не определяют величину коэффициента A при α_i.
- Часто предполагают, что псевдоотсчеты должны быть сопоставимыми с точностью определения частот Δ, которая пропорциональна $\Delta \approx \sqrt{N}$, где N – количество испытаний (толщина выравнивания) поэтому полагают:

$$A = \kappa \sqrt{N},\ \kappa \approx 1\ (0.5 \ldots 1);$$

Множественное выравнивание

Множественное выравнивание

- Способ написать несколько последовательностей друг под другом (может быть с пропусками) так, чтобы в одной колонке стояли гомологичные позиции.
- "Золотой стандарт" – совмещенные пространственные структуры гомологичных белков. Соответствующие позиции в разных последовательностях отвечают гомологичным позициям
- Задача. Найти способ (алгоритм и параметры), выравнивающий последовательности "золотого стандарта" правильно. Есть надежда, что в случаях, когда пространственные структуры неизвестны, этот алгоритм правильно выровняет последовательности.

Оценка качества множественного выравнивания

Энтропийная оценка

- Обычно считают, что колонки в выравнивании независимы. Поэтому качество выравнивания можно оценить как сумму качеств колонок:

$$S = G + \sum_{\text{columns}} S(m_k)$$

G – веса делеций, $S(m_k)$ – вес колонки

- Пусть c_{ia} – количество появлений аминокислоты a в колонке i. Вероятность колонки можно описать как

$$P(m_i) = \prod_a p_{ia}^{cia}$$

- Вероятность выравнивания = $\prod_i P(m_i)$; В качестве веса можно использовать логарифм вероятности:

$$S = \sum_{\text{columns}} S(m_k); \qquad S(m_k) = -\sum_a c_{ia} \log p_{ia} = H(m_i)$$

$H(m_i)$ – энтропия колонки; для вероятностей остатков принимают:

$$p_{ia} = \tilde{c}_{ia} / \sum_{a'} \tilde{c}_{ia'}$$

где $\tilde{c}_{ia}$ – количество остатков в колонке с поправкой на псевдоотсчеты

Оценка качества множественного выравнивания

Сумма пар

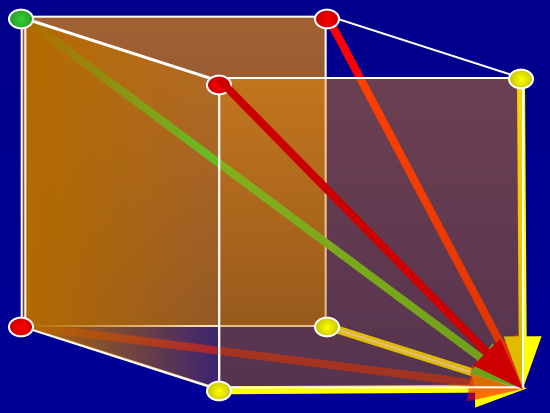
- Другой традиционный способ оценки – сумма весов матрицы соответствия аминокислотных остатков SP:

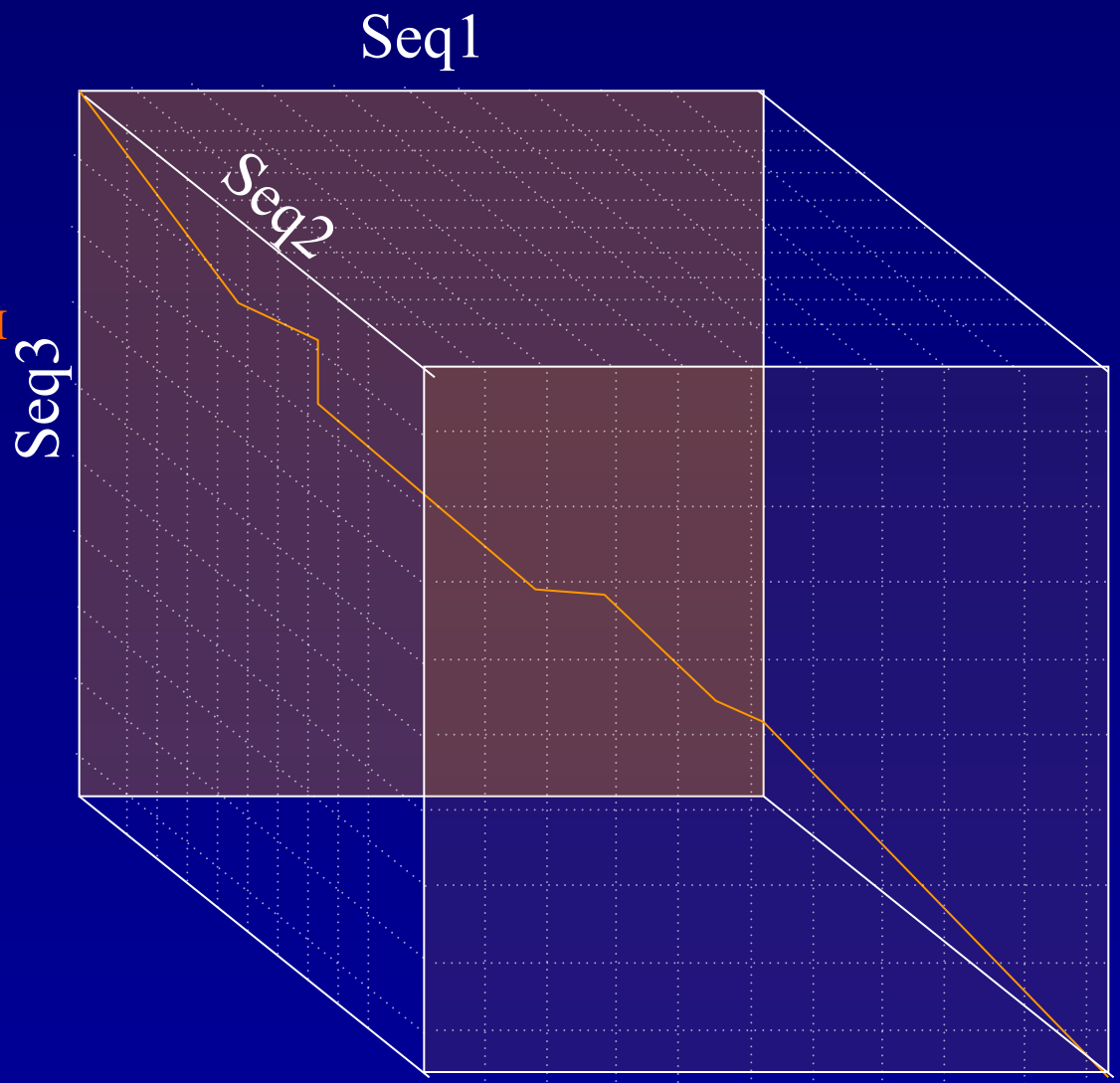
$$S(m_i) = \sum_{k<1} s(x^k_i, x^l_i);$$

- Способ не совсем правильный. Более правильная оценка для трех последовательностей $S(m_i) = \log (p_{abc} / q_a q_b q_c)$, а не $\log (p_{ab}/q_a q_b) + \log (p_{bc}/q_b q_c) + \log (p_{ac}/q_a q_c)$; (вспомним определение матрицы замен)

Если есть функционал, то его надо оптимизировать

- Элементарные переходы:
  - Сопоставление трех
  - Сопоставление двух и одна делеция
  - Делеция в двух последовательностях





Динамическое программирование для множественного выравнивания

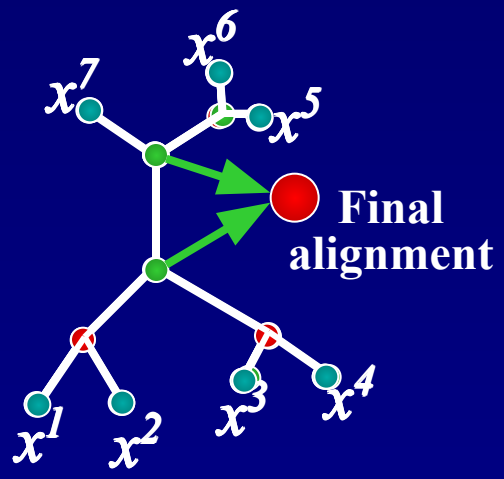
- Количество вершин равно $\prod_{посл.} L_i = O(L^N)$
- Количество ребер из каждой вершины = 2^N-1 ***(почему ?)***
- Количество операций равно

$$T = O(L^N)$$

- Надо запоминать обратные переходы в L^N вершинах.
- Если количество последовательностей > 4, то задача практически не разрешима.

Прогрессивное выравнивание

- Строится бинарное дерево (guide tree, путеводное дерево) – листья = последовательности
- Дерево обходится начиная с листьев. При объединении двух узлов строится ***парное*** выравнивание супер-последовательностей (профилей) и получается новая суперпоследовательность



Путеводное дерево строится приближенно – главное быстро. Обычно это кластерное дерево

Выравнивание профилей

- Выравнивание одной стопки последовательности относительно другой – обычное динамическое программирование.
- Оптимизируется сумма парных весов:

$$\sum_i S(m_i) \rightarrow \max, \quad S(m_i) = \sum_{k< l\le N} s(x^k_i, x^l_i)$$

- Если мы выравниваем две стопки – $0 < i \le n$ и $n < i \le N$, то сумму разбиваем на три части:

$$S(m_i) = \sum_{k< l\le n} s(x^k_i, x^l_i) + \sum_{n< k< l\le N} s(x^k_i, x^l_i) + \sum_{k\le n,\ n< l\le N} s(x^k_i, x^l_i)$$

- Две первые суммы являются внутренним делом стопок, последняя сумма отвечает за сравнение стопок (профилей)
- При сравнении используем расширенную матрицу сходства, добавив в нее сравнение делционного символа '-' :

$$s(-,-)=0,\ s(\ a\ ,-) = -d\ ;$$

- При множественном выравнивании обычно используют линейные штрафы за делеции

Взвешивание последовательностей

Это еще не все …

- При вычислении эмиссионных вероятностей используется предположение о независимости испытаний. Однако, в выравнивании часто встречаются близкие последовательности, и это предположение неверно. Например, если мы в выравнивание добавим много копий одной из последовательностей, то эмиссионные вероятности будут в основном отражать свойства именно этой последовательности.
- Пример: выравнивание содержит последовательности белка из человека, шимпанзе, гиббона, орангутанга, мыши, рыбы, мухи, комара, червяка. Очевидно, что последовательности приматов перепредставлены. Кроме того, последовательности двукрылых также перепредставлены.
- Поэтому при подсчете вероятностей необходимо каждую последовательность учитывать с весом, отражающем ее уникальность в данной выборке.

Взвешивание последовательностей

- Способ учета неравномерной представленности последовательностей в выборке называется взвешиванием последовательностей.
- Каждой последовательности в выравнивании присваивается свой вес β_k. Тогда частота каждого символа a в колонке k подсчитывается по формуле:

$$E^a_k = \sum_i \beta_i \delta(S^i_k, a) / \sum \beta_i$$

где S^i_k – буква в последовательности i в колонке k, β_i – вес последовательности i.

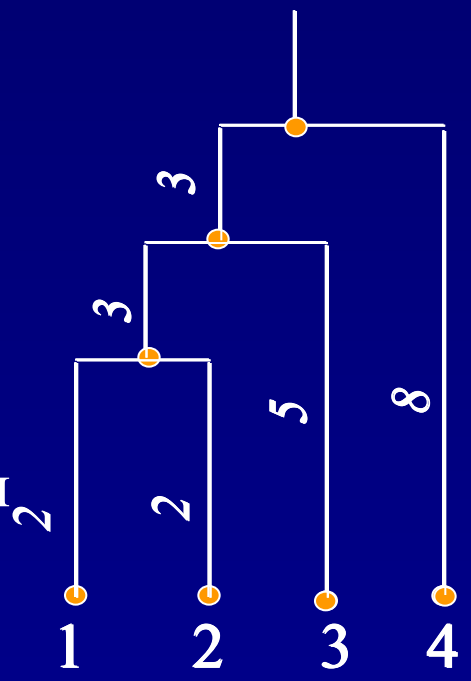
Взвешивание последовательностей

Метод Геритейна – Сонхаммера – Чотьи

- Пусть нам известно филогенетическое дерево с расстояниями на ветвях. На листьях – последовательности.
- В начале все веса последовательностей приравниваются длинам веток
- Далее веса определяем итеративно, внося поправки в веса по ходу движения вверх по дереву:

$$\Delta w_i = t_n \, w_i / \sum_{\text{k-листья ниже узла n}} w_i$$

- Смысл заключается в том, что длина ветки распределяется по дочерним узлам



$w_1 = 2 + 3/2 + 3.5/12 = 4.4$

$w_2 = 2 + 3/2 + 3.5/12 = 4.4$

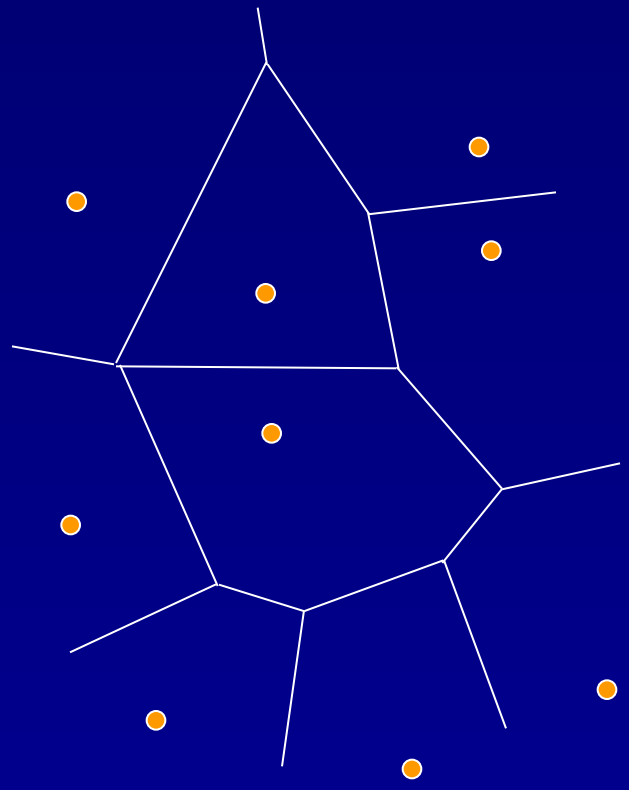
$w_3 = 5.25 + 3.5/12 = 6.25$

$w_4 = 8$

Взвешивание последовательностей

Многогранники Вороного

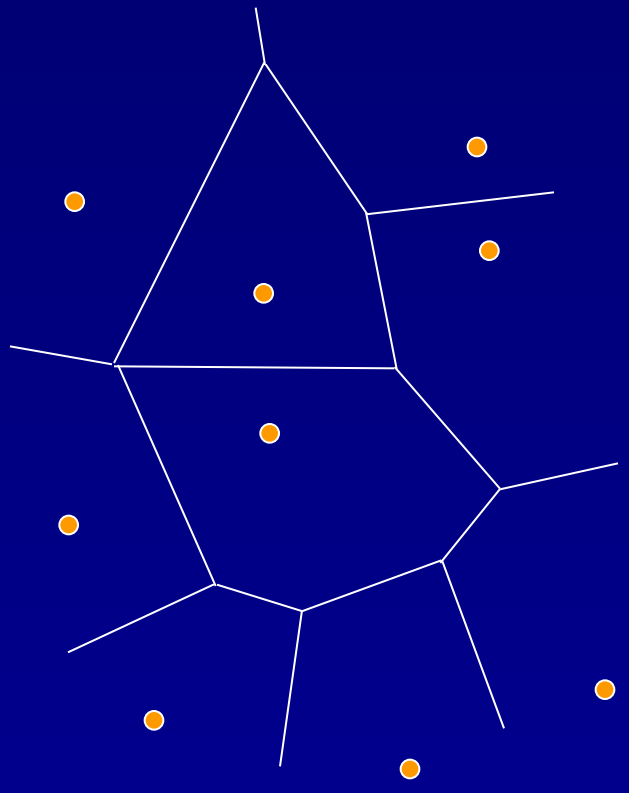
- Поместим объекты в некоторое метрическое пространство. Каждый объект хочет иметь "поместье" – некоторую область пространства. Отнесём точку пространства *x* к "поместью" объекта *A*, если *A* – самый близкий к *x* объект. Тогда границы между "поместьями" будут отрезками прямых, проходящих посредине между объектами.
- В результате все "поместья" будут иметь форму многогранника. Эта конструкция называется многогранниками Воронова.
- Можно определить вес последовательности как объем поместья. Вопрос только в том, как и в какое метрическое пространство помещать последовательности.



Взвешивание последовательностей

Многогранники Вороного

- Один из вариантов метрического пространства – большое количество случайных последовательностей
- Обычно при генерации случайных последовательностей для взвешивания по методу Вороного *i*-ая буква каждой последовательности выбирается равновероятно из букв, представленных в *i*-ой колонке входного выравнивания
- Метод часто используется, если время работы не слишком важно.



Взвешивание последовательностей

Максимизация энтропии – метод Хеникофф

- Пусть $k(i,a)$ – количество остатков типа a в колонке i, m_i – количество типов остатков в колонке i. Выберем вес для последовательности k равным

$$w_k(i)=1/(m_i\, k(i,a)).$$

- Такой вес обеспечивает наиболее равномерное распределение частот остатков в колонке. Чтобы задать вес для последовательности в целом, просуммируем соответствующие веса:

$$w_k = \sum_i w_k(i) = \sum_i 1/(m_i\, k(i,a)).$$

- Такой вес работает достаточно хорошо и считается быстро. Используется, например, в PSI-BLAST.

Взвешивание последовательностей

Максимизация энтропии

- Обобщенный подход:

$$\sum_i H_i(\mathrm{w}) \rightarrow \max, \sum_k w_k = 1;$$

где $H_i(w) = \sum_a p_{ia} \log p_{ia}$;

p_{ia} – вероятности встречаемости аминокислоты a в колонке i, подсчитанные с учетом весов последовательностей:

$$p_{ia} = \sum_k w_k \, \delta (x^k_{\mathrm{i}}, a);$$

- Задача максимизации приводит к системе уравнений:

$$\sum_k w_k = 1;$$

$$\sum_{\mathrm{i}} \partial H_i(w) / \partial w_k - \lambda = 0;$$

- Здесь неизвестные w_k и неопределенный множитель Лагранжа λ

ClustalW

1. Строится матрица расстояний с использованием попарных выравниваний.
2. По матрице расстояний строится дерево.
3. Строится прогрессивное выравнивание.

Используются дополнительные эвристики:

- Взвешивание последовательностей (с учетом только топологии дерева)
- На разных уровнях дерева используются разные матрицы сходства
- Используется контекстно-зависимые штрафы за открытие делеции
- Если при построении выравнивания появляются очень низкие веса, то дерево корректируется

Сравните время работы первого и третьего этапов

Улучшение выравнивания

- Недостаток прогрессивных методов: если для некоторой группы последовательностей выравнивание построено, то оно уже не перестраивается.
- Алгоритм итеративного улучшения
  1. Вынимаем из выравнивания одну последовательность
  2. По оставшимся последовательностям строим профиль
  3. Выравниваем вынутую последовательность с профилем. Фиксируем, иначе ли подровнялась эта последовательность.
  4. Переходим к этапу 1.
  5. Останавливаемся, если после перебора всех последовательнсотей ничего не изменилось.

Улучшение выравнивания

- Более мощный алгоритм итеративного улучшения
  1. Построим по выравниванию дерево
  2. Выберем ветвь дерева. Выбор ветви делит выравнивание на две части (последовательности по каждую сторону от ветви).
  3. Строим два профиля и выравниваем их друг с другом. Фиксируем, если выравнивание изменилось.
  4. Переходим к этапу 2.
  5. Заканчиваем, если при переборе всех ветвей ничего не изменилось.
- Этот алгоритм применён в программе Muscle, за счёт чего достигается преимущество в качестве над ClustalW.
- Преимущество в скорости достигается за счёт построения матрицы расстояний (см. первый этап ClustalW) не из парных выравниваний, а из сравнений частот слов в последовательностях.

Поиск сигналов

Постановка задачи

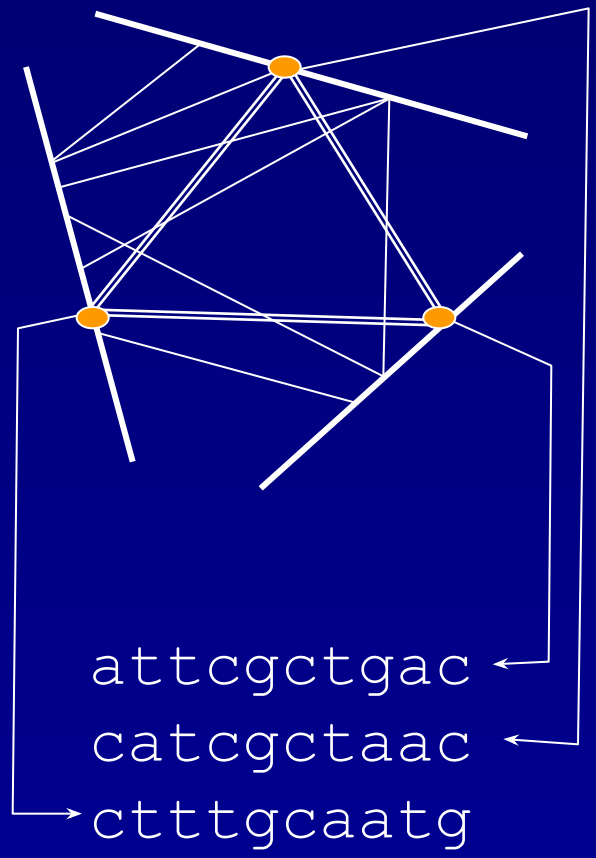
- Дано несколько (например, 20) последовательностей. Длина каждой последовательности равна 200
- В каждой последовательности найти короткий (длиной 20) фрагмент (сайт), такой, что все сайты между собой похожи.
- Например, даны регуляторные области совместно регулируемых генов. Найти сайты связывания белков-регуляторов.

Источник данных

- ChIP-Chip или ChIP-seq эксперименты
- SELEX
- Регуляторные области ортологичных генов
- Регуляторные области генов, принадлежащих общему метаболическому пути или регуляторной системе.

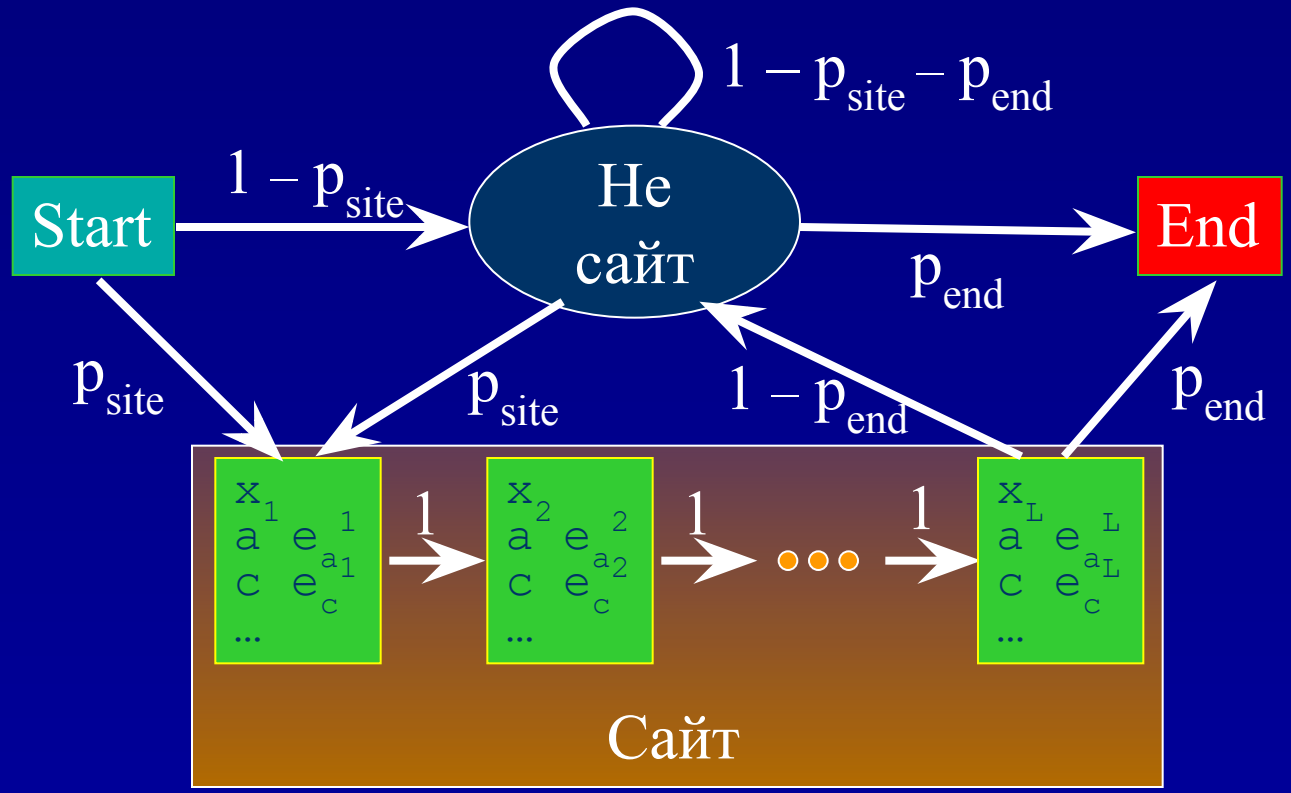
Графовая постановка задачи.

- Дан многодольный граф:
  - Каждой доле соответствует последовательность
  - Вершины – сайты
  - Ребра проводятся между всеми сайтами, или если эти сайты между собой похожи.
- На каждой клике графа определено число. Например, информационное содержание безделеционного множественного выравнивания сайтов
- **Задача:** Найти клику наибольшего веса



HMM-постановка задачи

- Найти HMM, описывающую наилучший сайт.
- Для описания сайта используют следующую модель:



Алгоритм максимизации ожидания (MEME)

- Допустим, нам приблизительно известна структура сайта.
- Применяем алгоритм Баума – Велча.
- Получаем структуру сайта.
- Алгоритм **MEME**:
  - В качестве исходной модели выбираем модель, индуцированную первым словом в первой последовательности (с учетом псевдоотсетов).
  - Находим HMM
  - Берем в качестве исходной следующее слово из первой последовательности.
  - Так перебираем ***все*** слова во ***всех*** последовательностях
  - Отбираем наилучшие HMM

Гиббс сэмплер

- **Задача:** найти набор позиций сайтов в последовательностях
- **Инициация:** В качестве решения выбираем произвольный набор позиций.
- **Итерации:**
  - Удаляем из выборки одну последовательность.
  - По позициям, определенным для остальных последовательностей строим профиль (HMM).
  - Для каждой позиции в удаленной последовательности рассчитываем вероятность того, что сайт находится там.
  - Разыгрываем позицию сайта в удаленной последовательности в соответствии с рассчитанными вероятностями.
  - Повторяем процедуру много раз для всех последовательностей

Вероятности для Гиббс сэмплера

- Вероятности для Гиббс сэмплера. Позиция разыгрывается с вероятностью, пропорциональной отношению:

$$P(pos) = Z^{-1}\frac{P_{signal}(pos)}{P_{background}(pos)}; Z = \sum_{pos}\frac{P_{signal}(pos)}{P_{background}(pos)}$$

$$P_{signal}(pos) = \prod_{i} f^{i}_{signal}\left(s_{pos+i-1}\right)$$

$$P_{background}(pos) = \prod_{i} f_{background}\left(s_{pos+i-1}\right)$$

$s(k)$ – символ в позиции k

$f^{i}_{\text{сигнал}}(\alpha)$ – частота появления символа α в позиции i сигнала. Часто используют поправки псевдоотсчетов и взвешивания последовательностей.

$f_{\text{фон}}(\alpha)$ – фоновая частота появления символа α

Дополнительные замечания

- Сигнал часто имеет структуру – палиндром, повтор.
- Обычно длина сигнала должна быть заранее известна.
- Стартуя со случайных сайтов мы можем получить:
  - Неправильное решение
  - Решение (сайты), которые по случайным причинам сдвинуты относительно настоящих сайтов

RNA

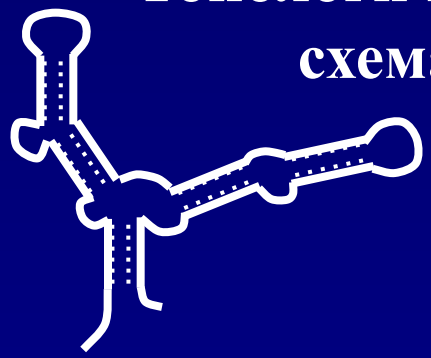
Вторичная структура РНК

- Вторичной структурой называется совокупность спаренных оснований
- Биологическая роль вторичной структуры:
  - Структурная РНК –
    - рибосомная,
    - тРНК
  - Регуляция –
    - Рибопереключатели
    - аттенюация
    - микроРНК
  - Рибозимы
  - Стабильность РНК

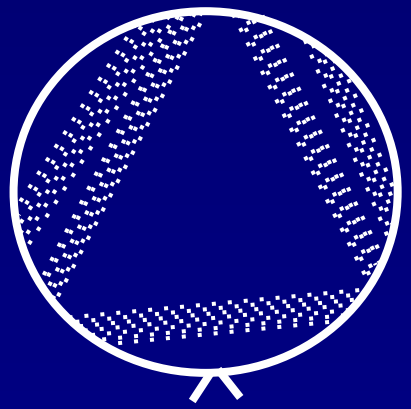
Элементы вторичной структуры

Способы представления вторичных структур

Топологическая схема



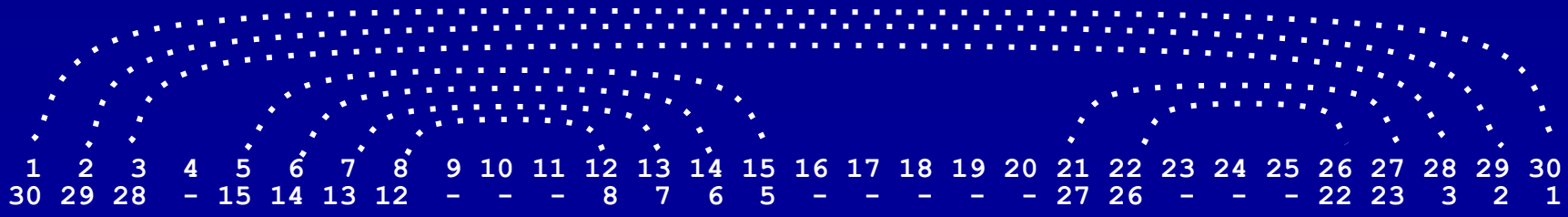
Круговая диаграмма



Список спиралей

	$from_1$	to_1	$from_2$	to_2
A	1	3	28	30
B	5	8	12	15
C	21	22	26	27

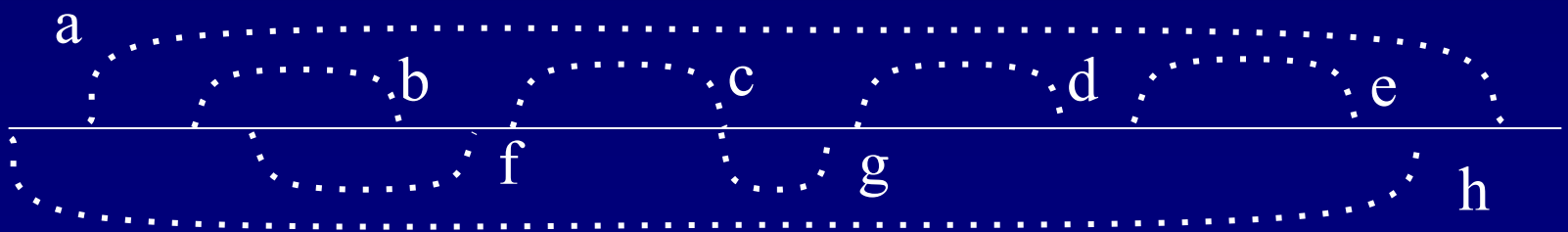
Массив спаренных оснований



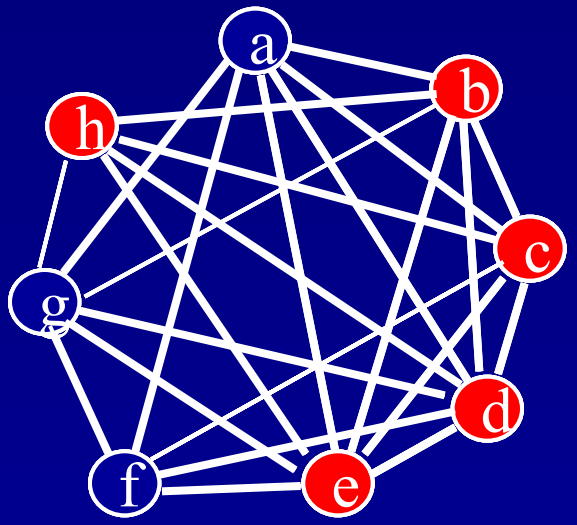
Задача

- ***Дана*** последовательность.
- ***Найти*** правильную вторичную структуру.
- ***Золотой стандарт:*** тРНК, рРНК.
- Количество возможных вторичных структур очень велико.
- Дополнительные ограничения:
  - Нет псевдоузлов. (На самом деле они очень редки и энергетически невыгодны)
- Количество возможных структур все равно очень велико
- Надо найти ***оптимальную*** структуру. А что оптимизировать? Как оптимизировать?

Комбинаторный подход



- Построим граф:
  - вершины – потенциальные нуклеотидные пары (или потенциальные спирали)
  - Ребро проводится, если пары совместимы (не образуют псевдоузлов и не имеют общих оснований)
- Допустимая вторичная структура – клика в этом графе

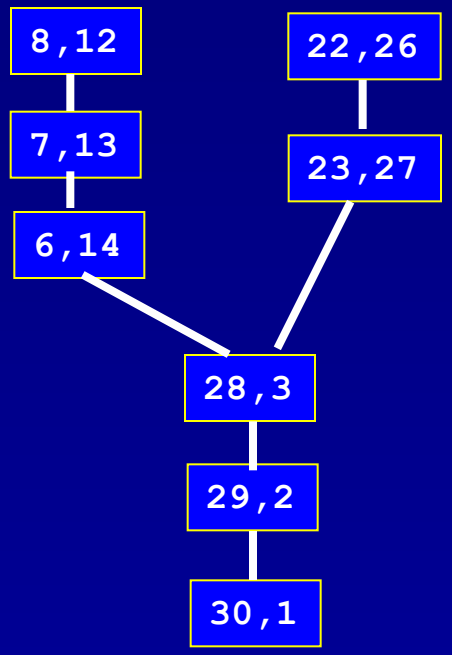


Структуры без псевдоузлов

- Структура без псевдоузлов = правильное скобочное выражение
- Может быть представлено в виде дерева
- Оценка количества возможных структур:

$$T(L) \approx 1.8^{L}$$

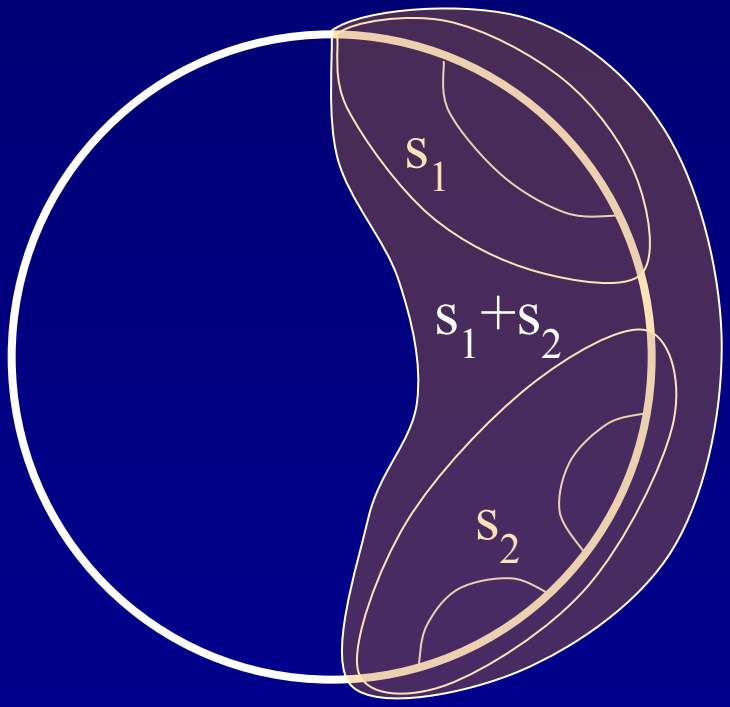
(очень много)



Оптимизация количества спаренных оснований

- Обозначим |s| - мощность структуры (количество спаренных оснований)
- Пусть s_1 и s_2 две непересекающиеся структуры (структуры без общих оснований)
- Тогда

$$|s_1+s_2| = |s_1| + |s_2|$$

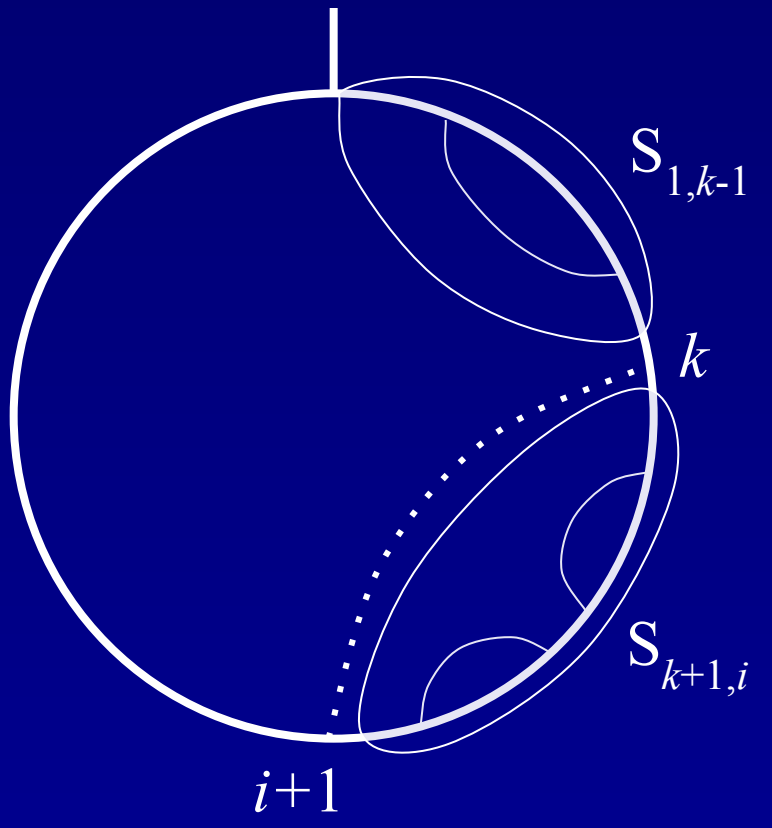


Оптимизация количества спаренных оснований

- Пусть нам известны оптимальные структуры S_{rt} для всех фрагментов

$$i \le r \le t \le j$$

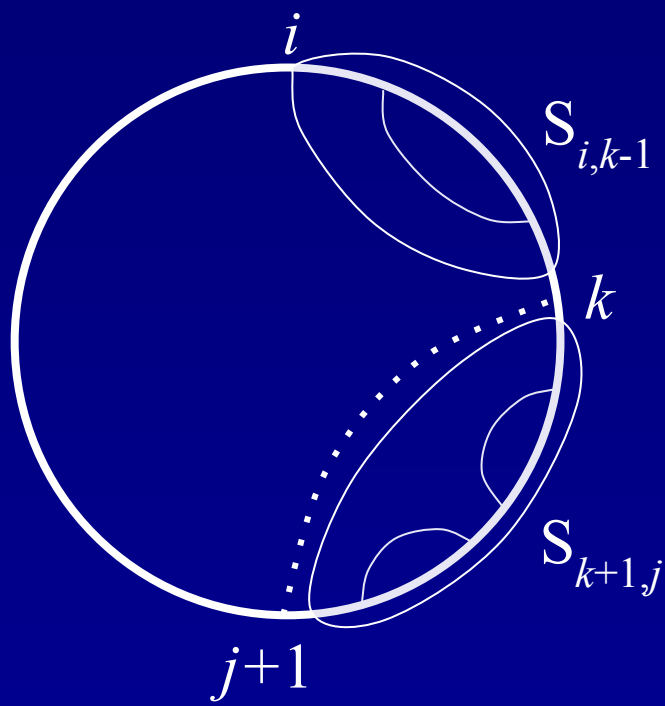
- Тогда можно найти оптимальную структуру для сегмента $[i, j+1]$
- Для этого нам надо понять, спаривать ли основание $j+1$, и, если спаривать, то с кем



Динамическое программирование для количества спаренных оснований (Нуссинофф)

- Количество спаренных оснований в оптимальной структуре $S^*_{i,j+1}$ определяется как максимум:

$$S^*_{i,j+1} = \max \{$$
$$S^*_{i,j};\ \text{(нет спаривания)}$$
$$\max_k (S^*_{i,k-1} + S^*_{k,j})+1;$$
$$(k \text{ спаривается с } j+1)$$
$$\};$$



Время работы алгоритма:

$$T \approx O(L^3)$$

Динамическое программирование для количества спаренных оснований

- При поиске оптимального количества спаренных оснований заполняется треугольная матрица весов $S_{i,j}$, $i < j$.
- Обозначим π_{ij} – номер основания, с которым надо спарить основание j при анализе сегмента $[i, j]$, или 0, если не надо спаривать. При оптимизации запоминаем треугольную матрицу спаривания (аналог матрицы обратных переходов)

Энергия вторичной структуры

- Энергия спиралей
- Энергия петель (энтропия)

Энергия спирали рассчитывается как сумма энергий стэкингов

	AU	CG			
AU	-2	-3.2			
CG	-3.2	-4.8			
GC	-3.7	-4.5			

A – U
C – G
A – U
G – C
C – G

$\Delta G = -3.2\ -3.2\ -3.7\ -4.5$

$= -14.6$

Энергия петель

- Энергия свободной цепи

$$\Delta G = B + 3/2\ kT \ln L$$

- Для шпилек при L=3..5 кроме энтропии есть некоторое напряжение структуры.
- Для внутренних петель и для мультипетель L – суммарная длина петель + количество ветвей.
- Параметр B зависит от типа петли
- Для выпячивания сохраняется стэкинг.
- Обычно используют не формулу, а таблицы.

Минимизация энергии

Обычное динамическое программирование не проходит – нет аддитивности.

- Определения
  - нуклеотид *h* называется **доступным** для пары *i•j* , если **НЕ** существует спаривания *k•l*, такого, что

$$i < k < h < l < j$$

  - Множество доступных нуклеотидов для пары *i•j* называется **петлей** L_{ij}, а пара *i•j* называется **замыкающей парой**. Частный случай петли – стэкинг.
  - Энергия структуры рассчитывается как сумма энергий петель (в том числе и стекингов):

$$\Delta G = \sum e(L_{ij})$$

Алгоритм Зукера

- Введем две переменные:
  - *W(i,j)* – минимальная энергия для структуры на фрагменте последовательности [*i, j*];
  - *V(i,j)* – минимальная энергия для структуры на фрагменте последовательности [*i, j*] при условии, что *i* и *j* спарены;
- Рекурсия:

$$V(i,j) = \min_{i<i1<j1<i2<j2<\ldots<ik<jk<j} \left(\Delta G_{loop(i,i1\ldots)} + \sum_{l}^{k} V(il, jl)\right)$$

$$W(i,j) = \min \begin{cases} W(i+1,j), & i \text{ не спарено} \\ W(i,j-1), & j \text{ не спарено} \\ V(i,j), & i,j \text{ спарены} \\ \min_{i<k<j}(W(i,k)+W(k+1,j)), & \\ & i,j \text{ спарены с кем - то} \end{cases}$$

Алгоритм Зукера

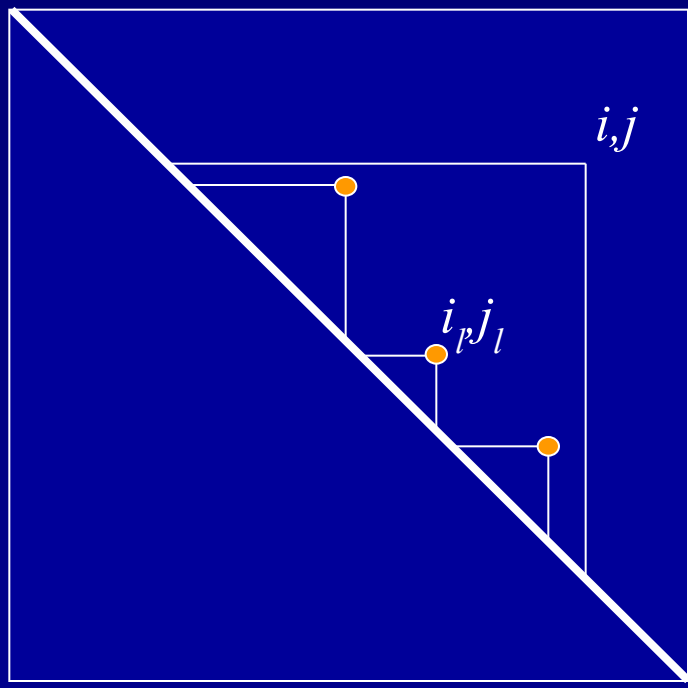
- Рекурсия для W требует времени

$$T \approx O(L^3)$$

- Рекурсия для V требует гораздо большего времени

$$T \approx O(2^L)$$

- Причина – мультипетли. Можно:
  - Ограничить размер или индекс мультипетель
  - Применить упрощенную формулу для их энергии
  - Просматривать мультипетли только если i+1, j-1 не спарены.
  - Применить приближенную эвристику



Проблемы минимизации энергии

1. Только около 60% тРНК сворачиваются в правильную структуру
2. Энергетические параметры определены не очень точно. Более того, в клетке бывают разные условия, и, соответственно, реализуются разные параметры.
3. Находится единственная структура с минимальной энергией, в то время как обычно существует несколько структур с энергией, близкой к оптимальной.

Решение проблем

- Искать субоптимальные структуры
- Искать эволюционно консервативные структуры.
  - структуры тРНК и рРНК определены именно так