Институт кибернетики им. В. М. Глушкова Национальной академии наук Украины

Островский Алексей Викторович

МЕТОДЫ РАСПОЗНАВАНИЯ НА ОСНОВЕ МОДЕЛЕЙ МАРКОВА СО СКРЫТЫМИ ПЕРЕМЕННЫМИ

Диссертация на соискание ученой степени кандидата
физико-математических наук

СОДЕРЖАНИЕ

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ

$\stackrel{\text{def}}{=}$ — равенство по определению;

$[b]$ — индикаторная функция от булевой переменной b, равная 1, если b истинно и 0, если b ложно;

$[\![b]\!]$ — логарифмическая индикаторная функция от булевой переменной b, равная 0, если b истинно и $-\infty$, если b ложно;

2^A — множество всех подмножеств множества A;

A^* — множество строк, состоящих из символов заданного конечного алфавита A;

$\log$ — натуральный логарифм;

$\mathbb{N}_0$ — множество неотрицательных целых чисел;

$|x|$ — длина строки x;

x_i — i-й символ строки x (с отсчетом от единицы);

$x_{i\ldots j}$ — подстрока x, состоящая из ее символов с i-го по j-й включительно;

xy — конкатенация строк x и y.

РАЗДЕЛ 1

ОБЗОР ЛИТЕРАТУРЫ И ВЫБОР НАПРАВЛЕНИЙ ИССЛЕДОВАНИЙ

§ 1.1. Постановка задач распознавания

¶ **1.1.1. Распознавание фрагментов генов.** Основой процессов жизнедеятельности любого биологического организма является совокупность его генов, каждый из которых содержит информацию о структуре определенного белка, синтезируемого в клетках организма. Вариативность генов обеспечивает механизмы эволюции и связанного с ней внутривидового разнообразия. Биохимической основой генов является дезоксирибонуклеиновая кислота (ДНК), структура которой была открыта в 50-х годах XX века.

ДНК является высокополимерной молекулой, состоящей из двух нитей, связанных между собой слабыми химическими связями и закрученных в спираль сложной формы. Каждая из нитей представляет собой последовательность, состоящую из четырех базовых оснований (нуклеотидов) — аденина (*adenine*, принятая аббревиатура A), цитозина (*cytosine*, C), гуанина (*guanine*, G) и тимина (*thymine*, T). Вследствие особенностей химического строения нуклеотидов они встречаются в молекуле ДНК исключительно в виде пар $A-T$ и $C-G$; таким образом, если в каком-либо месте одной из спиралей молекулы расположен аденин, то на соответствующем месте второй спирали всегда размещен тимин, и так далее (рис. 1.1) [1].

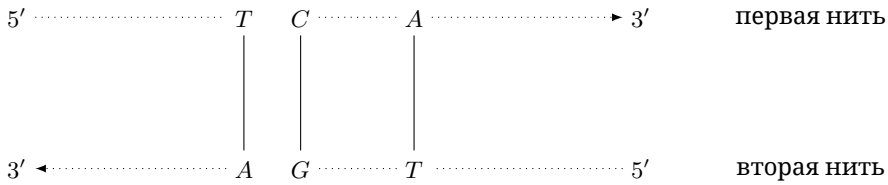


Рис. 1.1. Схематическое строение нитей ДНК

Направление перечисления оснований для каждой из нитей жестко связано с расположением в нуклеотидах атомов углерода, которые соединяют соседние основания (для них, в соответствии с химической терминологией, используются обозначения $3'$ и $5'$). Так как синтез цепочек нуклеотидов при копировании участков ДНК проходит в направлении от $5'$ к $3'$, и в этом же направлении записана информация в кодирующих белки участках генома, именно такой порядок перечисления оснований принят в литературе и будет использоваться далее. При этом выбранное направление противоположно для двух нитей.

Гены представляют собой небольшие (длиной до нескольких десятков тысяч оснований) участки ДНК, последовательность нуклеотидов в которых определяет структуру конкретного белка. Как и ДНК, белки являются полимерами, строящимися из простых элементов — аминокислот. В процессе синтеза белка на основе генетической информации участвуют двадцать аминокислот: аланин (принятое обозначение A), цистеин (C), аспарагиновая кислота (D), глутаминовая кислота (E), фенилаланин (F), глицин (G), гистидин (H), изолейцин (I), лизин (K), лейцин (L), метионин (M), аспарагин (N), пролин (P), глутамин (Q), аргинин (R), серин (S), трео-

нин (T), валин (V), триптофан (W) и тирозин (Y). Соответствие между последовательностью нуклеотидов гена и цепочкой аминокислот белка осуществляется с помощью общего для всех биологических организмов генетического кода, который сопоставляет каждой из $4^3 = 64$ возможных троек последовательных нуклеотидов (кодону) одну из перечисленных выше двадцати аминокислот (табл. 1.1). Три кодона — TAA, TAG и TGA — в универсальном генетическом коде не представляют ни одной аминокислоты, а служат сигналами прекращения синтеза белка.

Таблица 1.1

Универсальный генетический код живых организмов

Первое осн.	Второе основание				Третье осн.
	T	C	A	G	
T	фенилаланин (F)	серин (S)	тирозин (Y)	цистеин (C)	T
	фенилаланин (F)	серин (S)	тирозин (Y)	цистеин (C)	C
	лейцин (L)	серин (S)	$stop$	$stop$	A
	лейцин (L)	серин (S)	$stop$	триптофан (W)	G
C	лейцин (L)	пролин (P)	гистидин (H)	аргинин (R)	T
	лейцин (L)	пролин (P)	гистидин (H)	аргинин (R)	C
	лейцин (L)	пролин (P)	глутамин (Q)	аргинин (R)	A
	лейцин (L)	пролин (P)	глутамин (Q)	аргинин (R)	G
A	изолейцин (I)	треонин (T)	аспарагин (N)	серин (S)	T
	изолейцин (I)	треонин (T)	аспарагин (N)	серин (S)	C
	изолейцин (I)	треонин (T)	лизин (K)	аргинин (R)	A
	метионин (M)	треонин (T)	лизин (K)	аргинин (R)	G
G	валин (V)	аланин (A)	асп. к-та[1)] (D)	глицин (G)	T
	валин (V)	аланин (A)	асп. к-та (D)	глицин (G)	C
	валин (V)	аланин (A)	глут. к-та[2)] (E)	глицин (G)	A
	валин (V)	аланин (A)	глут. к-та (E)	глицин (G)	G

Примечания:

1) аспарагиновая кислота;
2) глутаминовая кислота.

Структура генов усложнялась в процессе эволюции, в результате чего участки ДНК, кодирующие генетическую информацию для эукариотов (организмы, клетки которых содержат ядро, в частности, растения и животные) имеют сложный вид (рис. 1.2). В соответствии с исполняемой при синтезе белков функции, выделяются основные составляющие генов эукариотов:

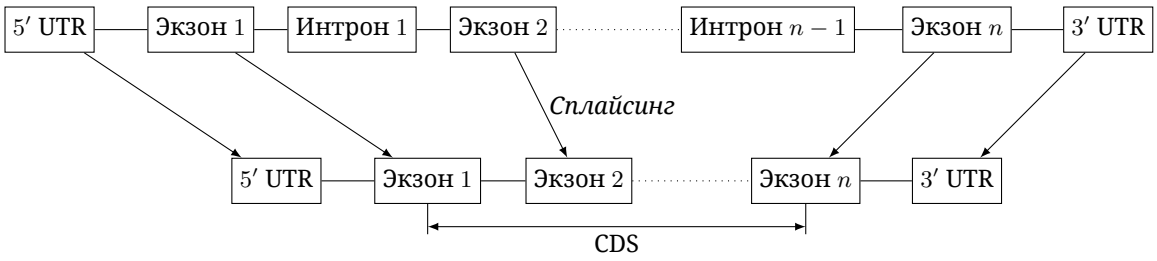


Рис. 1.2. Структура гена в ДНК и матричной РНК

- начальный и конечный некодирующие участки (*untranslated region*), обозначаемые как $5'$ UTR и $3'$-UTR соответственно. Эти участки не участвуют в процессе кодирования белков, но могут косвенно на него влиять;
- экзоны — участки ДНК, непосредственно кодирующие последовательность аминокислот, из которых состоит белок, с помощью универсального генетического кода;
- интроны — участки ДНК, расположенные между экзонами, не принимающие участия в синтезе белков. Их назначение на сегодняшний день не выяснено; возможно, интроны являются механизмом защиты от мутаций.

Поскольку UTR-участки генов лишь опосредствованно влияют на структуру белков, они не представляют интереса во многих задачах, связанных с анализом генетических данных; вследствие этого, в рамках данной работы они рассматриваться не будут. Таким образом, задача определения функциональных составляющих гена при условии, что известен участок ДНК, ему соответствующий, имеет следующую формулировку:

Задача 1.1. *Определить разбиение участка гена, заданного как последовательность известных нуклеотидов, заключенных между начальным и конечным некодирующими участками, на экзоны и интроны, т. е. указать для каждого основания класс (экзон или интрон), к которому оно относится.*

Сформулированная выше задача представляет интерес как составная часть более сложной проблемы поиска генов в последовательности нуклеотидов ДНК. В самом деле, вероятностные модели, с помощью которых ищется решение задачи 1.1, используются как часть математической модели для описания гена в целом; оценка вероятности расположения гена в определенном месте ДНК при этом зависит от вероятности оптимального разбиения выделенной каким-либо образом основной части гена на интроны и экзоны.

¶ **1.1.2. Определение вторичной структуры белка.** Белки — полимеры, составными элементами которых являются двадцать аминокислот; последовательность аминокислот, из которых состоит белок, называется его *первичной структурой*. Под влиянием разнообразных физических и химических факторов, в частности среды, в которой находится белок, цепочка аминокислот образует сложную пространственную структуру, которая определяет активные центры белка и, соответственно, его роль в биохимических реакциях, протекающих в клетке. Этот процесс — сворачивание белка — обеспечивает минимум потенциальной энергии химических связей между его составляющими.

Относительное местоположение аминокислот в рамках пространственной структуры белка можно аппроксимировать с помощью трех абстракций [2]:

- спирали — элементы пространственной структуры, имеющие форму правозакрученной винтовой линии, устойчивость которой поддерживается за счет водородных связей между аминокислотами, соответствующими последовательным виткам. В зависимости от количества аминокислот, которые составляют один виток спирали, выделяют три основных типа данной структуры: α-спирали ($3{,}6$ оснований на виток), 3_{10}-спирали ($3{,}0$ оснований на виток), а также π-спирали ($4{,}4$ оснований на виток).

- β-листы — формы пространственной структуры белка,состоящие из определенного количества относительно прямых цепей аминокислот (длиной от 3 до 10 оснований), которые связаны между собой двумя или тремя водородными связями и образуют складчатые, слабо закрученные листы.
- нерегулярные структуры — области высокой кривизны, повороты, а также остальные аминокислоты, не соответствующие спиралям или β-листам.

Совокупность сведений о принадлежности каждой из аминокислот к одному из трех перечисленных выше типов пространственных форм составляет *вторичную структуру* белка. Этот уровень пространственного строения белка определяется его первичной структурой [3]. Так, известно, что некоторые аминокислоты, в особенности глицин и пролин, практически никогда не встречаются в спиралях [4], в то время как другие (метионин, аланин, лейцин, глутаминовая кислота и лизин) имеют высокую склонность к образованию спиралей [5]. Вероятность участия аминокислоты в сложных пространственных структурах также зависит от ее полярности, и связанного с этой характеристикой отношения аминокислоты к воде.

Исходя из связи первичной и вторичной структуры белков, можно сформулировать задачу определения функциональных составляющих белка, аналогичную задаче 1.1:

Задача 1.2. *На основании заданной последовательности аминокислот белка определить для каждой аминокислоты вторичную структуру (спираль, β-лист либо отсутствие структуры), в которую она входит.*

В отличие от задачи 1.1, для большинства белков в специализированных базах данных имеются пропуски или вариации в последовательности аминокислот. В силу этого любой алгоритм, решающий задачу 1.2, должен учитывать наличие таких пропусков.

§ 1.2. Методы распознавания фрагментов генов

Большинство методов, применяемых на сегодняшний день для выделения функциональных фрагментов генов, в том или ином виде используют марковские модели со скрытыми параметрами [6; 7]. Такие модели описывают случайный процесс, в котором результаты наблюдения зависят от определенных ненаблюдаемых, или скрытых, переменных. Более формально [8]:

Определение 1.1. *Однородная скрытая марковская модель $\mathcal{M}$ — это дискретный случайный процесс, характеризуемый следующим набором параметров и переменных:*

- $Q = \{q_1, q_2, \ldots, q_n\}$ — *конечное множество скрытых состояний;*
- $O = \{o_1, o_2, \ldots, o_m\}$ — *конечное множество наблюдаемых состояний модели;*
- $\pi(q)$ — *начальная вероятность скрытого состояния $q \in Q$;*
- $p(q_j \mid q_i)$ — *вероятность перехода из скрытого состояния $q_i \in Q$ в состояние $q_j \in Q$ за единицу времени;*
- $\theta(o \mid q)$ — *вероятность наблюдения состояния $o \in O$ при скрытом состоянии $q \in Q$ в какой-либо момент времени.*

Другими словами, $\mathcal{M}$ описывается вектором $\{\pi(q_i)\}_{i=1}^n$ и двумя стохастическими матрицами: $A = \{p(q_j \mid q_i)\}_{i,j=1}^n$ размером $n \times n$ и $B = \{\theta(o_j \mid q_i)\}$ размером $n \times m$.

Согласно определению, совместная вероятность наблюдения цепочки состояний $\mathbf{o} \equiv o(1)o(2)\dots o(t)$ и соответствующей ей последовательности скрытых состояний $\mathbf{q} \equiv q(1)q(2)\dots q(t)$ составляет (рис. 1.3)

$$P(\mathbf{o}, \mathbf{q}) = P(\mathbf{o} \mid \mathbf{q})P(\mathbf{q}) = \pi(q(1))p(q(2) \mid q(1)) \times \dots \times p(q(t) \mid q(t-1)) \times \\ \times \theta(o(1) \mid q(1))\theta(o(2) \mid q(2)) \times \dots \times \theta(o(t) \mid q(t)). \quad (1.1)$$

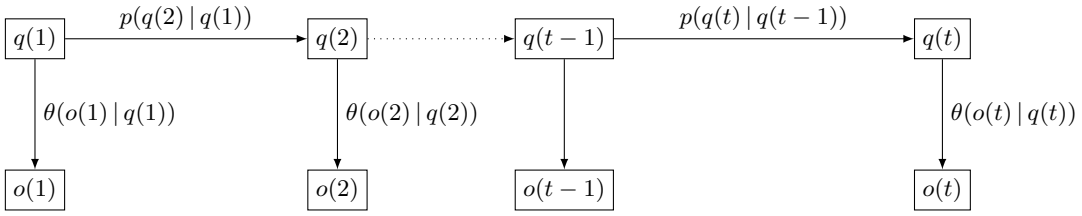


Рис. 1.3. Схематическое представление скрытой марковской модели. Подписи у стрелок обозначают вероятности переходов

Одна из задач, решаемых на скрытых марковских моделях — восстановление наиболее вероятной последовательности скрытых состояний, которые привели к известному выводу, т. е. отыскание

$$\hat{\mathbf{q}} \equiv \hat{q}(1)\hat{q}(2)\dots\hat{q}(t) = \arg\max_{\mathbf{q}} P(\mathbf{q} \mid \mathbf{o}) = \arg\max_{\mathbf{q}} \frac{P(\mathbf{o}, \mathbf{q})}{P(\mathbf{o})} = \arg\max_{\mathbf{q}} P(\mathbf{o}, \mathbf{q}). \quad (1.2)$$

Для поиска оптимальной цепочки применяется алгоритм Витерби, относящийся к методам динамического программирования [9]. В современных алгоритмах распознавания генов обыкновенные скрытые марковские модели не используются по причине их недостаточной сложности; вместо них применяются *обобщенные* скрытые марковские модели, в которых скрытые состояния могут генерировать цепочки наблюдаемых состояний произвольной длины.

Определение 1.2. *Однородная обобщенная скрытая марковская модель $\mathcal{M}^*$ — это дискретный случайный процесс, характеризуемый следующими параметрами:*

- $Q = \{q_1, q_2, \dots, q_n\}$ — *конечное множество скрытых состояний;*
- R_s — *конечное множество наблюдаемых состояний системы;*
- $\pi(q)$ — *начальная вероятность скрытого состояния $q \in Q$;*
- $p(q_j \mid q_i)$ — *вероятность перехода из состояния $q_i \in Q$ в $q_j \in Q$ за единицу времени;*
- $\theta(S \mid q_i)$ — *вероятность генерации скрытым состоянием $q_i \in Q$ последовательности $S \in R_s^*$ в произвольный момент времени.*

Таким образом, $\mathcal{M}^*$ описывается стохастической матрицей переходных вероятностей A размером $n \times n$ и n дискретными распределениями, введенными на множестве R_s^*.

Пусть состояния $q(1)$, $q(2)$,..., $q(t)$ порождают последовательности $S(1)$, $S(2)$,..., $S(t)$, которые вместе составляют строку S (рис. 1.4). Помимо самой цепочки состояний $\mathbf{q}$, представляет интерес набор величин

$$\mathbf{l} = \{l(k)\}_{k=1}^{t}, \quad l(k) \stackrel{\text{def}}{=} |S(k)|,$$

которые характеризуют разбиение S на порожденные скрытыми состояниями подстроки.

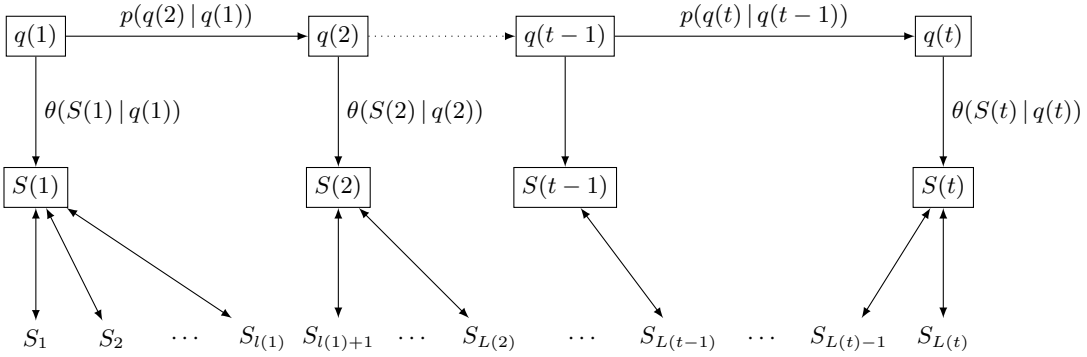


Рис. 1.4. Схематическое представление обобщенной скрытой марковской модели. Использовано обозначение $L(i) = \sum_{j=1}^{i} l(j)$ — общая длина строк, генерируемых первыми i скрытыми состояниями

Совместная вероятность наблюдаемых и скрытых состояний составляет

$$P(S, \mathbf{q}, \mathbf{l}) = P(S \mid \mathbf{q}, \mathbf{l}) P(\mathbf{l} \mid \mathbf{q}) P(\mathbf{q}) = \pi(q(1)) p(q(2) \mid q(1)) \times \cdots \times p(q(t) \mid q(t-1)) \times$$
$$\times \theta(S(1) \mid q(1)) \theta(S(2) \mid q(2)) \times \cdots \times \theta(S(t) \mid q(t)) P(\mathbf{l} \mid \mathbf{q}). \quad (1.3)$$

Вероятность наблюдения разбиения для заданных скрытых состояний равна

$$P(\mathbf{l} \mid \mathbf{q}) = \prod_{i=1}^{t} \varphi(l(i) \mid q(i)),$$

где через $\varphi(d \mid q)$ обозначена вероятность того, что состояние $q \in Q$ генерирует строку длины d:

$$\varphi(d \mid q) = \sum_{S \in R_s^d} p(S \mid q).$$

Задача отыскания оптимальной цепочки скрытых состояний и соответствующего разбиения наблюдаемой строки S тогда сводится к нахождению точки максимума

$$(\hat{\mathbf{q}}, \hat{\mathbf{l}}) = \arg\max_{\mathbf{q},\mathbf{l}} P(\mathbf{q}, \mathbf{l} \mid S) = \arg\max_{\mathbf{q},\mathbf{l}} P(S, \mathbf{q}, \mathbf{l}),$$

причем перебор производится по всевозможным цепочкам q произвольной длины и неубывающим последовательностям

$$\mathbf{l} \equiv (l_1, \ldots, l_t) : \forall i = 1, \ldots, t \quad l_i > 0, \ \sum_{i=1}^{t} l_i = |S|.$$

Работа [10] содержит описание типичной для решения задачи 1.1 структуры обобщенных моделей Маркова. Множество скрытых состояний системы состоит из четырех элементов:

$$Q = \{St, F, Ex, In\},$$

где через St и F обозначены начальное и конечное состояния соответственно, состояние Ex отвечает экзонам, состояние In — интронам. Для повышения эффективности модели рассматриваются кодоны — известно, что интроны могут разрывать кодон, поэтому более точной является модель с тремя состояниям для интронов In_1, In_2, In_3, в зависимости от остатка от деления длины интрона на 3. Поскольку порядок скрытых состояний модели задан априорно, из формулы (1.3) исчезает часть $P(\mathbf{q})$.

Особенностью алгоритмов для решения задачи 1.1 является сокращение перебора по возможным разбиениям $\mathbf{l}$: с помощью методов машинного обучения определяются короткие последовательности нуклеотидов (*сигналы*), потенциально отвечающие переходам между экзонами и интронами. Эта эвристика согласуется с экспериментальными фактами:

- большинство интронов в генах высших организмов начинается с последовательности нуклеотидов аденин–гуанин (AG) и заканчивается последовательностью гуанин–тимин (GT);
- первый экзон гена всегда начинается с цепочки ATG;
- ген заканчивается одним из трех стоп-кодонов — TAA, TAG или TGA.

¶ **1.2.1. Оценка параметров модели.** В работе [11] рассматривается применение моделей Маркова со скрытыми состояниями и их обобщений для эффективного решения задачи нахождения генов прокариотов (организмов, клетки которых не содержат ядра). Для обучения вероятностей генерации $\theta(S \mid q_i)$ предлагается двухфазовый метод ожидания-максимизации (EM-метод) [12], его целью является нахождения точки максимума (возможно, локального) для вероятности $P(S \mid \theta)$. В случае обобщенных моделей Маркова EM-метод носит название алгоритма Баума — Уэлша [13]. Для более эффективного применения этого алгоритма в работе [11] предлагается использовать модели Маркова с интервальной интерполяцией, которые позволяют отслеживать связи между близко расположенными нуклеотидами.

¶ **1.2.2. Обзор существующих моделей.**

¶ **1.2.2.1. GeneZilla.** Основной особенностью алгоритма GeneZilla [10] является использование эффективной модели вычислений выражения (1.3) на основе метода динамического программирования, что позволяет обойтись линейным объемом использованной оперативной памяти. Доказано, что этот подход эквивалентен традиционному методу вычисления (1.3) с помощью массивов префиксных сумм [14], при применении которого объем памяти алгоритма линейно зависит не только от длины генома, но и от числа состояний модели.

¶ **1.2.2.2. GlimmerHMM, TigrScan.** В работе [15] сравниваются два алгоритма нахождения генов и их структуры — GlimmerHMM и TigrScan. В то время как структура моделей Маркова в этих алгоритмах похожа на описанную выше, существуют некоторые различия:

- Модель GlimmerHMM заметно упрощена по сравнению с TigrScan и GeneZilla — в ней присутствуют три скрытых состояния: экзон, интрон и участок между генами.
- TigrScan позволяет создавать графическое представление для своих моделей, способствуя их эффективному анализу человеком.

¶ **1.2.2.3. Augustus.** Особенностью алгоритма Augustus [16] является более точное распределение интронов по длинам $\varphi(d \mid In)$, включающее в себя, помимо гипергеометрического распределения, эмпирическое распределение для короткого начального участка интрона. Это позволяет повысить точность обнаружения интронов при сохранении приемлемой скорости классификации. Другим улучшением Augustus является использование нескольких вероятностных моделей для различных длин интронов, что согласуется с биохимическими наблюдениями [17].

¶ **1.2.2.4. SNAP.** Алгоритм SNAP [18] позиционируется для обнаружения генов в новых геномах, для которых использование моделей, обученных на других организмах, может привести к значительному снижению эффективности. Улучшение качества достигается за счет использования более гибкой структуры модели, позволяющей выделить слабо меняющиеся от генома к геному параметры. Еще одна особенность SNAP — использование шести интронных состояний вместо трех, что позволяет избежать появления в выводе алгоритма генов, содержащих стоп-кодон в середине кодирующей последовательности.

¶ **1.2.2.5. Twinscan.** Основным отличием Twinscan [19] от алгоритмов, рассмотренных выше, является использование понятия родства между генами разных биологических видов. Большинство генов произвольного организма с незначительными вариациями присутствует в геномах родственных организмов. Выровняв нуклеотидные последовательности в геномах с помощью методов динамического программирования (например, BLAST), можно, таким образом, достичь более высокого качества классификации, чем при использовании генома одного вида.

Эффективность Twinscan продемонстрирована в [19] на примере определения генов человека; в качестве родственного организма используется мышь. Применение родственных связей заметно повышает количество точно распознанных экзонов — с ~ 15% до ~ 25%.

¶ **1.2.2.6. GenomeScan.** Так же, как и Twinscan, GenomeScan [20] объединяет два подхода к определению генов: статистический, использующий априорные особенности экзонов и интронов (напр., содержание нуклеотидов), и сравнительный, ищущий в геноме фрагменты, подобные уже известным генам других организмов. GenomeScan работает в два этапа:

1) ищутся потенциальные экзоны с использованием методов выравнивания последовательностей;
2) найденные участки конвертируются в вероятностные оценки, служащие входными данными для модели Маркова со скрытыми переменными, которая проводит их фильтрацию.

За счет своей структуры GenomeScan наиболее эффективен, если требуется отсутствие ложноположительных результатов (псевдогенов).

¶ **1.2.3. Эффективность существующих алгоритмов.** Работа [21] приводит результаты тестирования шести алгоритмов для поиска генов — Augustus, GeneZilla, GenomeScan, GlimmerHMM, SNAP и Twinscan. Для тестирования используются шесть основных критериев качества:

- четыре критерия, связанные с качеством распознавания отдельных нуклеотидов — нуклеотидные специфичность и чувствительность, коэффициент корреляции и средняя условная вероятность;
- два критерия, связанные с качеством распознавания переходов между экзонами и интронами — экзонные специфичность и чувствительность.

Согласно предложенным критериям, исследуемые алгоритмы показывают практически одинаковый уровень компетентности — ~ 90% для мер качества, связанных с отдельными нуклеотидами, и ~ 50% для мер, связанных с экзонами.

§ 1.3. Методы определения вторичной структуры белка

Алгоритмы предсказания пространственной структуры белка с момента своего зарождения в 70-х годах XX века претерпели значительные изменения [22]:

- Методы первого поколения (напр., алгоритм Чоу — Фасмана) основывались на оценках вхождения отдельных аминокислот в различные типы пространственных структур. Точность методов этого типа составляла около 50%.
- Методы второго поколения (напр., GOR-метод) использовали предположение о том, что вероятность вхождения аминокислоты во вторичные структуры завит не только от нее, но и от ее окружения, основываясь на байесовских моделях. Такие методы позволили предсказывать вторичную структуру белка с точностью до 65%.
- Методы третьего поколения, актуальные на сегодняшний день, используют для предсказания различные методы машинного обучения, такие как нейронные сети и SVM. Точность методов третьего поколения составляет 75–80%.

Для повышения устойчивости классификации методы третьего и, в некоторых случаях, второго поколения, как и в задаче предсказания структуры генов, используют сравнительный подход, ища известные белки, обладающие сходным с рассматриваемым белком строением.

¶ **1.3.1. Метод Чоу — Фасмана.** Алгоритм, предложенный в работах [23–25], использует для предсказания вторичной структуры эмпирически полученные оценки вероятностей участия каждой из двадцати базовых аминокислот в спиралях, β-листах и поворотах. Спирали и листы устанавливаются путем линейного поиска в цепочке аминокислот, составляющих белок, центров образования соответствующих структур. Эти центры определяются как последовательности из четырех оснований, обладающие наивысшей совместной условной вероятностью вхождения в определенную вторичную структуру. После нахождения всех центров, они расширяются за счет добавления в них соседствующих аминокислот до тех пор, пока вероятность вхождения последовательности из четырех оснований в рассматриваемую структуру не станет меньше априорно заданного порогового значения.

Сравнительно высокая эффективность алгоритма основывается на описанном в параграфе 1.1.2 экспериментальном наблюдении о неравномерном вхождении различных аминокислот в пространственные образования. Вместе с тем, в более поздних работах показано, что первоначальные параметры алгоритма являются статистически ненадежными [26], он склонен к завышению количества аминокислот в спиралях и занижению — в β-слоях [27]. Даже с уточненными параметрами и некоторыми другими модификациями точность алгоритма Чоу — Фасмана уступает методам третьего поколения [28].

¶ **1.3.2. GOR-метод.** Метод, описанный в работах [29; 30], применяет для предсказания вторичной структуры окна длиной 17 аминокислот. Оценки вероятностей принадлежности средней, девятой, аминокислоты каждому из трех базовых классов пространственных структур получаются с использованием трех позиционных весовых матриц (*position-specific scoring matrix*) размером 20×17. Для отсеивания ложных положительных классификаций аминокислот в качестве спиралей и слоев устанавливаются ограничения на минимальную длину структур — четыре и два основания, соответственно.

Современные методы, использующие весовые матрицы, повышают качество классификации за счет использования сведений о подобных (гомологичных) белках при помощи методов выравнивания [31] наподобие BLAST [32] и/или за счет анализа групп белков с неизвестной структурой на предмет сохраняемых фрагментов и вариаций [33–35].

¶ **1.3.3. Методы на основе нейронных сетей.** Многослойные нейронные сети [36] являются на сегодняшний день одним из наиболее точных методов предсказания вторичной структуры белков. Входными данными для сети являются составленные для каждого основания белка весовые матрицы размером $20\times m$ или $21\times m$, где m — длина окна, включающего соседей рассматриваемой аминокислоты. Ряды матриц соответствуют аминокислотам, а также, в случае использования матриц размера $21\times m$, специальному символу, обозначающему окончание белка. Последовательности для матриц формируются за счет выравнивания гомологичных белков с помощью PSI-BLAST [37] или подобных алгоритмов.

Архитектура сетей задается из априорных соображений; так, в [38] рассматривается многослойный персептрон; в [39] — многослойная сеть с фиксированным строением составляющих уровней; в [40] — ансамбли сетей. Присущий нейронным сетям недостаток — большое количество параметров и связанное с этим сильное переобучение — решается использованием дополнительного уровня сети, выделяющего информативные признаки [41].

¶ **1.3.4. Методы на основе SVM.** Метод опорных векторов (*support vector machine* — SVM) [42; 43] является одним из лучших методов классификации при наличии большого числа признаков или в случае малого объема обучающей выборки. При использовании с входными данными, сгенерированными подобно данным для нейронных сетей, метод опорных векторов демонстрирует сравнимую точность классификации, обладая определенными преимуществами:

- более высокая скорость сходимости;
- меньшая склонность к переобучению;
- более легкая трактовка свободных параметров модели — опорных векторов.

Основные различия между алгоритмами, использующими SVM, заключаются в особенностях агрегации базовых бинарных классификаторов: в [44] используются деревья, построенные из априорных соображений; в [45] рассматривается, помимо них, голосование среди классификаторов для всех пар классов и для каждого из классов против двух остальных; [46] использует эвристические апостериорные оценки вероятностей, предложенные в работе [47]. Работа [48] рассматривает двухслойные модели, сокращающие число признаков, что приводит к повышению точности.

¶ **1.3.5. Методы на основе байесовских процедур.** В [49] для предсказания вторичной структуры белка используется байесовский подход в сочетании с нестационарными цепями Маркова высоких порядков (до четвертого включительно). В силу того, что вторичные структуры имеют достаточно большую длину, прогнозируются классы сразу для двух или трех оснований:

$$P(h_s h_{s+1} \mid S) = \frac{P(S \mid h_s h_{s+1})}{P(S)}; \tag{1.4}$$

$$P(h_{s-1} h_s h_{s+1} \mid S) = \frac{P(S \mid h_{s-1} h_s h_{s+1})}{P(S)}; \tag{1.5}$$

где S — последовательность аминокислот, составляющая белок, h_{s-1}, h_s,h_{s+1} — скрытые состояния для соответствующих аминокислот. После получения оценок (1.4) или (1.5) с помощью простых выкладок вычисляются оценки $P(h_s \mid S)$ для отдельных аминокислот; каждая аминокислота принадлежит той пространственной структуре, для которой максимальна соответствующая условная вероятность.

Как показывают результаты вычислительного эксперимента [50; 51], при использовании прогнозирования для троек соседних аминокислот по формуле (1.5) точность метода составляет порядка 75–80%, т. е. сравнима с точностью методов третьего поколения.

Теоретическое обоснование возможности применения байесовского подхода к рассматриваемой задаче дают работы [52–54]. В них исследована эффективность байесовской процедуры распознавания и классификации и даны оценки погрешности сверху и снизу этой процедуры, которые в существенном совпадают, т. е. доказана субоптимальность байесовского подхода. Эта процедура распознавания может применяться в случае независимости информационных признаков. Для моделей Маркова информационные признаки, соответствующие отдельным шагам, естественно считать независимыми при условии, что переходные вероятности цепи приблизительно равны. Применение указанной субоптимальной процедуры оправдано для задач, в которых это условие выполняется.

¶ **1.3.6. Методы на основе алгебраических процедур.** Потенциально для распознавания фрагментов генов и прогнозирования вторичной структуры белков могут быть использованы методы на основе алгебраических процедур [55–57]. Разработанный математический аппарат, основанный на использовании псевдообратных и проекционных операторов, а также теории их возмущения, позволяет в аналитической (алгебраической) форме находить функциональные преобразователи входной информации (признаков обьекта) с минимизацией результирующей невязки. Таким образом, данные функциональные преобразователи имеют перспективу использования при построении классификаторов в задачах распознавания образов, определении скрытых зависимостей в задачах прогнозирования, построении адаптивных регуляторов

для управления сложными системами; получены важные рекуррентные соотношения, основанные на формулах Гревилля, которые позволяют добавлять или удалять строки без пересчета исходной матрицы.

§ 1.4. Выводы к разделу 1

Рассмотренные обобщенные скрытые модели Маркова со скрытыми переменными для решения задачи 1.1 и методы машинного обучения для задачи 1.2 обладают определенными недостатками:

- чрезмерная специфичность моделей, ограничивающая возможность их расширения для решения сходных проблем, в частности, другой рассмотренной задачи;
- использование допущений эвристического характера, не объяснимых с точки зрения выбранного математического аппарата (напр., фиксированные маркеры начала и конца интронов для задачи распознавания фрагментов генов);
- сложный характер обучаемых параметров моделей, затрудняющий их трактовку.

Описанные недостатки обуславливают необходимость разработки общего подхода, позволяющего объединить решения задач 1.1 и 1.2.

РАЗДЕЛ 2

ВЕРОЯТНОСТНЫЕ МОДЕЛИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ

Чрезмерная специфичность рассмотренных в разделе 1 вероятностных моделей подталкивает к применению для решения задач 1.1 и 1.2 моделей, которые бы сочетали в себе обобщающую способность обыкновенных скрытых марковских моделей и эффективность обобщенных моделей Маркова со скрытыми переменными. Возможный способ такого объединения, рассматриваемый ниже, — использование марковских цепей высоких порядков с сохранением разделения состояний на наблюдаемые и скрытые.

§ 2.1. Математическая формулировка задачи распознавания последовательностей скрытых состояний

В обеих рассмотренных в разделе 1.1 задачах задана конечная последовательность символов S, отдельные элементы которой — S_i, $i = 1, \ldots, n$ — принадлежат конечному алфавиту R_s. В случае распознавания фрагментов генов в роли символов выступают отдельные нуклеотиды:

$$R_s = R_{nt} \equiv \{A, C, G, T\}.$$

Для задачи определения вторичной структуры белков наблюдаемым символам соответствуют аминокислоты; поскольку в формулировке задачи оговорена возможность наличия неопределенных аминокислот в белках, множество наблюдаемых состояний в этом случае состоит из 21 элемента:

$$R_s = \{A, C, D, E, \ldots, W, Y, X\} \equiv R_{aa} \cup \{X\}.$$

Пользуясь терминологией моделей Маркова, строка S отвечает цепочке наблюдаемых состояний.

Каждому наблюдаемому символу $S_i \in R_s$ соответствует строго один неизвестный (скрытый) символ H_i из конечного множества R_h. Для задачи поиска экзонов и интронов в генах скрытых символа два — $R_h = \{ex, in\}$, где через ex обозначены нуклеотиды, входящие в состав экзона, через in — нуклеотиды, принадлежащие интронам. Для задачи поиска вторичной структуры белка алфавит скрытых символов состоит из трех элементов: $R_h = \{\alpha, \beta, o\}$, где α обозначены аминокислоты из спиралей, β — аминокислоты, принадлежащие β-листам, o — оставшиеся основания, не входящие в состав регулярных образований. В терминах марковских моделей, строке скрытых символов $H \equiv H_1 H_2 \ldots H_n$ соответствует цепочка скрытых состояний $\mathbf{q} = q(1)q(2)\ldots q(t)$.

Пользуясь введенными обозначениями, сформулируем обобщение задач 1.1 и 1.2:

Задача 2.1. *Найти алгоритм $\mathcal{A} : R_s^* \to R_h^*$, который сопоставляет произвольной последовательности наблюдаемых символов $S \in R_s^n$ соответствующую ей цепочку скрытых символов той же длины $H \in R_h^n$ согласно определенному критерию качества $\mathcal{L}$:*

$$\mathcal{A}(S) = \arg\max_H \mathcal{L}(S, H).$$

Поскольку каждому наблюдаемому состоянию в рамках рассматриваемого подхода соответствует строго одно скрытое состояние (в отличие от обобщенных скрытых моделей Маркова, где каждое скрытое состояние порождает цепочку наблюдаемых символов), рассмотрим их пары $Q_i = (S_i, H_i)$ — *полные состояния*. Множество всех возможных полных состояний является декартовым произведением алфавитов наблюдаемых и скрытых символов: $R_q = R_s \times R_h$; его размер составляет $4 \cdot 2 = 8$ для задачи распознавания фрагментов генов и $21 \cdot 3 = 63$ в случае определения вторичной структуры белка. Для произвольной последовательности символов $Q \equiv Q_1 Q_2 \dots Q_n \in R_q^*$ определены проекции на множества наблюдаемых и скрытых состояний:

$$\begin{aligned} \mathrm{Pr}_s &: R_q^* \to R_s^*, \quad \mathrm{Pr}_s(Q) = S_1 \dots S_n; \\ \mathrm{Pr}_h &: R_q^* \to R_h^*, \quad \mathrm{Pr}_h(Q) = H_1 \dots H_n. \end{aligned}$$

Для обеспечения большей наглядности, в дальнейшем для задачи распознавания фрагментов генов будем использовать однобуквенные обозначения для полных состояний: нуклеотиды, принадлежащие экзонам, обозначим заглавными буквами, а нуклеотиды, входящие в интроны, — прописными. Таким образом,

$$\begin{gathered} (A, ex) \equiv A, \quad (C, ex) \equiv C, \quad (G, ex) \equiv G, \quad (T, ex) \equiv T, \\ (A, in) \equiv a, \quad (C, in) \equiv c, \quad (G, in) \equiv g, \quad (T, in) \equiv t, \end{gathered}$$

$$R_q = \{A, C, G, T, a, c, g, t\}.$$

С учетом введенных обозначений возможна следующая формулировка задачи 2.1:

Задача 2.2. *Определить алгоритм $\mathcal{A} : R_s^* \to R_q^*$, который восстанавливает последовательность полных состояний $Q \in R_q^n$ по ее известной проекции на множество наблюдаемых состояний $S = \mathrm{Pr}_s(Q) \in R_s^n$ согласно критерию $\mathcal{L}$.*

В рамках вероятностного подхода на алгоритм $\mathcal{A}$ налагаются следующие ограничения:

- Алгоритм должен обучаться на прецедентах, что означает зависимость используемых в алгоритме оценок вероятностей от определенного конечного набора последовательностей $T = \{Q_i\} \subset R_q^*$, для которых известны как наблюдаемые, так и скрытые строки состояний.
- Алгоритм должен максимизировать условную вероятность появления последовательности скрытых символов при заданной цепочке наблюдаемых символов (*принцип максимума правдоподобия*):

$$\mathcal{A}(S) = \arg\max_Q P(Q \mid S).$$

Поскольку

$$P(Q \mid S) = \frac{P(S \mid Q) P(Q)}{P(S)},$$

где, в силу определения, $P(S \mid Q) = [\mathrm{Pr}_s(Q) = S]$, а априорная вероятность $P(S)$ не зависит от переменной Q,

$$\mathcal{A}(S) = \arg\max_Q P(Q)[\mathrm{Pr}_s(Q) = S].$$

§ 2.2. Понятие симметрии в ДНК и белках

Обоснование возможности применения к решению задач 1.1 и 1.2 обыкновенных моделей Маркова можно получить с помощью исследования статистик, касающихся последовательностей нуклеотидов в ДНК и аминокислот в белках. С этой целью теоретически обоснуем связи между статистиками (*симметрии*), являющиеся следствием марковского свойства; их экспериментальное наблюдение служит аргументом в пользу рассматриваемых вероятностных моделей.

¶ **2.2.1. Определение симметрии в ДНК.** Из описанной в разделе 1.1.1 структуры ДНК следует симметричность вида $N_1(i) = N_2(\bar{i})$, где $i \in \{A, C, G, T\} \equiv R_{nt}$ — произвольный нуклеотид, $\bar{i}$ — комплементарный нуклеотид ($\bar{A} = T$, $\bar{C} = G$, $\bar{G} = C$, $\bar{T} = A$), N_1 и N_2 — количество вхождений нуклеотида в первую и вторую нить ДНК соответственно. Понятие комплементарности допускает обобщение для произвольных цепочек нуклеотидов:

Определение 2.1. *Для последовательности нуклеотидов $x \in R_{nt}^*$ обратно комплементарной называется последовательность $\bar{x} \in R_{nt}^*$, полученная путем замены каждого нуклеотида из x на комплементарный и обращения полученной строки: $\bar{x} \stackrel{\text{def}}{=} \overline{x_{|x|}x_{|x|-1}\dots x_1}$.*

С учетом определения 2.1, симметрия по нитям ДНК имеет общий вид

$$\forall x \in R_{nt}^* \quad N_1(x) = N_2(\bar{x})$$

или, если перейти к терминологии теории вероятностей,

$$\forall x \in R_{nt}^* \quad P_1(x) = P_2(\bar{x}), \tag{2.1}$$

где P_1, P_2 — вероятности вхождения цепочки нуклеотидов в соответствующую нить ДНК.

В работах [58; 59] было высказано следующее предположение:

Гипотеза 2.1. *Помимо симметрии вида (2.1), в последовательностях нуклеотидов, образованных ДНК, наблюдается также аналогичная симметрия в пределах каждой из нитей:*

$$\forall x \in R_{nt}^* \quad P_1(x) = P_1(\bar{x}), P_2(x) = P_2(\bar{x}). \tag{2.2}$$

В частности, для отдельных оснований выполняются отношения $P_s(A) = P_s(T)$, $P_s(C) = P_s(G)$, где $s \in \{1, 2\}$.

Замечание. Из формул (2.1) и (2.2) следует

$$\forall x \in R_{nt}^* \quad P_1(x) = P_1(\bar{x}) = P_2(x), \tag{2.3}$$

т. е. вероятности встречи нуклеотидных последовательностей одинаковы по обеим нитям ДНК. В связи с этим в дальнейших выкладках нижние индексы у вероятностей P и количеств n цепочек нуклеотидов будут опущены.

Гипотеза 2.1 подтверждается результатами исследований генетического кода человека и других организмов [60; 61]. Для всех исследованных последовательностей ДНК отношения (2.2) выполняются с точностью порядка 1%. Причины симметрии в пределах каждой нити ДНК на

настоящий момент не вполне ясны; вероятно, одно из оснований для такой симметрии — необходимость одинакового веса обоих нитей для большей устойчивости этой молекулы.

В предположении о том, что последовательность нуклеотидов обладает марковским свойством, т. е. для нее справедливо равенство

$$P(x) = P(x_1)P(x_2 \mid x_1) \dots P(x_{|x|} \mid x_{|x|-1}),$$

вышеприведенная гипотеза выводится из более общих оснований.

Теорема 2.1. *Если для последовательности ДНК соблюдается симметрия по основаниям и парам оснований, т. е. выполняются равенства*

$$\forall i, j \in R_{nt} \quad P(i) = P(\bar{i}), P(ij) = P(\bar{j}\bar{i}),$$

а сама последовательность удовлетворяет марковскому свойству, то для нее выполняются отношения (2.2).

Доказательство.

Проведем доказательство по индукции, начав с цепочек нуклеотидов длиной три, т. е. с доказательства равенства

$$P(ijk) = P(\bar{k}\bar{j}\bar{i}). \tag{2.4}$$

Поскольку для последовательности справедливо свойство Маркова, то

$$P(ijk) = P(ij)P(k \mid j), \quad P(\bar{k}\bar{j}\bar{i}) = P(\bar{k}\bar{j})P(\bar{i} \mid \bar{j}).$$

Исходя из определения условной вероятности, $P(k \mid j) = P(jk)/P(j)$, откуда

$$P(ijk) = \frac{P(ij)P(jk)}{P(j)}.$$

Аналогично,

$$P(\bar{k}\bar{j}\bar{i}) = \frac{P(\bar{k}\bar{j})P(\bar{j}\bar{i})}{P(\bar{j})}.$$

Поскольку из условия теоремы $P(j) = P(\bar{j})$, $P(jk) = P(\bar{k}\bar{j})$, $P(ij) = P(\bar{j}\bar{i})$, равенство (2.4) действительно справедливо.

Далее, предположим, что для всех возможных цепочек длины l формула (2.2) справедлива. Рассмотрим произвольную цепочку длины $l + 1$:

$$P(x_1x_2 \dots x_lx_{l+1}) = P(x_1 \dots x_l)P(x_{l+1} \mid x_l) = \frac{P(x_1 \dots x_l)P(x_lx_{l+1})}{P(x_l)}; \tag{2.5}$$

$$P(\bar{x}_{l+1}\bar{x}_l \dots \bar{x}_2\bar{x}_1) = P(\bar{x}_{l+1}\bar{x}_l)P(\bar{x}_{l-1} \dots \bar{x}_1 \mid \bar{x}_l) = \frac{P(\bar{x}_{l+1}\bar{x}_l)P(\bar{x}_l \dots \bar{x}_1)}{P(\bar{x}_l)}, \tag{2.6}$$

откуда следует, что $P(x_1x_2 \dots x_lx_{l+1}) = P(\bar{x}_{l+1}\bar{x}_l \dots \bar{x}_2\bar{x}_1)$, т. е. формула (2.2) выполняется и для цепочек длины $l + 1$. Таким образом, теорема доказана. ∎

Теорема 2.1 обобщается на случай, когда последовательность оснований соответствует марковской цепи произвольного порядка:

Теорема 2.2. *Пусть последовательность ДНК описывается цепью Маркова m-го порядка, т. е. вероятность вхождения в последовательность нуклеотида зависит исключительно от m предыдущих нуклеотидов. Пусть также соблюдается симметрия по цепочкам оснований длины m и $m+1$, то есть для любой строки такой длины вероятность ее обнаружения в нити ДНК совпадает с вероятностью для обратно комплементарной последовательности: $P(x) = P(\bar{x})$ при $|x| = m$ или $|x| = m+1$. Тогда симметрия подобного вида выполняется для всех цепочек длины $l \geqslant m$.*

Доказательство.

Так же как и в прошлый раз, будем вести доказательство по индукции. Базисом будет сформулированное в условии теоремы утверждение о симметрии для цепочек длины $m+1$. Предположим, что утверждение выполняется для всех цепочек некоторой длины $l > m+1$.

По аналогии с (2.5) и (2.6) справедливы отношения:

$$P(x_1x_2\ldots x_lx_{l+1}) = P(x_1\ldots x_l)P(x_{l+1}\,|\,x_{l-m+1}\ldots x_l) = \frac{P(x_1\ldots x_l)P(x_{l-m+1}\ldots x_{l+1})}{P(x_{l-m+1}\ldots x_l)};$$

$$P(\bar{x}_{l+1}\bar{x}_l\ldots\bar{x}_2\bar{x}_1) = P(\bar{x}_{l+1}\ldots\bar{x}_{l-m+1})P(\bar{x}_{l-m}\ldots\bar{x}_1\,|\,\bar{x}_l\ldots\bar{x}_{l-m+1}) = \\ = \frac{P(\bar{x}_{l+1}\ldots\bar{x}_{l-m+1})P(\bar{x}_l\ldots\bar{x}_1)}{P(\bar{x}_l\ldots\bar{x}_{l-m+1})}.$$

Так как

$$P(\bar{x}_l\ldots\bar{x}_{l-m+1}) = P(x_{l-m+1}\ldots x_l);$$

$$P(\bar{x}_{l+1}\ldots\bar{x}_{l-m+1}) = P(x_{l-m+1}\ldots x_{l+1});$$

из условия теоремы, $P(\bar{x}_l\ldots\bar{x}_1) = P(x_1\ldots x_l)$ по предположению, то

$$P(\bar{x}_{l+1}\ldots\bar{x}_1) = P(x_1\ldots x_{l+1}),$$

т. е. теорема справедлива для цепочек длины $l+1$. ∎

¶ **2.2.2. Определение симметрии в белках.** Поскольку последовательности ДНК описывают все белки организма, наличие симметрии по отношению к нуклеотидам должно влиять на закономерности строения белков. Вместе с тем, логический переход от нуклеотидов к аминокислотам, несмотря на наличие жесткого кода, использующегося при синтезе, затруднен из-за сложной структуры ДНК — лишь небольшая ее часть используется для кодирования белков, а гены включают не применяемые при кодировании участки. Поэтому наиболее естественными объектами для исследования симметрии в белках являются бактерии, геном которых имеет простое строение и не содержит интронов.

Пусть $T_1, T_2 \subset R^*_{aa}$ — множества, состоящие из белков, получаемых при трансляции генов, расположенных на соответствующих нитях ДНК. В предположении, что участки, кодирующие белки, занимают бóльшую часть последовательности ДНК, из соотношения (2.3) для троек нуклеотидов следует симметрия по отдельным аминокислотам и их цепочкам:

Гипотеза 2.2. *Для совокупности белков, синтезируемых организмом, выполняется отношение*

$$N_1(x) \approx N_2(x),$$

где $x \in R^*_{aa}$ — последовательность аминокислот, N_1 и N_2 — число ее вхождений в белки, синтезируемые из генов, расположенных на соответствующей нити ДНК:

$$N_s(x) \stackrel{\text{def}}{=} \sum_{Q \in T_s} \sum_i [Q_{i \ldots i+|x|-1} = x], \quad s \in \{1, 2\}.$$

Так как суммарные длины белков, кодируемых генами на разных нитях ДНК, не совпадают, между числом вхождений и вероятностью нельзя провести прямое соответствие. Тем не менее, логично ожидать выполнение следующего предположения:

Гипотеза 2.3. *Пусть $x \in R^*_{aa}$ — произвольные аминокислоты, f_s — частота вхождения последовательности на нити $s \in \{1, 2\}$, т. е. несмещенная оценка вероятности*

$$f_s(x) \stackrel{\text{def}}{=} N_s(x) / \sum_{Q \in T_s} |Q|.$$

Тогда выполняется отношение

$$f_1(x) \approx f_2(x).$$

Помимо предположений, связанных с вероятностями появления в белках отдельных аминокислот, выведем соотношения, связывающие их в определенные группы [62]. Для этого рассмотрим, какие аминокислоты кодируются обратно комплементарными тройками нуклеотидов. Эти сведения допускают наглядную интерпретацию в форме графа, вершинами которого являются отдельные аминокислоты, а каждое ребро связывает собой аминокислоты, синтезируемые определенной парой обратно комплементарных кодонов (рис. 2.1–2.3). Три пары аминокислот (глутамин–лейцин, серин–аргинин, пролин–аргинин), а также пара L–$stop$ в таком графе связаны двумя ребрами.

Исходя из гипотезы 2.1, сформулируем предположение о количестве вхождений в белки аминокислот из каждого из трех полученных связных подграфов:

Гипотеза 2.4. *Для совокупности белков организма выполняются отношения*

$$N(D) + N(H) + N(N) + N(Y) \approx N(I) + N(M) + N(V);$$

$$N(E) + N(K) + N(Q) + N(stop_1) \approx N(F) + N(L);$$

$$N(C) + N(G) + N(R) + N(S) + N(W) \approx N(A) + N(P) + N(T) + N(stop_2),$$

где $N(x) = N_1(x) + N_2(x)$ — суммарное число вхождений аминокислоты во все белки, синтезируемые организмом.

Замечание. Третье соотношение не является полностью точным, поскольку граф отношений комплементарности между аминокислотами, изображенный на рис. 2.3, в отличие от графов на рис. 2.1 и 2.2, не является двудольным.

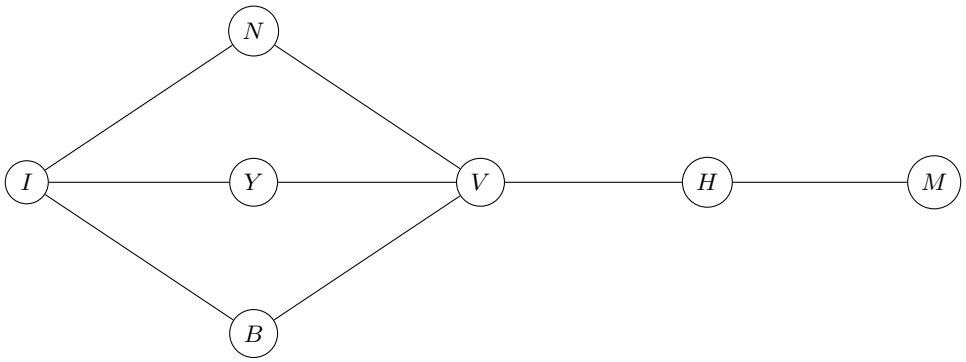


Рис. 2.1. Соотношения комплементарности для группы аминокислот $\{B, H, I, M, N, V, Y\}$

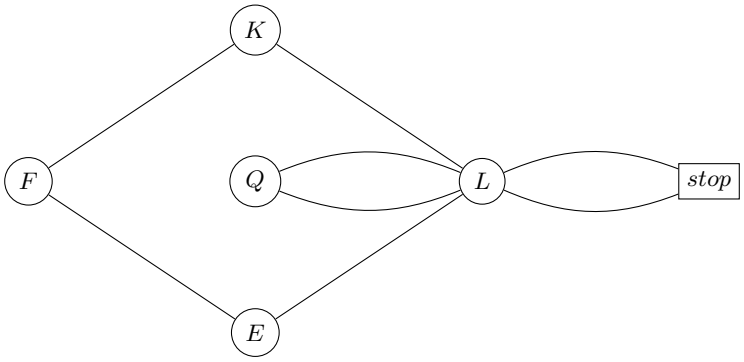


Рис. 2.2. Соотношения комплементарности для группы аминокислот $\{E, F, K, L, Q\}$ и двух стоп-кодонов

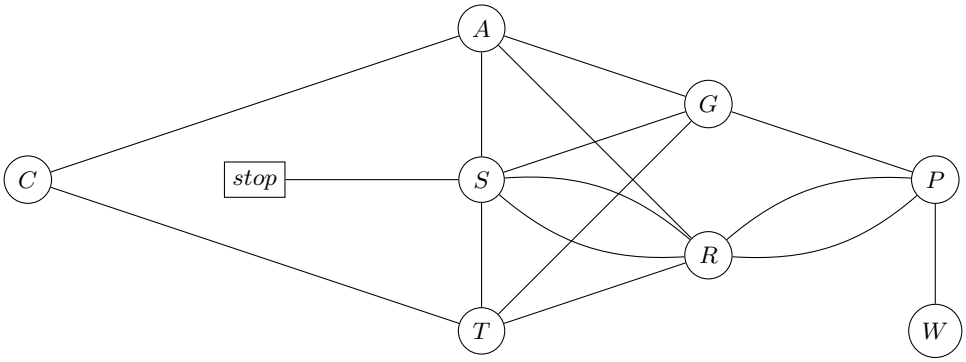


Рис. 2.3. Соотношения комплементарности для аминокислот A, C, G, P, R, S, T, W и стоп-кодона

§ 2.3. Адаптация модели Маркова для распознавания последовательностей скрытых состояний

Построим описанный в разделе 2.1 алгоритм $\mathcal{A} : R_s^* \to R_q^*$, используя обыкновенные скрытые модели Маркова, для которых каждое скрытое состояние порождает в точности одно наблюдаемое состояние.

Пусть $\mathcal{M}(1)$ — марковская модель, в которой скрытыми состояниями являются элементы алфавита полных состояний R_q, наблюдаемыми состояниями — символы из алфавита наблюдаемых состояний R_s. Вероятности порождения наблюдаемых состояний в такой модели имеют простой вид

$$\theta(s \mid x) = [\mathbf{Pr}_s(x) = s], \quad s \in R_s, x \in R_q,$$

т. е. каждое скрытое состояние порождает строго одно наблюдаемое состояние, определенное как проекция скрытого состояния на алфавит R_s (рис. 2.4). Остальные параметры модели $\mathcal{M}(1)$, а именно начальные вероятности $\pi(x)$, $x \in R_q$, и переходные вероятности $p(y \mid x)$, $x, y \in R_q$, в соответствии с приведенным в разделе 2.1 требованием к алгоритму $\mathcal{A}$, могут быть получены с помощью обучающей выборки. При этом алгоритм Баума — Уэлша, который используется для оценки неизвестных параметров марковской модели, существенно упрощается по сравнению с общим случаем в силу простого вида вероятностей порождения $\theta(s \mid x)$.

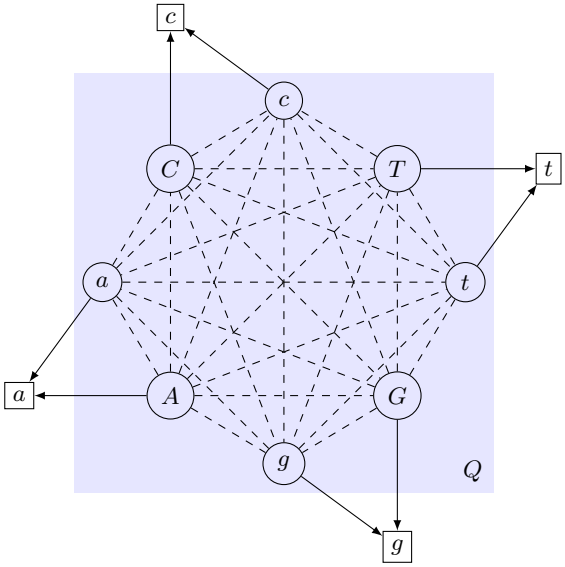


Рис. 2.4. Структура скрытой марковской модели $\mathcal{M}(1)$ для задачи определения фрагментов генов

Оставаясь в рамках обыкновенных скрытых марковских моделей, обобщим вышеописанную модель на случай, когда каждому скрытому и наблюдаемому состоянию соответствует не один символ из алфавита R_h, а два или более символа. Полученная в результате модель $\mathcal{M}(l)$ обладает следующими характеристиками:

- множество наблюдаемых состояний R_s^l;
- множество скрытых состояний R_q^l;

- распределение генерируемых моделью последовательностей состояний по длинам φ;
- вероятности генерации наблюдаемых состояний

$$\theta(o \mid q) = [\Pr_s(q) = o], \quad o \in R_s^l, q \in R_q^l.$$

Каждое наблюдаемое состояние модели может порождаться $|R_h|^l$ скрытыми состояниями. Начальные и переходные вероятности, которых в модели $\mathcal{M}(l)$ насчитывается $|R_q|^l$ и $|R_q|^{2l}$ соответственно, по-прежнему находятся из обучающей выборки.

Для того чтобы перейти к построению алгоритма $\mathcal{A}$, сделаем следующее предположение о наблюдаемых цепочках символов:

Гипотеза 2.5. *Последовательности наблюдаемых символов $S \in R_s^*$ порождаются моделью $\mathcal{M}(l)$.*

В рамках гипотезы 2.5 задача 2.2 сводится к следующему виду:

Задача 2.3. *Определить цепочку скрытых состояний $\mathbf{q} \equiv q(1)q(2)\dots q(t)$ модели $\mathcal{M}(l)$, которая порождает заданную строку $S \in R_s^*$ с максимальной условной вероятностью $P(\mathbf{q} \mid S)$.*

Замечание. Поскольку на длину строки S не накладывается ограничений, при длине генерируемых цепочек $l > 1$ возможен случай, когда эта строка не может быть порождена моделью $\mathcal{M}(l)$. На практике используемые значения l достаточно малы, так что последние скрытые символы можно без существенного снижения качества алгоритма задать произвольно. Для обеих рассматриваемых задач эти символы можно восстановить, пользуясь знаниями из предметной области: для генов последние несколько нуклеотидов практически всегда принадлежат к последнему экзону; последние аминокислоты белка обычно не принадлежат к спиралям или β-листам. Исходя из этого, в дальнейшем будем полагать, что длина наблюдаемой строки кратна длине скрытых состояний модели: $|S| = lt$, где t — натуральное число.

Согласно формулам вероятности генерации последовательности наблюдаемых состояний (1.1), задача нахождения оптимальной цепочки скрытых состояний $\mathbf{q}$ по известной наблюдаемой строке S сводится к отысканию экстремума выражения

$$P(S, \mathbf{q}) = \pi(q(1)) \prod_{i=2}^{t} p(q(i) \mid q(i-1)) \prod_{i=1}^{t} [\Pr_s(q(i)) = S_{il-l+1\dots il}].$$

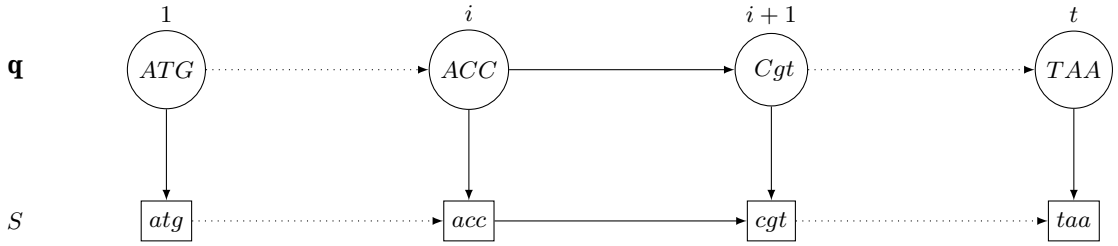
Чтобы избавиться от произведений, осуществим переход к логарифмам вероятности:

$$\log P(S, \mathbf{q}) = \log \pi(q(1)) + \sum_{i=2}^{t} \log p(q(i) \mid q(i-1)) + \sum_{i=1}^{t} [\![\Pr_s(q(i)) = S_{il-l+1\dots il}]\!]. \tag{2.7}$$

Так как

$$\log P(S, \mathbf{q}) = \log P(S_{1\dots lt-l}, q(1)\dots q(t-1)) + \log p(q(t) \mid q(t-1)) + [\![Pr_s(q(t)) = S_{lt-l+1\dots lt}]\!],$$

поставленная задача обладает ключевым свойством, позволяющим применить к ней метод динамического программирования: ее решение выражается через оптимальное решение задачи того же типа меньшего размера.



$$F(i+1, Cgt) = \max\big\{F(i, acc) + \log p(Cgt \mid acc),$$
$$F(i, acC) + \log p(Cgt \mid acC),$$
$$\vdots$$
$$F(i, ACC) + \log p(Cgt \mid ACC)\big\}.$$

Рис. 2.5. Структура модели $\mathcal{M}(3)$ для задачи распознавания фрагментов генов и схема подсчета для нее функции F

Введем обозначение

$$F(i, x) = \max \log P(S_{1\ldots il}, q(1)\ldots q(i) \mid q(i) = x),$$

где максимум берется по всевозможным состояниям $q(1), q(2), \ldots, q(i-1)$. В силу определения выполняется равенство

$$\max \log P(S, \mathbf{q}) = \max_{x \in R_q^l} F(t, x). \tag{2.8}$$

Из уравнений вида (2.7) выводится рекуррентная формула для функции F (рис. 2.5):

$$F(i, x) = [\![\Pr_s(x) = S_{il-l+1\ldots il}]\!] + \max_y \big(F(i-1, y) + \log p(x \mid y)\big), \tag{2.9}$$

где y — произвольное скрытое состояние модели ($y \in R_q^l$). Граничными равенствами для функции F являются

$$F(1, x) = [\![\pi_s(x) = S_{1\ldots l}]\!] + \log \pi(x). \tag{2.10}$$

С помощью формул (2.8)–(2.10) находится не только наибольшая совместная вероятность для цепочек наблюдаемых и скрытых состояний, но и сами скрытые состояния $q(1), q(2), \ldots, q(t)$: для этого достаточно при каждом вычислении функции F по формуле (2.9) запоминать переменную y, которая доставляет максимум выражению в правой части формулы.

Предложенный способ восстановления цепочки $\mathbf{q}$ является частным вариантом алгоритма Витерби с поправкой на специфический вид вероятностей генерации наблюдаемых состояний $\theta(o \mid q)$. Пошаговая запись алгоритма приведена в листинге 2.1.

Теорема 2.3. *Алгоритм 2.1 отказывается от классификации тогда и только тогда, когда вероятность наблюдения заданной цепочки символов $P(S)$ согласно модели $\mathcal{M}(l)$ равна нулю. В противном случае алгоритм возвращает цепочку состояний $\mathbf{q}$, которая максимизирует условную вероятность $P(\mathbf{q} \mid S)$.*

Алгоритм 2.1. Алгоритм Витерби для модели $\mathcal{M}(l)$.

Вход: последовательность S из символов алфавита R_s длины lt; вероятности вида $\pi(x)$ и $p(x \mid y)$, $x, y \in R_s^l$;

Выход: цепочка $\mathbf{q}$, для которой достигается максимум $P(\mathbf{q} \mid S)$;

1: **для всех** состояний $x \in R_s^l$ — Инициализация функции F
2: **если** $\Pr_s(x) = S_{1\ldots l}$, **то**
3: $F(1, x) := \log \pi(x)$;
4: **иначе**
5: $F(1, x) := -\infty$.
6: **для** $i = 2, \ldots, t$
7: **для всех** состояний $x \in R_s^l$
8: $F(i, x) := -\infty$;
9: **если** $\Pr_s(x) = S_{il-l+1\ldots il}$, **то**
10: **для всех** состояний $y \in R_s^l$
11: **если** $F(i, x) < \log p(x \mid y) + F(i-1, y)$, **то**
12: $ptr(i, x) := y$; — запомнить точку максимума
13: $F(i, x) := \log p(x \mid y) + F(i-1, y)$;
14: **если** $\max\limits_x F(t, x) = -\infty$, **то**
15: **вернуть** ε — отказ от классификации
16: **иначе**
17: $q(t) := \arg\max\limits_x F(t, x)$;
18: **для** $i = t, t-1, \ldots, 2$ — восстановление $\mathbf{q}$ по массиву ptr
19: $q(i-1) := ptr(i, q(i))$.

Доказательство.

Пусть вероятность появления цепочки S равна нулю. Поскольку

$$0 = P(S) = \sum_{\mathbf{q}} P(S, \mathbf{q}) \Rightarrow \forall \mathbf{q} \in R_q^{lt} \quad P(S, \mathbf{q}) = 0,$$

как следует из равенства (2.8), $\max\limits_x F(t, x) = -\infty$, т. е. приведенный выше алгоритм откажется от классификации.

Пусть $P(S) > 0$. Тогда существует одна или несколько цепочек скрытых состояний, для которых $P(S, \mathbf{q}) > 0$, следовательно, $\max\limits_x F(t, x) > -\infty$, то есть при работе алгоритма не будет исполнен шаг 15. По индукции можно убедиться в том, что все запоминаемые точки максимума $ptr(i, q(i))$, к которым производится доступ на шаге 19, будут определены корректно, так как значение $F(i, q(i)) \geqslant F(t, q(t)) > -\infty$ меняется по сравнению со значением, первоначально заданным на шаге 8. Таким образом, алгоритм завершит свою работу, полностью восстановив цепочку состояний $q(1) \ldots q(t)$, которая в соответствии с построением функции F является одним из решений задачи 2.3. ■

Основной вклад в оценку сложности алгоритма 2.1 дают три вложенных цикла, расположенных на шагах 6, 7 и 10. Первый из этих циклов будет выполняться $t - 1$ раз; второй и третий — по $|R_q|^l$ раз, однако за счет проверки условия на шаге 9 эффективная сложность цикла

по переменной x составляет $|R_h|^l$. Следовательно, вычислительная сложность алгоритма равна $O(t|R_h|^l|R_q|^l) = O(t|R_s|^l|R_h|^{2l})$; сложность линейным образом зависит от длины наблюдаемой строки S и экспоненциальным — от длины состояния модели l.

Подсчет функции $F(i, x)$, производимый на шаге 13 алгоритма 2.1, а также выполнение условия оператора ветвления на шаге 11 зависят от значений функции вида $F(i-1, \cdot)$. Исходя из этого, отсутствует необходимость хранить таблицу F целиком: при каждом выполнении тела цикла по переменной i в памяти достаточно хранить две ее строки, соответствующие i и $i-1$. Итак, объем памяти, потребляемый алгоритмом, определяется размером таблицы ptr. Поскольку эта таблица состоит из $t|R_q|^l$ элементов, каждый из которых является строкой длины l, потребление памяти алгоритмом составляет $\Theta(tl|R_q|^l)$.

Начальные и переходные вероятности для скрытых состояний в алгоритме 2.1 полагаются заданными. Несмещенные оценки этих вероятностей определяются с помощью обучающей выборки $T \subset R_q^*$:

$$\hat{\pi}(x) = \frac{N_{st}(T, x)}{|T|}, \quad \hat{p}(x \mid y) = \frac{N(T, yx)}{N(T, y)}, \quad x, y \in R_q^l,$$

где приняты следующие обозначения:

- $N_{st}(T, x) = |\{Q \in T \mid Q_{1\ldots l} = x\}|$ — общее число строк, начинающихся с заданной цепочки x;
- $N(T, x) = \sum\limits_{Q \in T} \sum\limits_{i} [Q_{i\ldots i+|x|-1} = x]$ — число вхождений строки x во все последовательности выборки.

§ 2.4. Обобщение модели Маркова для распознавания цепочек скрытых состояний

Рассмотрим последовательность полных состояний Q, являющуюся посимвольным декартовым произведением наблюдаемой строки S и цепочки скрытых состояний H. Гипотеза 2.5 утверждает, что вероятность встретить в Q определенную последовательность полных состояний длины l зависит от l предыдущих символов этой строки, то есть в рамках этой гипотезы вероятностное распределение символов строки Q задается вероятностями вида $P(Q_{i\ldots i+l-1} \mid Q_{i-l\ldots i-1})$. Избавимся от требования равенства длин зависимой и влияющей на нее частей строки, сформулировав следующее предположение:

Гипотеза 2.6. *Последовательности наблюдаемых состояний порождаются моделью $\mathcal{M}(l, m)$, в которой вероятность встречи в цепочке полных состояний Q любой последовательности длины m $Q_{i\ldots i+m-1}$ зависит исключительно от предыдущих l ее символов $Q_{i-l\ldots i-1}$.*

В дальнейшем будет полагаться, что $l \geqslant m$, т. е. цепочка полных состояний, влияющая на принадлежность рассматриваемой группы наблюдаемых символов определенному скрытому классу, не короче самой этой группы.

Модель $\mathcal{M}(l, m)$ характеризуется следующими параметрами:

- длина зависимой последовательности полных состояний m;
- длина предшествующей цепочки состояний, оказывающих влияние на зависимые состояния — l;

- распределение генерируемых цепочек по длине:

$$\varphi(i) \equiv P(|Q| = i), \quad \forall i \in \mathbb{N};$$

- начальные вероятности для строки Q:

$$\pi(x) \equiv P(Q_{1\ldots i} = x), \quad \forall i = 1, \ldots, l, x \in R_q^i;$$

- переходные вероятности

$$p(y \,|\, x) \equiv P(Q_{i\ldots i+m-1} = y \,|\, Q_{i-l\ldots i-1} = x), \quad x \in R_q^l, y \in R_q^m.$$

Общее число параметров модели составляет $O(|R_q|^l + |R_q|^{l+m}) = O(|R_q|^{l+m})$.

В рамках гипотезы 2.6 задача восстановления скрытой последовательности символов формулируется как

Задача 2.4. *Определить строку $Q \in R_q^*$, которая порождает цепочку наблюдаемых символов $S \in R_s^*$ с максимальной условной вероятностью $P(Q \,|\, S)$.*

Замечание. Как и в случае использования моделей $\mathcal{M}(l)$, установим ограничения на длину наблюдаемой строки S: будем полагать, что $|S| = l + mt$, где t — натуральное число. В отличие от предыдущего случая, это требование не является абсолютным и связано исключительно с упрощением выкладок.

Согласно гипотезе 2.6, вероятность появления последовательности полных состояний определяется выражением $P(Q) = \varphi(|Q|)P_n(Q)$, где

$$P_n(Q) = \pi(Q_{1\ldots l}) \prod_{i \in J(Q)} p(Q_{i\ldots i+m-1} \,|\, Q_{i-l\ldots i-1}); \tag{2.11}$$

$P_n(Q)$ — вероятность генерации строки Q при фиксированной ее длине. Множество индексов в (2.11) определено как $J(Q) = \{l + 1, l + m + 1, \ldots, |Q| - m + 1\}$.

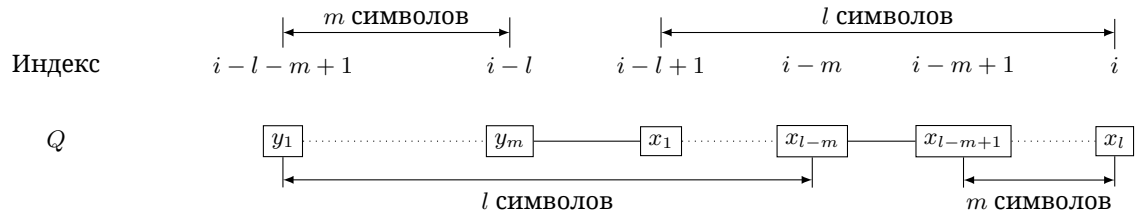
После перехода к логарифмам вероятности выражение (2.11) преобразуется к

$$\log P_n(Q) = \log \pi(Q_{1\ldots l}) + \sum_{i \in J(Q)} \log p(Q_{i\ldots i+m-1} \,|\, Q_{i-l\ldots i-1}). \tag{2.12}$$

Длина всех строк Q, генерирующих определенный вывод, одинакова; таким образом, задача 2.4 сводится к условной оптимизации

$$\log P_n(Q) \to \max_Q \quad \text{s.t. } \Pr_s(Q) = S.$$

Замечание. Равенство (2.11) соответствует порождающему подходу к вычислению вероятности: подразумевается, что строка Q генерируется некоторым алгоритмом, причем на каждом шаге, соответствующем индексу i, определяется подстрока Q длиной m на основе предшествующих l состояний. Возможен и другой, «аналитический» подход, при котором шаг индекса i в выражении (2.11) равен единице: $J(Q) = \{l + 1, l + 2, \ldots, |Q| - m + 1\}$. При значении длины зависимого фрагмента строки $m = 1$, которое, как будет показано в ходе вычислительного эксперимента, дает наилучшие результаты, оба выражения для вероятности $P_n(Q)$ совпадают.



Индекс	$i-6$	$i-5$	$i-4$	$i-3$	$i-2$	$i-1$	i
S	a	c	g	a	t	t	g
Q	?	?	g	A	t	T	G

$$F_a(i, gAtTG) = \max\big\{F_a(i-2, ACgAt) + \log p(TG \mid ACgAt),$$
$$F_a(i-2, AcgAt) + \log p(TG \mid AcgAt),$$
$$F_a(i-2, aCgAt) + \log p(TG \mid aCgAt),$$
$$F_a(i-2, acgAt) + \log p(TG \mid acgAt)\big\}.$$

Рис. 2.6. Общая схема и пример вычисления функции F_a для модели $\mathcal{M}(5, 2)$ в случае распознавания фрагментов генов

По аналогии с функцией F из раздела 2.3, для решения задачи 2.4 введем функцию

$$F_a(i, x) = [\![\Pr_s(x) = S_{i-l+1\ldots i}]\!] + \max_Q \log P_n(Q_{1\ldots i} \mid Q_{i-l+1\ldots i} = x), \tag{2.13}$$

которая связана с искомым максимумом соотношением

$$\max \log P_n(Q \mid S) = \max_{x \in R_q^l} F_a(|S|, x). \tag{2.14}$$

На основании формулы (2.12) для функции F_a получаем рекуррентный способ вычисления, аналогичный уравнению (2.9) (рис. 2.6):

$$F_a(i, x) = [\![\Pr_s(x) = S_{i-l+1\ldots i}]\!] + \max_y \big(F_a(i-m, yx_{1\ldots l-m}) + \log p(x_{l-m+1\ldots l} \mid yx_{1\ldots l-m})\big). \tag{2.15}$$

Максимум в данном случае берется по всем последовательностям длины m. Основное отличие (2.15) от (2.9) — вхождение состояния x в правой части уравнения не только в переходную вероятность, но и в функцию F_a.

Граничными равенствами для функции F_a являются

$$F_a(l, x) = [\![\Pr_s(x) = S_{1\ldots l}]\!] + \log \pi(x). \tag{2.16}$$

С учетом введенного выше условия $|S| = l + mt$, именно такого рода равенства будут актуальными при рекурсивном вычислении $F_a(|S|, x)$.

После замены формул (2.8)–(2.10), используемых в алгоритме 2.1, выражениями (2.14)–(2.16), получается алгоритм нахождения оптимальной строки полных состояний Q по заданной наблюдаемой строке S (листинг 2.2).

Теорема 2.4. *Алгоритм 2.2 отказывается от классификации строки S тогда и только тогда, когда ее вероятность наблюдения $P(S)$ согласно используемой модели $\mathcal{M}(l, m)$ равна нулю. В противном случае алгоритм возвращает одну из цепочек Q, которые максимизируют условную вероятность $P(Q \mid S)$.*

Алгоритм 2.2. Обобщение алгоритма Витерби для модели $\mathcal{M}(l, m)$

Вход: последовательность S из символов R_s, $|S| = l + mt$; вероятности вида $\pi(x)$, $x \in R_q^l$ и $p(y \mid x)$, $x \in R_q^l$, $y \in R_q^m$;

Выход: цепочка $Q \in R_q^*$, для которой достигается максимум $P(Q \mid S)$;

1: **для всех** строк $x \in R_q^l$ — Инициализация F_a
2: **если** $\Pr_s(x) = S_{1\ldots l}$, **то**
3: $F_a(l, x) := \log \pi(x)$;
4: **иначе**
5: $F_a(l, x) := -\infty$.
6: **для** $i = 1, \ldots, t$
7: **для всех** строк $x \in R_q^l$
8: $F_a(l + im, x) := -\infty$;
9: **если** $\Pr_s(x) = S_{im+1\ldots im+l}$, **то**
10: **для всех** строк $y \in R_q^m$
11: $head := x_{l-m+1\ldots l}$;
12: $tail := yx_{1\ldots l-m}$;
13: $prob := \log p(head \mid tail) + F_a(l + (i-1)m, tail)$;
14: **если** $F_a(l + im, x) < prob$, **то**
15: $ptr(i, x) := y$; — запомнить точку максимума
16: $F_a(l + im, x) := prob$;
17: **если** $\max\limits_x F_a(|S|, x) = -\infty$, **то**
18: **вернуть** ε — отказ от классификации
19: **иначе**
20: $Q_{tm+1\ldots tm+l} := \arg\max\limits_x F_a(|S|, x)$;
21: **для** $i = t, t-1, \ldots, 1$ — восстановление Q по массиву ptr
22: $Q_{im-m+1\ldots im} := ptr(i, Q_{im+1\ldots im+l})$.

Доказательство.

По аналогии с доказательством теоремы 2.3. ∎

Вычислительная сложность алгоритма 2.2 определяется тремя циклами, выполняющимися на шагах 6, 7 и 10 и составляет, с учетом проверки условия на 9-м шаге, $O(t|R_s|^l|R_h|^{l+m})$. Как и для алгоритма 2.1, сложность линейно зависит от длины исследуемой строки и экспоненциально — от параметров модели l и m.

Вероятности $\pi(x)$ и $p(y \mid x)$, которые полагаются в алгоритме 2.2 заданными, как и в случае моделей $\mathcal{M}(l)$, получаются с использованием обучающей выборки $T \subset R_q^*$. Исходя из формулы (2.12), оценки для начальных и переходных вероятностей вычисляются по формулам

$$\hat{\pi}(x) = \frac{|\{Q \in T \mid Q_{1\ldots l} = x\}|}{|T|} = \frac{N_{st}(T, x)}{|T|}; \tag{2.17}$$

$$\hat{p}(y \mid x) = \frac{\sum\limits_{Q \in T} \sum\limits_{i \in J(Q)} [Q_{i-l\ldots i-1} = x][Q_{i\ldots i+m-1} = y]}{\sum\limits_{Q \in T} \sum\limits_{i \in J(Q)} [Q_{i-l\ldots i-1} = x]}. \tag{2.18}$$

Множество индексов $J(Q)$, используемое в уравнении (2.18), соответствует его определению для (2.11):

$$J(Q) = \big\{l + 1 + jm \,|\, j = 0, 1, \ldots, \lfloor(|Q| - l - 1)/m\rfloor\big\}.$$

При использовании обозначения

$$N^{l,m}(T, x) = \sum_{Q \in T} \sum_{i \in J(Q)} [Q_{i-l \ldots i-l+|x|-1} = x], \tag{2.19}$$

формулу (2.18) можно записать в более кратком виде

$$\hat{p}(y \,|\, x) = \frac{N^{l,m}(T, xy)}{N^{l,m}(T, x)}.$$

Теорема 2.5. *Оценки* (2.17) *и* (2.18) *максимизируют вероятность обучающей выборки* T *согласно модели* $\mathcal{M}(l, m)$*, соответствуя принципу максимума правдоподобия:*

$$(\hat{\pi}, \hat{p}) = \arg\max_{\pi,p} P(T \,|\, \pi, p).$$

Доказательство.

В силу выпуклости функции $\log x$ перейдем к логарифмам вероятности:

$$\arg\max_{\pi,p} P(T \,|\, \pi, p) = \arg\max_{\pi,p} \log P(T \,|\, \pi, p).$$

Из равенства (2.12) следует

$$\log P(T \,|\, \pi, p) = \sum_{Q \in T} \log P(Q \,|\, \pi, p) = \sum_{Q \in T} \log \varphi(|Q|) + \sum_{Q \in T} \log \pi(Q_{1 \ldots l}) + \\ + \sum_{Q \in T} \sum_{i \in J(Q)} \log p(Q_{i \ldots i+m-1} \,|\, Q_{i-l \ldots i-1}). \tag{2.20}$$

Для двух сумм в выражении (2.20) произведем перегруппировку по последовательностям длины l и $l + m$, составленным из символов алфавита R_q:

$$\log P(T \,|\, \pi, p) = \sum_{Q \in T} \log \varphi(|Q|) + \sum_{|x|=l} N_{st}(T, x) \log \pi(x) + \sum_{|x|=l} \sum_{|y|=m} N^{l,m}(T, xy) \log p(y \,|\, x). \tag{2.21}$$

Здесь предполагается, что при $N_{st}(T, x) = 0$ или $N^{l,m}(T, xy) = 0$ соответствующий член суммы всегда равен нулю.

Задача максимизации выражения (2.21) является задачей условной оптимизации с ограничениями

$$\forall x \in R_q^l, y \in R_q^m \quad \pi(x) \geqslant 0, p(y \,|\, x) \geqslant 0;$$

$$\sum_{|x|=l} \pi(x) = 1; \quad \forall x \in R_q^l \sum_{|y|=m} p(y \,|\, x) = 1.$$

Таким образом, функция Лагранжа задачи равна

$$L(\pi, p, \lambda) = \log P(T \,|\, \pi, p) - \lambda_{st}\Big(\sum_{|x|=l} \pi(x) - 1\Big) - \sum_{|x|=l} \lambda_x \Big(\sum_{|y|=m} p(y \,|\, x) - 1\Big).$$

Приравнивая ее производную по переменным $\pi(x)$ нулю, получаем

$$\forall x \in R_q^l \quad \frac{N_{st}(T,x)}{\pi(x)} - \lambda_{st} = 0 \Leftrightarrow N_{st}(T,x) = \lambda_{st}\pi(x).$$

После суммирования последнего выражения по всем цепочкам x,

$$|T| = \sum_{|x|=l} N_{st}(T,x) = \lambda_{st} \sum_{|x|=l} \pi(x) = \lambda_{st},$$

откуда оптимальное значение переменных $\pi(x)$ равно

$$\pi^*(x) = \frac{N_{st}(T,x)}{\lambda_{st}} = \frac{N_{st}(T,x)}{|T|} = \hat{\pi}(x).$$

Аналогично, для переменных $p(y \mid x)$ оптимальное значение равно

$$p^*(y \mid x) = \frac{N^{l,m}(T,xy)}{\sum\limits_{|z|=m} N^{l,m}(T,xz)} = \frac{N^{l,m}(T,xy)}{N^{l,m}(T,x)} = \hat{p}(y \mid x),$$

что и требовалось доказать. ∎

§ 2.5. Спецификация структур данных для разработанных алгоритмов распознавания

Существенный недостаток алгоритма 2.2 — работа в цикле на шаге 10 со всеми возможными состояниями y, в то время как большая часть этих состояний заведомо приводит к значению рекуррентно вычисляемой функции F_a, равному отрицательной бесконечности. В самом деле, в силу определения этой функции достаточно рассматривать значения $y \in R_q^m$, для которых выполняется условие $\mathrm{Pr}_s(y) = S_{i-l-m+1\ldots i-l}$. Таким образом, в циклах по переменным x и y на них накладываются определенные ограничения, делающие рациональным переход от цепочек состояний к их представлению в виде целых чисел. Еще одной мотивацией подобного перехода является простота реализации и большая скорость работы при индексации хранимых в памяти массивов, отвечающих переменным F_a и ptr, по целому числу по сравнению с индексацией с помощью строк, наблюдаемая для большинства языков программирования.

Пусть на алфавитах R_s и R_h введены отношения порядка, $\prec_s$ и $\prec_h$ соответственно. Упорядочим строки из множества R_s^*:

$$\forall x, y \in R_s^* \quad x \prec_s y \Leftrightarrow \exists j : (\forall i < j x_i = y_i) \wedge \big((j > |x|) \vee (x_j \prec_s y_j)\big);$$

аналогично введем отношение порядка для строк скрытых символов. Последовательности полных состояний также могут быть упорядочены:

$$\forall x, y \in R_q^* \quad x \prec_q y \Leftrightarrow (\mathrm{Pr}_s(x) \prec_s \mathrm{Pr}_s(y)) \vee \big((\mathrm{Pr}_s(x) = \mathrm{Pr}_s(y)) \wedge (\mathrm{Pr}_h(x) \prec_h \mathrm{Pr}_h(y))\big).$$

Для описания строк, принадлежащих множеству R_q^*, с помощью целых чисел будем использовать набор из двух функций:

- $\rho(x) : R_q^* \to \mathbb{N}_0$ — порядковый номер (с отсчетом от нуля) последовательности $x \in R_q^*$ среди цепочек полных состояний, генерирующих тот же вывод, т. е. среди элементов множества $\{z \in R_q^* \mid \mathrm{Pr}_s(z) = \mathrm{Pr}_s(x)\}$, упорядоченных в алфавитном порядке;

- $\sigma(x) : R_q^* \to \mathbb{N}_0$ — порядковый номер строки $\Pr_s(x)$ среди наблюдаемых строк той же длины, другими словами, множества $R_s^{|x|}$.

Упорядоченная пара величин $\langle \rho(x), \sigma(x) \rangle$ или выражение $|R_s|^{|x|} \cdot \rho(x) + \sigma(x)$ однозначно задают цепочку x среди всех последовательностей, входящих во множество $R_q^{|x|}$. Следующие выражения характеризуют конкатенацию строк и фрагменты цепочки полных состояний:

$$\rho(xy) = \rho(x) \cdot |R_h|^{|y|} + \rho(y); \quad \sigma(xy) = \sigma(x) \cdot |R_s|^{|y|} + \sigma(y);$$

$$\rho(x_{i\ldots j}) = \lfloor \rho(x) / |R_h|^{|x|-j} \rfloor \bmod |R_h|^{j-i+1};$$

$$\sigma(x_{i\ldots j}) = \lfloor \sigma(x) / |R_s|^{|x|-j} \rfloor \bmod |R_h|^{j-i+1}.$$

Пример. Пусть рассматривается задача распознавания фрагментов генов, то есть

$$R_s = \{A, C, G, T\},\ R_h = \{ex, in\},\ R_q = \{A, a, C, c, G, g, T, t\},$$

порядок на множествах наблюдаемых и скрытых символов соответствует латинскому алфавиту:

$$A \prec_s C \prec_s G \prec_s T; \quad ex \prec_h in;$$

соответственно,

$$A \prec_q a \prec_q C \prec_q c \prec_q G \prec_q g \prec_q T \prec_q t.$$

Тогда $\rho(aCt) = 5$, поскольку строка aCt расположена на шестом месте среди цепочек из скрытых символов длины 3, которые порождают строку ACT:

$$ACT = \Pr_s(ACT) = \Pr_s(ACt) = \Pr_s(AcT) = \Pr_s(Act) =$$
$$= \Pr_s(aCT) = \Pr_s(\underline{aCt}) = \Pr_s(acT) = \Pr_s(act).$$

$\sigma(aCt) = 013_4 = 7$, так как до строки ACT в R_s^3 расположены цепочки состояний AAA, AAC, AAT, ACA, ACC и ACG.

С учетом введенных обозначений, после предложенной модификации алгоритма 2.2 циклы на шагах 7 и 10 вместо переменных x и y будут использовать величины $\rho(x)$ и $\rho(y)$ (такая замена позволяет избавиться от проверки условия на 9-м шаге алгоритма). Этот переход приведет к замене переменных $head$ и $tail$ на выражения

$$\rho_h \equiv \rho(head) = \rho(x) \bmod |R_h|^m,$$

$$\rho_t \equiv \rho(tail) = \lfloor \rho(x) / |R_h|^m \rfloor + \rho(y) \cdot |R_h|^{l-m}.$$

Значения $\sigma_h \equiv \sigma(head)$ и $\sigma_t \equiv \sigma(tail)$, которые наряду с величинами ρ_h и ρ_t требуются для определения вероятности $p(head \,|\, tail)$, можно определить по соответствующим символам строки S. Сама переходная вероятность является функцией четырех параметров: $p(head \,|\, tail) = \hat{p}(\rho_h, \sigma_h, \rho_t, \sigma_t)$; начальная вероятность, соответственно, зависит от двух переменных: $\pi(x) = \hat{\pi}(\rho_x, \sigma_x)$.

Остается нерешенным вопрос модификации переменной ptr. Вместо точки максимума y в рекуррентной формуле (2.15) достаточно запоминать $\rho(y)$, поскольку значение $\sigma(y)$ получается из наблюдаемой строки S. Для восстановления последовательности полных состояний Q по массиву сохраненных значений ptr введем функцию $\tau : R_s^* \times \mathbb{N}_0 \to R_q^*$, для которой выполняется тождество

$$\forall x \in R_q^* \quad \tau(\mathrm{Pr}_s(x), \rho(x)) \equiv x.$$

Таким образом, функция τ восстанавливает скрытую цепочку состояний по генерируемому ей выводу s и порядковому номеру цепочки во множестве $\mathrm{Pr}_s^{-1}(s)$.

Вычисление $\tau(s, \rho)$ производится по следующему алгоритму:

1) перевести число ρ в систему исчисления с основанием $|R_h|$:

$$\rho = \rho_1 |R_h|^{|s|-1} + \rho_2 |R_h|^{|s|-2} + \cdots + \rho_{|s|-1}|R_h| + \rho_{|s|};$$

2) сформировать цепочку состояний скрытых состояний $h = h_1 \ldots h_{|s|}$ так, чтобы символ h_i был $(\rho_i + 1)$-м по порядку в алфавите R_h;

3) на основе наблюдаемой и скрытой строк создать последовательность полных состояний: $\tau(s, \rho) = s \times \rho$.

Пример. Пусть рассматривается задача распознавания экзонов и интронов. Тогда в соответствии с данными, вычисленными в предыдущем примере, выполняется равенство

$$\tau(ACT, 5) = \tau(ACT, \rho(aCt)) = aCt.$$

Этот же ответ может быть получен с учетом равенств

$$5 = 101_2; \quad ACT \times (in, ex, in) = aCt.$$

Модифицированная с учетом изложенных выше предложений версия алгоритма для модели $\mathcal{M}(l, m)$ приведена в листинге 2.3. За счет трех вложенных циклов на шагах 3, 4 и 6 время его работы составляет $O(t|R_h|^{l+m})$, что на практике значительно меньше, чем время работы исходного алгоритма 2.2. Уменьшается и объем памяти, потребляемый алгоритмом: переменная ptr после оптимизации состоит из $t|R_h|^l$ целочисленных элементов; таким образом, память, необходимая для вычислений, составляет $\Theta(t|R_h|^l)$.

§ 2.6. Аппроксимация отсутствующих данных в обобщенной вероятностной модели

Одна из потенциальных проблем приведенных выше алгоритмов — нехватка статистики о некоторых начальных и переходных вероятностях в обучающей выборке. В алгоритмах 2.1–2.3 подразумевалось, что при неопределенности подобного рода оценки всех соответствующих переходных вероятностей равны нулю. Это приводит к отказу от классификации при подаче на вход алгоритма строки S, содержащей последовательность s длины l, которая не входит ни

Алгоритм 2.3. Оптимизированный алгоритм для модели $\mathcal{M}(l, m)$

Вход: последовательность S из символов R_s, $|S| = l + mt$;
вероятности вида $\hat{\pi} : \mathbb{N}_0^2 \to \mathbb{R}$ и $\hat{p} : \mathbb{N}_0^4 \to \mathbb{R}$;

Выход: цепочка $Q \in R_q^*$, для которой достигается максимум $P(Q \mid S)$;

для $\rho_x = 0, 1, \ldots, |R_h|^l - 1$ — Инициализация F_a
  $F_a(l, \rho_x) := \log \hat{\pi}_s(\rho_x, \sigma(S_{1\ldots l})$;
для $i = 1, \ldots, t$
  для $\rho_x = 0, 1, \ldots, |R_h|^l - 1$
    $F_a(l + im, \rho_x) := -\infty$;
    для $\rho_y = 0, 1, \ldots, |R_h|^m - 1$
      вычислить $\rho_h, \sigma_h, \rho_t, \sigma_t$ по ρ_x, ρ_y и S;
      $prob := \log \hat{p}(\rho_h, \sigma_h, \rho_t, \sigma_t) + F_a(l + (i-1)m, \rho_0)$;
      если $F_a(l + im, x) < prob$, **то**
        $ptr(i, \rho_x) := \rho_y$; — запомнить точку максимума
        $F_a(l + im, \rho_x) := prob$;
если $\max\limits_{\rho} F_a(|S|, \rho) = -\infty$, **то**
  вернуть ε — отказ от классификации
иначе
  $\rho_x := \arg\max\limits_{\rho} F_a(|S|, \rho)$;
  $Q_{tm+1\ldots tm+l} := \tau(S_{tm+1\ldots tm+l}, \rho_x)$;
  для $i = t, t-1, \ldots, 1$ — восстановление Q по массиву ptr
    $\rho_y := ptr(i, \rho_x)$;
    $Q_{im-m+1\ldots im} := \tau(S_{im-m+1\ldots im}, \rho_y)$;
    $\rho_x := \lfloor \rho_x / |R_h|^m \rfloor + \rho_y \cdot |R_h|^{l-m}$. — сдвинуть x на m символов влево

в одну из наблюдаемых частей строк из обучающей выборки:

$$\forall x \in R_q^l \quad (\Pr_s(x) = s) \Rightarrow N^{l,m}(T, x) = 0.$$

Таким образом, возникает вопрос об аппроксимации переходных вероятностей с помощью других статистик, полученных с помощью обучающей выборки T.

Для решения сформулированной проблемы рассмотрим три метода:

- использование значений, полученных с помощью априорных сведений;
- голосование по старшинству среди нескольких алгоритмов, упорядоченных по убыванию сложности;
- применение подобного подхода в рамках вычисления каждой переходной или начальной вероятности.

Поскольку, как будет показано в ходе вычислительного эксперимента (раздел 4.4), модели с длиной зависимой части $m = 1$ для обеих рассматриваемых задач наиболее эффективны, ниже будем рассматривать модели вида $\mathcal{M}(l, 1)$.

¶ **2.6.1. Априорная оценка вероятностей.** Наиболее простым методом аппроксимации начальных и переходных вероятностей, которые невозможно получить из обучающей выборки T, является использование малых значений, выбранных из априорных соображений:

$$\forall x \in R_q^l \quad (\forall Q \in T \quad Q_{1\ldots l} \neq x) \Rightarrow (\hat{\pi}(x) = \varepsilon_\pi);$$

$$\forall x \in R_q^l, y \in R_q \quad (\forall Q \in T\ \forall i \quad Q_{i\ldots i+l} \neq xy) \Rightarrow (\hat{p}(y \mid x) = \varepsilon_p).$$

Числа ε_π и ε_p подбираются с расчетом на то, чтобы при подсчете функции $F_a(i, x)$ согласно рекуррентной формуле (2.15) не отдавать предпочтения строкам x, отсутствующим в обучающей выборке. Вероятностную модель, соответствующую этому методу оценки, будем обозначать как $\mathcal{M}_\varepsilon(l, \varepsilon_\pi, \varepsilon_p)$.

¶ **2.6.2. Использование композиций моделей.** При голосовании по старшинству необходимо упорядочивание используемых алгоритмов от наиболее сложных к наиболее общим, чтобы каждая строка $S \in R_s^*$ классифицировалась с использованием наиболее специфичной модели, для которой обучающая выборка содержит достаточно данных. Для того чтобы упорядочить модели по их сложности, докажем следующее утверждение:

Теорема 2.6. *Пусть алгоритм $\mathcal{A}$, построенный на основе модели $\mathcal{M}(l, 1)$ и обучающей выборки $T \subset R_q^*$, не отказывается от классификации некоторой строки $S \in R_s^*$: $\mathcal{A}(S) \neq \varepsilon$. Тогда алгоритм $\mathcal{A}'$, построенный на основе модели $\mathcal{M}(l', 1)$, где $0 < l' < l$, и выборки T, также определенным образом классифицирует строку S: $\mathcal{A}'(S) \neq \varepsilon$.*

Доказательство.

Приведенное утверждение в силу логического тождества $(a \Rightarrow b) \equiv (\neg b \Rightarrow \neg a)$ эквивалентно следующему: из отказа классификации алгоритма $\mathcal{A}'$ следует отказ алгоритма $\mathcal{A}$.

Согласно теореме 2.4, отказ от классификации строки S алгоритмом $\mathcal{A}'$ происходит тогда и только тогда, когда вероятность генерации этой строки моделью $\mathcal{M}(l', 1) - P'(S) -$ равна нулю. Поскольку

$$P'(S) = \sum_{Q \in R_q^*} P(S \mid Q) P'(Q) = \sum_{Q \in R_q^*} P'(Q)[\mathrm{Pr}_s(Q) = S],$$

условие $P'(S) = 0$ эквивалентно

$$\forall Q \in R_q^* \quad (\mathrm{Pr}_s(Q) = S) \Rightarrow (P'(Q) = 0).$$

В силу используемого моделью $\mathcal{M}(l', 1)$ вида вероятности $P'(Q)$ (2.11), выполняется тождество

$$P'(Q) = 0 \Leftrightarrow \left[\begin{array}{l} \hat{\pi}'(Q_{1\ldots l'}) = 0, \\ \exists i \in \overline{l'+1, |Q|} \quad \hat{p}'(Q_i \mid Q_{i-l'\ldots i-1}) = 0, \end{array} \right.$$

где $\hat{\pi}'$ и $\hat{p}'$ — оценки начальных и переходных вероятностей, полученные с помощью формул (2.17)–(2.18).

С учетом используемых в алгоритме оценок начальных вероятностей, в случае выполнения равенства $\hat{\pi}'(Q_{1\ldots l'}) = 0$ в выборке T не существует строк, начинающихся так же, как и Q:

$$\forall X \in T \quad X_{1\ldots l'} \neq Q_{1\ldots l'}.$$

Алгоритм 2.4. Алгоритм голосования по старшинству

Вход: последовательность $S \in R_s^*$; модели $\mathbf{M} = \{\mathcal{M}_i\}_{i=1}^k$; выборка $T \subset R_q^*$

Выход: строка скрытых состояний $H \in R_h^*$, соответствующая S

1: **для** $i = 1, 2, \ldots, k$
2: Обучить модель $\mathcal{M}_i$ на выборке T;
3: Построить $\mathcal{A}_i$ — алгоритм, использующий модель $\mathcal{M}_i$;
4: **для** $i = 1, 2, \ldots, k$
5: $H := \mathcal{A}_i(S)$;
6: **если** $|H| > 0$, **то**
7: **выход**

Поскольку $Q_{1\ldots l'}$ является префиксом строки $Q_{1\ldots l}$,

$$\forall X \in T \quad X_{1\ldots l} \neq Q_{1\ldots l},$$

т. е. оценка начальной вероятности для модели $\mathcal{M}(l, 1)$ равна нулю, как и вероятность последовательности Q согласно этой модели $P(Q)$. Если же нулю равна какая-либо из переходных вероятностей, то, в силу (2.18), соответствующая подстрока Q длины $l' + 1$ не встречается в строках из обучающей выборки:

$$\exists i \, \forall X \in T, \; j = \overline{l' + 1, |X|} \quad X_{j-l'\ldots j} \neq Q_{i-l'\ldots i}.$$

Тогда в выборке не может находиться строка $Q_{i-l\ldots i}$, содержащая в себе $Q_{i-l'\ldots i}$, откуда следует

$$\hat{p}(Q_i \,|\, Q_{i-l\ldots i-1}) = 0 \Rightarrow P(Q) = 0.$$

Итак, из равенства $P'(Q) = 0$ следует $P(Q) = 0$, причем это равенство выполняется для всех последовательностей полных состояний Q, наблюдаемые части которых составляют строку S. Отсюда вероятность наблюдения S при использовании модели $\mathcal{M}(l, 1)$ равна нулю, т. е. алгоритм $\mathcal{A}$ откажется от классификации этой строки. ■

В соответствии с теоремой 2.6, множество используемых моделей $\mathbf{M}$ должно быть упорядочено по убыванию параметра l. Пример $\mathbf{M}$ — упорядоченное множество

$$\mathbf{M} = \{\mathcal{M}(l_{max} - i, 1) \,|\, i = 0, 1, \ldots, l_{max} - l_{min}\}, \tag{2.22}$$

где l_{min} и l_{max} — минимальный и максимальный порядки используемых базовых моделей. Соответствующая этому множеству модель в дальнейшем будем обозначать $\mathcal{M}_c(l_{min}, l_{max})$.

Нахождение цепочки скрытых состояний, соответствующих строке S, осуществляется согласно алгоритму 2.4. Вычислительная сложность композиции при использовании множества моделей (2.22) составляет

$$O(|S| \sum_{i=l_{min}}^{l_{max}} |R_h|^i) = O(|S| \cdot |R_h|^{l_{max}}).$$

¶ **2.6.3. Оценка отдельных вероятностей.** Применим описанный выше переход при недостатке информации к более общим моделям для аппроксимации отдельных неопределенных переходных вероятностей.

Пусть строки полных состояний соответствуют вероятностной модели $\mathcal{M}(l,1)$; тогда, согласно определению,

$$p(y\,|\,x) = \frac{P(xy)}{P(x)} = \frac{\sum_z P(zxy)}{\sum_z P(zx)} = \frac{\sum_z P(zx)p(y\,|\,zx)}{\sum_z P(zx)}, \tag{2.23}$$

где $y \in R_q$, $x \in R_q^{l'}$, $z \in R_q^{l-l'}$, $0 < l' < l$.

Если ввести обозначение

$$w_{zx} = \frac{P(zx)}{\sum\limits_{|u|=l-l'} P(ux)}, \quad \sum_z w_{zx} \equiv 1,$$

выражение (2.23) представимо в виде взвешенной суммы

$$p(y\,|\,x) = \sum_z w_{zx} p(y\,|\,zx), \tag{2.24}$$

т. е. переходные вероятности моделей более высокого порядка можно приблизить соответствующими оценками вероятности для более общих моделей.

Аналогично (2.24), для начальных вероятностей справедливо равенство

$$\forall l' \in [1, l-1]\ \forall x \in R_q^{l'} \quad \pi(x) = \sum_{|z|=l-l'} \pi(zx) \tag{2.25}$$

откуда следует связь между оценками $\hat\pi$ для моделей разных порядков. После объединения (2.24) и (2.25) получаем следующее утверждение:

Гипотеза 2.7. *Оценку вероятности $\hat p(y\,|\,x)$ для модели $\mathcal{M}(l,1)$ согласно обучающей выборке $T \subset R_q^*$ можно приблизить с помощью оценок $\hat p(y\,|\,x_{l-l'+1\ldots l})$ для моделей $\mathcal{M}(l',1)$, $0 < l' < l$, полученных на той же выборке. Аналогично, оценка $\hat\pi(x)$ допускает приближение оценками $\hat\pi(x_{l-l'+1\ldots l})$.*

Для вычисления оценки переходной вероятности вектор

$$(\hat p_1(y\,|\,x), \hat p_2(y\,|\,x), \ldots, \hat p_{l-1}(y\,|\,x)) \stackrel{\text{def}}{=} (\hat p(y\,|\,x_l), \hat p(y\,|\,x_{l-1\ldots l}), \ldots, \hat p(y\,|\,x_{2\ldots l}));$$

может подвергаться различным преобразованиям, из которых рассмотрим два:

- $\hat p(y\,|\,x)$ определяется как первая ненулевая вероятность при упорядочивании моделей по убыванию сложности:

$$\hat p(y\,|\,x) := \hat p_j(y\,|\,x), \quad j = \max\{i\,|\,\hat p_i(y\,|\,x) > 0\}. \tag{2.26}$$

- $\hat p(y\,|\,x)$ определяется средним арифметическим оценок:

$$\hat p(y\,|\,x) := \frac{1}{l - l_{min}} \sum_{i=l_{min}}^{l-1} \hat p_i(y\,|\,x), \tag{2.27}$$

где нижний предел суммирования $0 < l_{min} < l$ выбран с целью отсеять слишком общие модели.

Те же два метода комбинирования оценок могут использоваться для оценок начальных вероятностей $\hat{\pi}(x)$. Вероятностную модель, использующую аппроксимацию вида (2.26), будем в дальнейшем обозначать как $\mathcal{M}_f(l)$, модель, оценивающую недостающие вероятности согласно (2.27) — как $\mathcal{M}_s(l_{min}, l)$.

Сложность определения отдельной вероятности по обоим предложенным методам составляет $O(l)$; следовательно, сложность алгоритма 2.3 с оценкой вероятностей по формулам (2.26) или (2.27) равна $O(l|S| \cdot |R_h|^l)$.

§ 2.7. Выводы к разделу 2

Рассмотрено обобщение задач распознавания фрагментов генов и нахождения вторичной структуры белков. Для решения этой задачи предложены обобщения обыкновенных скрытых марковских моделей на основе цепей Маркова произвольного порядка; для выяснения возможности применения моделей такого рода к рассматриваемой предметной области предложен тест на основе соотношений определенного рода в ДНК и белках. Рассмотрены вспомогательные вопросы, касающиеся реализации моделей на практике, такие как оптимизация структур хранения данных, а также аппроксимация отсутствующих данных.

РАЗДЕЛ 3

КОМПОЗИЦИИ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ

Рассмотренные в разделе 2 модели составлены в предположении, что все наблюдаемые последовательности состояний генерируются на основании единого набора начальных и переходных вероятностей. Подобное утверждение в недостаточной степени обосновано в применении к обеим сформулированным в разделе 1.1 прикладным задачам. В самом деле, из исследований генов известно, что они обладают различной структурой, зависящей от местоположения в геноме, длины и других факторов. Аналогично, на основании внутреннего устройства белки можно сгруппировать в несколько достаточно больших кластеров, заметно отличающихся друг от друга по своим свойствам.

Таким образом, для решения поставленной задачи логично исследовать композиции алгоритмов, которые строятся в предположении, что наблюдаемые строки состояний порождаются с помощью нескольких вероятностных моделей. В соответствии с рассуждениями о возможности выделения нескольких классов генов и белков, рассмотрим композиции моделей, для которых множества порождаемых строк разделены между собой:

Гипотеза 3.1. *Наблюдаемые строки порождаются совокупностью вероятностных моделей* $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$*, устроенных таким образом, что*

$$P(\mathcal{M}_i \mid S) = [S \in G_i],$$

где множества порождаемых моделями наблюдаемых последовательностей $G_i \subset R_s^*$ *образуют покрытие пространства наблюдаемых строк:*

$$\bigcup_{i=1}^{k} G_i = R_s^*, \quad G_i \cap G_j = \varnothing \text{ при } i \neq j.$$

Из определения композиции следует, что каждая наблюдаемая строка может быть порождена строго одной моделью из набора $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$:

$$P(S \mid \mathcal{M}_i) = \frac{P(\mathcal{M}_i \mid S)P(S)}{P(\mathcal{M}_i)} > 0 \Rightarrow P(\mathcal{M}_i \mid S) > 0 \Rightarrow S \in G_i,$$

откуда

$$\forall i \neq j \quad P(S \mid \mathcal{M}_j) = \frac{[S \in G_j]P(S)}{P(\mathcal{M}_j)} = 0.$$

Таким образом, априорная вероятность наблюдения строки полных состояний $Q \in R_q^*$ составляет

$$P(Q) = \sum_{i=1}^{k} P(Q \mid \mathcal{M}_i)P(\mathcal{M}_i) = w_j P(Q \mid \mathcal{M}_j), \tag{3.1}$$

где $j = \arg_i(Pr_s(Q) \in G_i)$, $w_j \equiv P(\mathcal{M}_j)$ — априорная вероятность соответствующей модели.

Согласно принципу максимума правдоподобия, алгоритмическая композиция, отвечаю-

щая формуле (3.1), имеет вид

$$\mathcal{A}_c(S) = \begin{cases} \mathcal{A}_1(S), & \text{если } S \in G_1, \\ \mathcal{A}_2(S), & \text{если } S \in G_2, \\ \vdots & \\ \mathcal{A}_k(S), & \text{если } S \in G_k, \end{cases} \tag{3.2}$$

где составные алгоритмы $\mathcal{A}_1$, $\mathcal{A}_2$,..., $\mathcal{A}_k$ построены на основе моделей $\mathcal{M}_1$, $\mathcal{M}_2$,..., $\mathcal{M}_k$ соответственно.

Замечание. Композиция (3.2) является частным случаем смесей алгоритмов [63; 64]

$$\mathcal{A}_c(S) = C\Big(\sum_i g_i(S)\mathcal{A}_i(S)\Big),$$

где $g_i : R_s^* \to [0,1]$ — весовые функции, C — решающее правило. В случае использования формулы (3.2) полагается, что каждый из составляющих алгоритмов является эксклюзивно компетентным в предоставленной ему области: $g_i(S) = [S \in G_i]$. Весовые функции, заданные подобным образом, позволяют фактически не определять пространство для проведения промежуточных алгебраических операций сложения и умножения на действительное число, что для рассматриваемой задачи представляет определенные затруднения. Помимо этого, при таком определении алгоритмической композиции становится возможным использование тривиального решающего правила $C(S) \equiv S$.

Задача построения смеси алгоритмов вида (3.2) состоит из двух основных подзадач:

- определение областей компетентности G_1,..., G_k;
- обучение алгоритмов $\mathcal{A}_1$,..., $\mathcal{A}_k$ на основании полученного разбиения множества наблюдаемых строк и обучающей выборки T.

§ 3.1. Ограничение области компетентности вероятностных моделей

Пусть построение составных алгоритмов композиции $\mathcal{A}_c$ осуществляется на основе принципа максимума правдоподобия, т. е. путем максимизации правдоподобия для цепочки полных состояний по известной наблюдаемой строке:

$$\forall S \in G_i \quad \mathcal{A}_i(S) = \arg\max_Q P(Q)[\Pr_s(Q) = S].$$

Основной интерес при рассмотрении композиций, таким образом, представляет вид вероятности строки полных состояний Q для моделей $\mathcal{M}_i$. Рассмотрим модель $\mathcal{M}(G; l, m)$, полученную изменением описанной в разделе 2.4 модели $\mathcal{M}(l, m)$, чтобы ее вероятностное распределение на множестве порождаемых строк удовлетворяло условию

$$\Pr_s(Q) \notin G_i \Rightarrow P(Q \mid \mathcal{M}_i) = 0.$$

Для этого зададим распределение вероятностей при заданной длине цепочки

$$P_n(Q \mid \mathcal{M}(G;l,m)) = \begin{cases} P_n(Q \mid \mathcal{M}(l,m))/P_g(|Q|), & \text{если } \Pr_s(Q) \in G, \\ 0, & \text{если } \Pr_s(Q) \notin G; \end{cases}$$

где вероятность $P_n(Q \mid \mathcal{M}(l,m))$ считается по формуле (2.12), P_g — нормировочный множитель:

$$P_g(n) = \sum_{z \in G^{(n)}} P_n(z), \quad G^{(n)} \equiv \{Q \in R_q^n \mid \Pr_s(Q) \in G\}.$$

Таким образом, параметрами модели $\mathcal{M}(G;l,m)$ являются:

- распределение цепочек по длинам $\varphi(n)$;
- начальные вероятности $\pi(x)$, $x \in R_q^l$;
- переходные вероятности $p(y \mid x)$, $x \in R_q^l$, $y \in R_q^m$;
- множество $G \subset R_s^*$.

Теорема 3.1. *При определенных ограничениях на область действия G оценки начальных и переходных вероятностей* (2.17)*,* (2.18) *соответствуют принципу максимума правдоподобия для модели $\mathcal{M}(G;l,m)$:*

$$(\hat{\pi}, \hat{p}) = \arg\max_{\pi,p} P(T \mid \mathcal{M}(G;l,m)).$$

Доказательство.

Пусть задана выборка $T \subset R_q^*$, $\Pr_s(T) \subset G$. Тогда логарифм условной вероятности ее порождения моделью $\mathcal{M}(G;l,m)$ равен

$$\log P(T) = \sum_{Q \in T} \log \varphi(|Q|) + \sum_{Q \in T} \log P_n(Q \mid \pi, p) - \sum_{Q \in T} \log P_g(|Q|).$$

Функция Лагранжа для задачи $\log P(T) \to \max$ с учетом ограничений на начальные и переходные вероятности имеет вид

$$L(\pi,p,\lambda) = \log P(T) - \lambda_{st}\Big(\sum_{|x|=l} \pi(x) - 1\Big) - \sum_{|x|=l} \lambda_x \Big(\sum_{|y|=m} p(y \mid x) - 1\Big).$$

Частные производные этого выражения по начальным вероятностям π

$$\frac{\partial L}{\partial \pi(x)} = \frac{N_{st}(T,x)}{\pi(x)} - \sum_{Q \in T} \frac{1}{P_g(|Q|)} \frac{\partial P_g(|Q|)}{\partial \pi(x)} - \lambda_{st} = 0;$$

$$N_{st}(T,x) - \sum_{Q \in T} \frac{\pi(x)}{P_g(|Q|)} \frac{\partial P_g(|Q|)}{\partial \pi(x)} = \lambda_{st} \pi(x). \tag{3.3}$$

Согласно определению,

$$P_g(n) = \sum_{z \in G^{(n)}} [z_{1\ldots l} = x] \pi(x) \prod p(\cdot \mid \cdot) + \text{const}(\pi(x));$$

$$\frac{\partial P_g(n)}{\partial \pi(x)} = \frac{1}{\pi(x)} \sum_{z \in G^{(n)}} [z_{1\ldots l} = x] P_n(z) \equiv \frac{P_g^x(n)}{\pi(x)}.$$

После суммирования выражения (3.3) по всем строкам x получаем

$$|T| - \sum_{Q \in T} \frac{1}{P_g(|Q|)} \sum_{|x|=l} P_g^x(|Q|) = \lambda_{st}. \tag{3.4}$$

Поскольку

$$\sum_{|x|=l} P_g^x(n) = \sum_{|x|=l} \sum_{z \in G^{(n)}} [z_{1\ldots l} = x] P_n(z) = \sum_{z \in G^{(n)}} P_n(z) \equiv P_g(n),$$

из формулы (3.4) следует

$$\lambda_{st} = |T| - \sum_{Q \in T} \frac{P_g(|Q|)}{P_g(|Q|)} = |T| - |T| = 0,$$

откуда (3.3) эквивалентно

$$N_{st}(T, x) = \sum_{Q \in T} \frac{P_g^x(|Q|)}{P_g(|Q|)}.$$

Величина $P_g^x(n)$ равна вероятности генерации моделью $\mathcal{M}(l, m)$ цепочки, начинающейся с x и находящейся во множестве $G^{(n)}$ при условии, что ее длина равна n. Если распределение префиксов строк в $G^{(n)}$ мало отличается от общего распределения согласно модели $\mathcal{M}(l, m)$, выполняется равенство

$$\forall n \in \mathbb{N} \quad P_g^x(n) \cong \pi(x) P_g(n); \tag{3.5}$$

отсюда

$$N_{st}(T, x) = \sum_{Q \in T} \pi(x) = \pi(x)|T|,$$

то есть оптимальные значения начальных вероятностей, как и для модели $\mathcal{M}(l, m)$, вычисляются по формуле (2.17). При $G = R_s^*$, как и следует ожидать, равенство (3.5) выполняется строго.

Аналогично, для отдельных переходных вероятностей $p_{xy} \equiv p(y \mid x)$

$$\frac{\partial L}{\partial p_{xy}} = \frac{N^{l,m}(T, xy)}{p_{xy}} - \sum_{Q \in T} \frac{1}{P_g(|Q|)} \frac{\partial P_g(|Q|)}{\partial p_{xy}} - \lambda_x = 0; \tag{3.6}$$

$$\frac{\partial P_g(n)}{\partial p_{xy}} = \sum_{z \in G^{(n)}} \alpha(z, xy) p_{xy}^{\alpha(z,xy)-1} \frac{P_n(z)}{p_{xy}^{\alpha(z,xy)}} = \frac{1}{p_{xy}} \sum_{z \in G^{(n)}} \alpha(z, xy) P_n(z), \tag{3.7}$$

где введено обозначение $\alpha(z, xy) \equiv N^{l,m}(\{z\}, xy)$ — число вхождений вероятности $p(y \mid x)$ в формулу вычисления вероятности $P_n(z)$. Сумма в правой части равенства (3.7) является математическим ожиданием величины $\alpha(z, xy)$ для строк, входящих во множество $G^{(n)}$:

$$\sum_{z \in G^{(n)}} \alpha(z, xy) P_n(z) = \mathsf{E}_{G^{(n)}}[\alpha(z, xy)] \equiv E_{xy}(n).$$

После умножения обеих частей (3.6) на вероятность p_{xy} и суммирования по всем цепочкам $y \in R_q^m$, с учетом тождества

$$\forall n \in \mathbb{N} \quad \sum_{|y|=m} E_{xy}(n) = E_x(n)$$

получаем

$$\lambda_x = N^{l,m}(T,x) - \sum_{Q \in T} \frac{E_x(|Q|)}{P_g(|Q|)},$$

откуда после подстановки в (3.6) следует

$$N^{l,m}(T,xy) - \sum_{Q \in T} \frac{E_{xy}(|Q|)}{P_g(|Q|)} = p_{xy} N^{l,m}(T,x) - p_{xy} \sum_{Q \in T} \frac{E_x(|Q|)}{P_g(|Q|)}. \tag{3.8}$$

Если распределение вероятностей для величин $\alpha(z,x)$ и $\alpha(z,xy)$ в пределах множества $G^{(n)}$ близко к распределению для всех строк той же длины, выполняется равенство

$$E_{xy}(n) \cong p_{xy} E_x(n),$$

откуда уравнение (3.8) преобразуется к виду

$$N^{l,m}(T,xy) = p_{xy} N^{l,m}(T,x),$$

то есть оптимальные переходные вероятности совпадают с таковыми для модели $\mathcal{M}(l,m)$. ■

Согласно теореме 3.1, для вычисления параметров моделей $\mathcal{M}_i$ достаточно сделать следующее допущение о виде разбиения $\{G_i\}_{i=1}^k$:

Гипотеза 3.2. *Распределение начальных и переходных вероятностей во множествах G_i для произвольной длины цепочек не отличается от такового для всего пространства R_s^*.*

В дальнейшем будем полагать, что составные модели $\{\mathcal{M}_i\}_{i=1}^k$ соответствуют классу $\mathcal{M}(G_i; l, 1)$, т. е. для каждой из них выполняется равенство

$$\forall Q : \mathrm{Pr}_s(Q) \in G_i \quad P_n(Q \,|\, \mathcal{M}_i) = \pi(Q_{1\ldots l}) \prod_{i=l+1}^{|Q|} p(Q_i \,|\, Q_{i-l\ldots i-1}),$$

где

- $\pi(x)$, $x \in R_q^l$ — начальные вероятности для строки полных состояний;
- $p(y \,|\, x)$, $x \in R_q^l$, $y \in R_q$ — переходные вероятности.

Замечание. Сужение класса используемых вероятностных моделей служит для упрощения выкладок — результаты последующих изысканий с незначительными поправками можно обобщить для моделей вида $\mathcal{M}(G_i; l, m)$ при $m > 1$. Дополнительным стимулом к ограничению длины зависимой части строки полных состояний $m = 1$ является факт, что это значение, как показывает вычислительный эксперимент, соответствует наиболее эффективным алгоритмам.

При заданной обучающей выборке $T \subset R_q^*$ параметры модели класса $\mathcal{M}(G; l, 1)$ оцениваются согласно формулам (2.17) и (2.18). При использовании обозначений

- $N_{st}(T,x) \equiv |\{Q \in T \,|\, Q_{1\ldots l} = x\}|$ — число строк в выборке, начинающихся с определенной последовательности символов;
- $N(T,x) \equiv N^{l,1}(T,x) = \sum\limits_{Q \in T} \sum\limits_{i=l+1}^{|Q|} [Q_{i-l\ldots i-l+|x|-1} = x]$ — общее количество вхождений цепочки символов во все строки выборки;

оценки параметров модели записываются в более кратком виде:

$$\hat{\pi}(x) = N_{st}(T, x)/|T|; \tag{3.9}$$

$$\hat{p}(x) = N(T, xy)/N(T, x). \tag{3.10}$$

Определение 3.1. *Оценки начальных и переходных вероятностей* (3.9), (3.10) *на основе обучающей выборки* $T \subset R_q^*$ *при наличии неоднозначности в вопросе выбора выборки в дальнейшем будут обозначаться* $\hat{\pi}[T]$ *и* $\hat{p}[T]$ *соответственно. Модель вида* $\mathcal{M}(G; l, 1)$*, использующая оценки* $\hat{\pi}[T]$ *и* $\hat{p}[T]$*, будет обозначаться как* $\mathcal{M}[T]$.

Замечание. Приведенное определение допускает неоднозначность в выборе области действия G для модели $\mathcal{M}[T]$. Поскольку в исследуемом далее случае разбиение $\{G_i\}_{i=1}^k$ генерируется, исходя из априорных соображений, эта неопределенность не существенна; единственное ограничение, налагаемое на область, состоит во включении $\Pr_s(T) \subset G$.

С учетом описанных выше предположений, задача обучения алгоритмов, входящих в композицию, по заданной обучающей выборке и разбиению пространства наблюдаемых строк $\{G_i\}_{i=1}^k$ становится тривиальной. В самом деле, в соответствии с теоремой 3.1 оптимальные параметры модели $\mathcal{M}_i$, соответствующей области G_i, получаются путем обучения на тех строках из выборки, наблюдаемые части которых лежат в этой области: $\mathcal{M}_i = \mathcal{M}[T_i]$, где

$$T_i = \{Q \in T \mid \Pr_s(Q) \in G_i\}. \tag{3.11}$$

Подобно множествам $G_1, G_2, \ldots, G_k$, части обучающей выборки $T_1, T_2, \ldots, T_k$ образуют ее разбиение:

$$\bigcup_{i=1}^{k} T_i = T, \quad T_i \cap T_j = \varnothing \text{ при } i \neq j.$$

Таким образом, из двух сформулированных выше проблем остается нерешенной задача нахождения оптимального разбиения $\{G_i\}_{i=1}^k$. В соответствии с принципом максимума правдоподобия, оптимальным будем полагать разбиение, которое приводит к наиболее высокой вероятности порождения заданной обучающей выборки:

Гипотеза 3.3. *Оптимальное разбиение множества наблюдаемых строк* $\mathbf{G} = \{G_i\}$ *соответствует максимуму условной вероятности порождения обучающей выборки* T *моделями* $\mathcal{M}_i = \mathcal{M}(G_i; l, 1)$*, параметры которых получены согласно формулам* (3.9), (3.10) *для соответствующих частей выборки:*

$$P(T \mid \mathcal{M}[T_1], \mathcal{M}[T_2], \ldots, \mathcal{M}[T_k]) \to \max_{T_1, \ldots, T_k},$$

разбиение $\{T_i\}_{i=1}^k$ *определено равенством* (3.11).

В силу выбранного вида алгоритмических композиций, модели $\mathcal{M}[T_i]$, порожденные различными базовыми алгоритмами, независимы друг от друга. Таким образом, справедливо равенство

$$\log P(T \mid \mathcal{M}[T_1], \mathcal{M}[T_2], \ldots, \mathcal{M}[T_k]) = \sum_{i=1}^{k} \log P(T_i \mid \mathcal{M}[T_i]). \tag{3.12}$$

Непосредственная оптимизация выражения (3.12) представляет сложность при большом количестве фрагментов выборки k. В связи с этим в дальнейшем будем использовать поэтапную максимизацию этой вероятности. Оптимальное разбиение будем строить в виде бинарного дерева: вначале будет искаться оптимальное разбиение множества наблюдаемых строк R_s^* на два подмножества $G_1 \cup G_2 = R_s^*$; затем одного из полученных множеств на две части, и так далее до тех пор, пока количество множеств в разбиении не достигнет k.

Для выбора оптимального бинарного разбиения следует, как описано выше, отображать оптимальное разбиение на заданную обучающую выборку. В качестве меры разделительной способности для разбиения множества строк T на две части, T^+ и T^-, будем использовать выражение

$$\Delta(T, T^+, T^-) = \log P(T^+ \mid \mathcal{M}[T^+]) + \log P(T^- \mid \mathcal{M}[T^-]) - \log P(T \mid \mathcal{M}[T]), \tag{3.13}$$

которое характеризует прирост функции правдоподобия для выборки при использовании двух моделей вместо одной:

$$\Delta(T, T^+, T^-) = \frac{P(T \mid \mathcal{M}[T^+], \mathcal{M}[T^-])}{P(T \mid \mathcal{M}[T])}.$$

После сложения выражений (3.13) для частей выборки, подвергшихся делению на каждом из шагов описанного выше алгоритма, получим (3.12) с точностью до не зависящего от финального разбиения слагаемого

$$-\log P(T \mid \mathcal{M}[T]) = \text{const}(\{G_i\}_{i=1}^k).$$

Следовательно, алгоритм построения **G** является жадным способом максимизации вероятности (3.12).

§ 3.2. Вычисление функционала качества разбиения

Согласно данному выше определению модели $\mathcal{M}[T]$,

$$\log P(T \mid \mathcal{M}[T]) = \sum_{Q \in T} \log \hat{\pi}(Q_{1\ldots l}) + \sum_{Q \in T} \sum_{j=l+1}^{|Q|} \log \hat{p}(Q_j \mid Q_{j-l\ldots j-1}). \tag{3.14}$$

За счет равенства (3.9),

$$\sum_{Q \in T} \log \hat{\pi}(Q_{1\ldots l}) = \sum_{Q \in T} \big(\log N_{st}(T, Q_{1\ldots l}) - \log |T|\big) = \sum_{Q \in T} \log N_{st}(T, Q_{1\ldots l}) - |T| \log |T|.$$

Если в последнем выражении совершить переход от суммирования по строкам из выборки T к суммированию по всевозможным префиксам этих строк длины l, оно приобретет вид

$$\sum_{Q \in T} \log \hat{\pi}(Q_{1\ldots l}) = \sum_{|x|=l} N_{st}(T, x) \log N_{st}(T, x) - |T| \log |T|. \tag{3.15}$$

(В уравнении (3.15), а также в последующем изложении полагается, что при $x = 0$ справедливо равенство $x \log x = 0$.)

Аналогично, в силу (3.10)

$$\sum_{Q\in T}\sum_{j=l+1}^{|Q|}\log\hat{p}(Q_j\,|\,Q_{j-l\ldots j-1}) = \sum_{Q\in T}\sum_{j=l+1}^{|Q|}\big(\log N(T,Q_{j-l\ldots j}) - \log N(T,Q_{j-l\ldots j-1})\big),$$

или, после перехода к суммированию по всевозможным цепочкам полных состояний длины l и $l+1$,

$$\sum_{Q\in T}\sum_{j=l+1}^{|Q|}\log\hat{p}(Q_j\,|\,Q_{j-l\ldots j-1}) = \sum_{|z|=l+1}N(T,z)\log N(T,z) - \sum_{|x|=l}N(T,x)\log N(T,x). \tag{3.16}$$

После подстановки выражений (3.15) и (3.16) в (3.14) условная вероятность генерации части обучающей выборки представляется в виде

$$\log P(T\,|\,\mathcal{M}[T]) = \sum_{|x|=l}N_{st}(T,x)\log N_{st}(T,x) - |T|\log|T| + \\ + \sum_{|z|=l+1}N(T,z)\log N(T,z) - \sum_{|x|=l}N(T,x)\log N(T,x). \tag{3.17}$$

Теорема 3.2. *При определении вероятности порождения выборки согласно уравнению* (3.17) *мера разделительной способности* (3.13) *является неотрицательной:*

$$\forall T^+\subseteq T\subset R_q^*, T^- = T\setminus T^+ \quad \Delta(T,T^+,T^-)\geqslant 0.$$

Доказательство.

Рассмотрим члены меры, соответствующие эмпирическим начальным распределениям вероятности $\hat{\pi}$ для всей выборки T и ее частей T^+ и T^-:

$$\Delta_\pi(T,T^+,T^-) = \sum_{|x|=l}N_{st}(T^+,x)\log N_{st}(T^+,x) + \\ + \sum_{|x|=l}N_{st}(T^-,x)\log N_{st}(T^-,x) - \sum_{|x|=l}N_{st}(T,x)\log N_{st}(T,x) + \\ + |T|\log|T| - |T^+|\log|T^+| - |T^-|\log|T^-|. \tag{3.18}$$

Обозначим для краткости $N^+(x)\equiv N_{st}(T^+,x)$, $N^-(x)\equiv N_{st}(T^-,x)$, тогда выполняется тождество $N_{st}(T,x) = N^+(x)+N^-(x)$. Суммируемые выражения преобразуются следующим образом:

$$N^+(x)\log N^+(x) + N^-(x)\log N^-(x) - N_{st}(T,x)\log N_{st}(T,x) = \\ = N^+(x)\log\frac{N^+(x)}{N^+(x)+N^-(x)} + N^-(x)\log\frac{N^-(x)}{N^+(x)+N^-(x)} = \\ = (N^+(x)+N^-(x))\psi(\gamma_x), \tag{3.19}$$

где приняты обозначения

$$\gamma_x \equiv \frac{N^+(x)}{N^+(x) + N^-(x)};$$

$$\psi(r) = \begin{cases} r\log r + (1-r)\log(1-r), & \text{если } r \in (0,1), \\ 0, & \text{если } r = 0 \text{ или } r = 1. \end{cases}$$

С учетом (3.19) равенство (3.18) записывается в виде

$$\Delta_\pi(T, T^+, T^-) = \sum_{|x|=l} (N^+(x) + N^-(x))\psi(\gamma_x) - |T|\psi(|T^+|/|T|),$$

или, если ввести обозначение $\lambda_x \equiv (N^+(x) + N^-(x))/|T|$,

$$\Delta_\pi(T, T^+, T^-) = |T|\Big(\sum_{|x|=l} \lambda_x \psi(\gamma_x) - \psi(|T^+|/|T|)\Big).$$

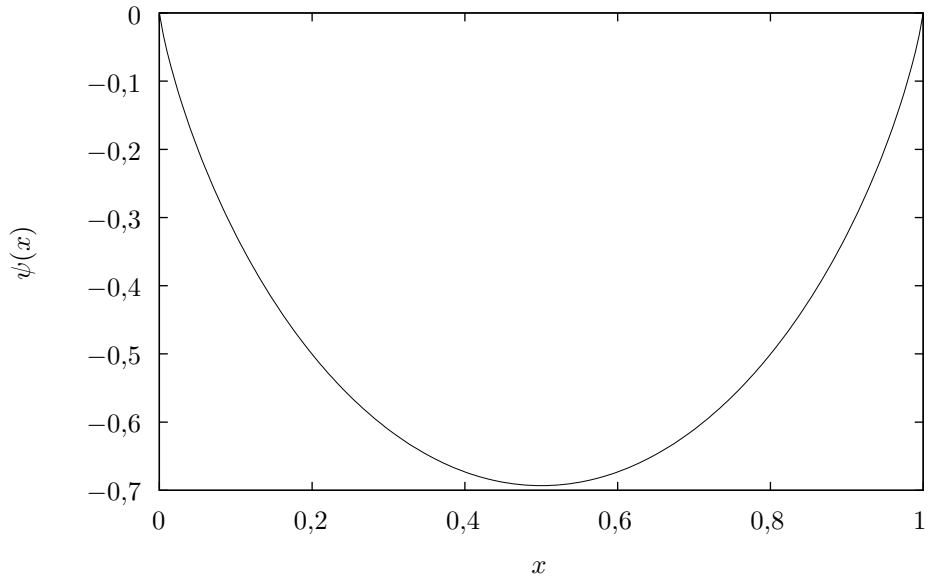


Рис. 3.1. График функции $\psi(x) = x\log x + (1-x)\log(1-x)$

Функция $\psi(r)$ является выпуклой на интервале $(0,1)$ (рис. 3.1), поскольку

$$\begin{aligned} \psi'(r) &= 1 + \log r - 1 - \log(1-r) = \log r - \log(1-r); \\ \psi''(r) &= 1/r + 1/(1-r) > 0 \quad \forall r \in (0,1). \end{aligned}$$

В силу определения переменных γ и λ справедливо тождество

$$\forall x \in R_q^l \quad \lambda_x\gamma_x = \frac{N^+(x) + N^-(x)}{|T|}\frac{N^+(x)}{N^+(x) + N^-(x)} = \frac{N^+(x)}{|T|};$$

следовательно,

$$\frac{|T^+|}{|T|} = \sum_{|x|=l} \lambda_x\gamma_x.$$

Также выполняется условие

$$\sum_{|x|=l} \lambda_x = \frac{1}{|T|} \sum_{|x|=l} \big(N^+(x) + N^-(x)\big) = \frac{|T^+| + |T^-|}{|T|} = 1.$$

Таким образом, в силу неравенства Йенсена для функции ψ,

$$\Delta_\pi(T, T^+, T^-) = |T| \Big(\sum_{|x|=l} \lambda_x \psi(\gamma_x) - \psi\big(\sum_{|x|=l} \lambda_x \gamma_x \big) \Big) \geqslant 0. \tag{3.20}$$

Аналогично доказывается неравенство для части меры, соответствующей переходным вероятностям. Для этого достаточно в выражении (3.17) перегруппировать члены в соответствии с первыми l их символами:

$$\log P(T \mid \mathcal{M}[T]) = \cdots + \sum_{|x|=l} \Big(\sum_{|y|=1} N(T, xy) \log N(T, xy) - N(T, x) \log N(T, x) \Big).$$

Поскольку для каждой из $|R_q|^l$ полученных групп выполняются равенства

$$\sum_{|y|=1} N(T, xy) = N(T, x),$$

они аналогичны выражению (3.15) для начальных вероятностей с учетом тождества

$$\sum_{|x|=l} N_{st}(T, x) = |T|.$$

Следовательно, для каждой из этих групп после подстановки в выражение для меры (3.13) можно применить рассуждения, подобные приведенным выше. Таким образом, часть меры, соответствующая переходным вероятностям, также неотрицательна:

$$\Delta_p(T, T^+, T^-) = \sum_{|x|=l} N(T, x) \big(\dots \big) \geqslant 0;$$

$$\Delta(T, T^+, T^-) = \Delta_\pi(T, T^+, T^-) + \Delta_p(T, T^+, T^-) \geqslant 0,$$

что и требовалось доказать. ■

Следствие 3.2.1. *Мера разделительной способности $\Delta(T, T^+, T^-)$ равна нулю тогда и только тогда, когда модели $\mathcal{M}[T^+]$ и $\mathcal{M}[T^-]$ совпадают между собой и с моделью $\mathcal{M}[T]$.*

Доказательство.

Поскольку функция $\psi(r)$, используемая в доказательстве, является строго выпуклой, неравенство Йенсена в формуле (3.20) превращается в равенство только при условии, что для всех строк $x \in R_q^l$ значения γ_x равны между собой. Пусть $\forall x \in R_q^l \ \gamma_x = \gamma^+$; тогда после суммирования тождества

$$N^+(x) = \gamma^+ (N^+(x) + N^-(x))$$

по всем значениям переменной x получится $|T^+| = \gamma^+ |T|$. Тогда для любой строки $x \in R_q^l$ выполняются равенства

$$\hat{\pi}[T^+](x) = \frac{N^+(x)}{|T^+|} = \frac{\gamma^+ (N^+(x) + N^-(x))}{\gamma^+ |T|} = \hat{\pi}[T](x);$$

$$\hat{\pi}[T^-](x) = \frac{N^-(x)}{|T^-|} = \frac{(1-\gamma^+)(N^+(x)+N^-(x))}{(1-\gamma^+)|T|} = \hat{\pi}[T](x);$$

Аналогично,

$$\forall x \in R_q^l, y \in R_q \quad \hat{p}[T](y \mid x) = \hat{p}[T^+](y \mid x) = \hat{p}[T^-](y \mid x).$$

Следовательно, все параметры моделей $\mathcal{M}[T]$, $\mathcal{M}[T^+]$ и $\mathcal{M}[T^-]$ совпадают, что и требовалось доказать. ■

На практике равенство $\Delta(T, T^+, T^-) = 0$ наблюдается исключительно для тривиального разбиения, для которого одно из множеств T^+ и T^- является пустым. Чем больше отличаются друг от друга модели $\mathcal{M}[T^+]$ и $\mathcal{M}[T^-]$, тем выше значение $\Delta(T, T^+, T^-)$. Таким образом, предложенный метод построения разбиения в действительности ищет группы строк, максимально различающихся по структуре соответствующих им марковских цепей.

Вычисление величины $\Delta(T, T^+, T^-)$, как видно из уравнений (3.13) и (3.17), включает в себя два этапа:

1) Сбор статистик

$$N_{st}(T, x), N_{st}(T^+, x), N(T, x), N(T^+, x), N(T, z), N(T^+, z)$$

для всех значений $x \in R_q^l$, $z \in R_q^{l+1}$. Статистики для второй части выборки T^- получаются из тождеств типа $N(T^-, x) = N(T, x) - N(T^+, x)$.

2) Непосредственное вычисление выражения (3.13) с учетом вида логарифмов условных вероятностей (3.17).

После объединения (3.13) и (3.17) выражение для меры качества разбиения приобретает вид

$$\Delta(T, T^+, T^-) = \sum_{|x|=l} \Psi(N_{st}(T, x), N_{st}(T^+, x)) - \Psi(|T|, |T^+|) + \\ + \sum_{|z|=l+1} \Psi(N(T, z), N(T^+, z)) - \sum_{|x|=l} \Psi(N(T, x), N(T^+, x)), \quad (3.21)$$

где функция $\Psi : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{R}$ определена как

$$\Psi(a, b) = b \log b + (a - b) \log(a - b) - a \log a.$$

Таким образом, нахождение меры качества соответствует листингу 3.1.

Подходящей структурой для хранения переменных вида $N_{st}(\cdot, \cdot)$ и $N(\cdot, \cdot)$ при имплементации алгоритма 3.1 являются хэш-таблицы. При их использовании вычислительная сложность инициализации переменных (шаги 1–5) составляет $O(1)$. Сложность подсчета статистик (шаги 6–12) определяется тремя факторами:

- числом итераций шагов 11 и 12 — $\Theta(L(T))$, где $L(T)$ — суммарная длина строк выборки T;
- сложностью вычисления хэш-функции для последовательностей состояний длиной l и $l+1$ — $\Theta(l)$;

Алгоритм 3.1. Нахождение качества разбиения

Вход: выборка $T \subset R_q^*$; ее разбиение $\{T^+, T^-\}$; порядок марковских цепочек l

Выход: мера качества разбиения $\Delta \equiv \Delta(T, T^+, T^-)$

1: **для всех** $x \in R_q^l$
2: $N_{st}(T, x) := 0$; $N_{st}(T^+, x) := 0$;
3: $N(T, x) := 0$; $N(T^+, x) := 0$;
4: **для всех** $z \in R_q^{l+1}$
5: $N(T, z) := 0$; $N(T^+, z) := 0$;
6: **для всех** $Q \in T$
7: $N_{st}(T, Q_{1\ldots l}) := N_{st}(T, Q_{1\ldots l}) + 1$;
8: $N_{st}(T^+, Q_{1\ldots l}) := N_{st}(T^+, Q_{1\ldots l}) + [Q \in T^+]$;
9: **для** $t \in \{l, l+1\}$ — длина исследуемых цепочек
10: **для** $j = 1, 2, \ldots, |Q| - t + 1$
11: $N(T, Q_{j\ldots j+t-1}) := N(T, Q_{j\ldots j+t-1}) + 1$;
12: $N(T^+, Q_{j\ldots j+t-1}) := N(T^+, Q_{j\ldots j+t-1}) + [Q \in T^+]$;
13: $\Delta := -\Psi(|T|, |T^+|)$;
14: **для всех** $x \in R_q^l$
15: $\Delta := \Delta + \Psi(N_{st}(T, x), N_{st}(T^+, x)) - \Psi(N(T, x), N(T^+, x))$;
16: **для всех** $z \in R_q^{l+1}$
17: $\Delta := \Delta + \Psi(N(T, z), N(T^+, z))$.

- временем доступа к таблицам по известной хэш-функции. При адекватной реализации хэш-функции время доступа составляет в среднем $O(1)$.

Следовательно, алгоритмическая сложность подсчета статистик составляет в среднем $O(l \cdot L(T))$.

Сложность непосредственного подсчета меры качества (шаги 14–17 алгоритма 3.1) равна $O(|R_q|^l)$; поскольку итерация в циклах на шагах 14 и 16 производится по всем элементам соответствующих хэш-таблиц, время доступа к ним гарантированно составляет $O(1)$. Так как количество ненулевых элементов таблиц, т. е. различных встречающихся в строках выборки T подстрок длины l и $l+1$ не может превышать общую длину всех строк выборки, оценку сложности можно уточнить до $O(L(T))$.

Итак, общая сложность алгоритма определяется сложностью подсчета статистик и составляет в среднем $O(l \cdot L(T))$. Объем памяти, потребляемой алгоритмом, определяется размером хэш-таблиц, и, с учетом приведенного выше замечания, составляет $O(\min(|R_q|^l, L(T)))$.

§ 3.3. Связь функционала качества разбиения с информационной энтропией

Выражение для вероятности порождения части выборки (3.17) содержит члены вида $x \log x$, напоминающие члены в формулах для вычисления информационной энтропии эмпирических вероятностных распределений $\hat{\pi}$ и $\hat{p}$. В самом деле, энтропия для распределения на-

чальной вероятности составляет

$$\mathsf{H}(\hat{\pi}) = \sum_{|x|=l} \hat{\pi}(x) \log \hat{\pi}(x) = -\frac{1}{|T|} \sum_{|x|=l} N_{st}(T,x) \big(\log N_{st}(T,x) - \log |T|\big) =$$

$$= \log |T| - \frac{1}{|T|} \sum_{|x|=l} N_{st}(T,x) \log N_{st}(T,x). \quad (3.22)$$

Энтропия для условного распределения $\hat{p}$, согласно определению, находится по формуле

$$\mathsf{H}(\hat{p}) = \mathsf{H}(\hat{z} \,|\, \hat{x}) = \mathsf{H}(\hat{z}, \hat{x}) - \mathsf{H}(\hat{x}), \quad (3.23)$$

где $\hat{z}$ — эмпирическое распределение для последовательностей из $l+1$ полных состояний, $\hat{x}$ — аналогичное распределение для последовательностей длины l. Значение энтропии для случайной величины $\hat{x}$ равно

$$\mathsf{H}(\hat{x}) = \sum_{|x|=l} \frac{N(T,x)}{L_l(T)} \log \frac{N(T,x)}{L_l(T)} = \log L_l(T) - \frac{1}{L_l(T)} \sum_{|x|=l} N(T,x) \log N(T,x), \quad (3.24)$$

где через $L_l(T)$ обозначено общее количество последовательностей длины l в строках, входящих в T:

$$L_l(T) = \sum_{Q \in T} \big(|Q| - l + 1\big) = \sum_{Q \in T} |Q| - (l-1)|T|.$$

После подстановки в формулу (3.23) равенства (3.24) и аналогичного ей выражения для энтропии $\mathsf{H}(\hat{z})$ с учетом приближенного равенства

$$L_l(T) \approx L_{l+1}(T) \approx L(T) = \sum_{Q \in T} |Q|$$

получаем

$$\mathsf{H}(\hat{p}) \approx -\frac{1}{L(T)} \Big(\sum_{|z|=l+1} N(T,z) \log N(T,z) - \sum_{|x|=l} N(T,x) \log N(T,x) \Big). \quad (3.25)$$

В результате сравнения (3.22) и (3.25) с (3.17) выходит следующая оценка для условной вероятности порождения выборки:

$$\log P(T \,|\, \mathcal{M}[T]) \approx -\mathsf{H}(\hat{\pi})|T| - \mathsf{H}(\hat{p})L(T).$$

Таким образом, нахождение максимума выражения (3.12) соответствует минимизации энтропии эмпирических распределений для начальных и переходных вероятностей, получаемых с помощью отдельных частей обучающей выборки, по сравнению с энтропией для всей выборки в целом. Это наблюдение служит дополнительным теоретическим обоснованием выбора меры разделительной способности (3.13).

§ 3.4. Использование предикатов для построения бинарных разбиений

При формулировке принципа построения оптимального разбиения в виде бинарного дерева не конкретизировалось, каким образом происходит деление одной из областей на каждом

шаге алгоритма. С этой целью могут использоваться произвольные предикаты вида $I : R_s^* \to \{0,1\}$. Мера качества разделения множества $T \subset R_q^*$, порождаемого таким предикатом, составляет

$$\Delta(T, I) \stackrel{\text{def}}{=} \Delta(T, \{Q \in T \,|\, I(\text{Pr}_s(Q))\}, \{Q \in T \,|\, \neg I(\text{Pr}_s(Q))\}).$$

Согласно теореме 3.2 и ее следствию, для любого нетривиального предиката следует ожидать выполнения строгого неравенства $\Delta(T, I) > 0$.

Одним из видов предикатов, которые могут использоваться при построении разбиения, являются предикаты, основанные на концентрации отдельных наблюдаемых символов либо их групп. В общем случае вместо отдельных символов в предикатах могут использоваться короткие цепочки длиной u. Если ввести обозначение

$$n(S, x) \stackrel{\text{def}}{=} \frac{1}{|S|} \sum_{i=1}^{|S|-u+1} [S_{i\ldots i+u-1} = x], \quad S \in R_s^*, x \in R_s^u;$$

$$n(S, X) \stackrel{\text{def}}{=} \sum_{x \in X} n(S, x), \quad X \subset R_s^u;$$

вышеописанный тип предикатов имеет вид $I_{X,\omega}(S) \stackrel{\text{def}}{=} [n(S, X) > \omega]$, где ω — заданная пороговая концентрация, $\omega \in (0, 1)$. Множество всех таких предикатов для конкретного значения длины u будем обозначать $\mathfrak{I}_u$. Вследствие простого вида предикатов при малых значениях длины ($u \ll l$) справедлива гипотеза 3.2.

Вычислительная сложность определения $I_{X,\omega}(S)$ составляет $\Theta(u|X||S|)$, то есть линейным образом зависит от длины входной строки, длины цепочек, для которых ищутся концентрации, а также от количества этих цепочек.

Помимо простоты вычисления, достоинством описанных предикатов является простота их интерпретации. Так, для задачи распознавания фрагментов генов известно, что концентрации нуклеотидов отличаются для генов и не кодирующих гены участков ДНК, а также для интронов и экзонов; логично предположить, что неравномерность распределения по нуклеотидам или коротким цепочкам нуклеотидов соблюдается и для различных типов генов. Аналогично, разница в концентрациях определенных групп аминокислот вполне ожидаема для различных типов белков.

Задача максимизации выражения (3.13), которую необходимо решать на каждом шаге описанного в разделе 3.1 алгоритма, в рамках использования предикатов вида $I_{X,\omega}$ сводится к поиску оптимальных значений параметров X и ω для частей обучающей выборки, входящих в текущее разбиение. Поскольку мощность множества предикатов $\mathfrak{I}_u$ совпадает с мощностью континуума, необходимо сократить множество рассматриваемых предикатов с учетом рассматриваемой части обучающей выборки:

Задача 3.1. *По заданному множеству строк полных состояний $T \subset R_q^*$ определить конечное множество предикатов* $\text{Pred}(T) \subset \mathfrak{I}_u$, $|\text{Pred}(T)| < \infty$, *которые являются близкими к оптимальному предикату:*

$$\exists I^* \in \text{Pred}(T) \quad \Delta(T, I^*) \approx \max_{I \in \mathfrak{I}_u} \Delta(T, I).$$

Для произвольного значения длины цепочек u, которые используются при подсчете кон-

центрации, количество различных множеств цепочек X составляет $2^{a(u)}$, где $a(u) = |R_s|^u$ — общее число строк выбранной длины. Случаи $X = \varnothing$ и $X = R_s^u$ приводят к тривиальным предикатам; оставшиеся $2^{a(u)} - 2$ множеств X допускают разбиение на пары в соответствии с тождеством

$$\forall X \subset R_s^u, \omega \in (0,1) \quad I_{X,\omega} \sim I_{Y,1-\omega}, \text{ где } Y = R_s^u \setminus X,$$

выполняющимся вследствие

$$\forall X \subset R_s^u, S \in R_s^* \quad n(S, R_s^u \setminus X) = 1 - n(S, X).$$

Таким образом, перебор первого параметра предикатов $I_{X,\omega}$ можно ограничить $2^{a(u)-1} - 1$ элементами множества $\mathbf{X} \subset 2^{R_s^u}$:

$$\varnothing \notin \mathbf{X},\ R_s^u \notin \mathbf{X},\ \forall X \in \mathbf{X} \quad R_s^u \setminus X \notin \mathbf{X}.$$

Поскольку сложность вычисления предикатов $I_{X,\omega}$ возрастает с увеличением количества элементов множества X, поместим в $\mathbf{X}$ множества минимально возможного размера: все подмножества R_s^u из 1, 2,..., $\lfloor (|R_s|^u - 1)/2 \rfloor$ элементов, а также, если $|R_s|$ четно, половину подмножеств R_s^u размером $|R_s|^u/2$.

Пример. Для задачи определения экзонов и интронов в генах при использовании концентраций отдельных нуклеотидов ($u = 1$) достаточно рассматривать $2^{4-1} - 1 = 7$ множеств оснований:

$$X \in \mathbf{X} = \big\{\{A\}, \{C\}, \{G\}, \{T\}, \{A, C\}, \{A, G\}, \{A, T\}\big\}.$$

Остальные 7 нетривиальных наборов нуклеотидов сводятся к вышеперечисленным; так, поскольку $n(S, \{C, G\}) = 1 - n(S, \{A, T\})$, предикаты вида $I_{\{C,G\},\omega}$ эквивалентны $I_{\{A,T\},1-\omega}$.

Несмотря на то, что множество возможных пороговых концентраций ω континуально, из-за конечности множества строк T для каждого набора наблюдаемых цепочек $X \in \mathbf{X}$ существует не более $|T|$ значений ω, которые приводят к различным мерам $\Delta(T, I_{X,\omega})$. В самом деле, достаточно рассмотреть значения концентрации из множества

$$\Omega(T, X) \stackrel{\text{def}}{=} \{n(\mathrm{Pr}_s(Q), X) \,|\, Q \in T\}. \tag{3.26}$$

Каждый предикат, порождаемый этими пороговыми значениями, выполняется для различного числа проекций строк из множества T: от нуля для значения

$$\omega = \omega_{max}(T, X) \stackrel{\text{def}}{=} \max_{Q \in T} n(\mathrm{Pr}_s(Q), X),$$

до $|T| - 1$ при

$$\omega = \omega_{min}(T, X) \stackrel{\text{def}}{=} \min_{Q \in T} n(\mathrm{Pr}_s(Q), X),$$

Итак, общее количество предикатов из $\mathfrak{I}_u$, среди которых гарантированно находится максимум $\Delta(T, I)$, составляет $\Theta(2^{a(u)}|T|)$. На практике подсчет меры качества такого числа сечений множества T не представляется возможным; для достижения приемлемого размера множества $\mathrm{Pred}(T)$ будем сокращать количество элементов множеств Ω и, в определенных случаях, $\mathbf{X}$.

§ 3.5. Построение множества пороговых концентраций

Для того чтобы с теоретической точки зрения обосновать возможность сокращения размера множества порогов Ω для каждого набора $X \subset R_s^u$, покажем, что при малом изменении пороговой концентрации ω качество разбиения, порождаемого предикатом $I_{X,\omega}$, также меняется мало:

Теорема 3.3. *Пусть $T \subset R_q^*$ — обучающая выборка; I и $\tilde{I}$ — предикаты, различающиеся исключительно значением пороговой концентрации:*

$$I(S) = [n(S,X) < \omega],\ \tilde{I}(S) = [n(S,X) < \tilde{\omega}], \quad \omega < \tilde{\omega}$$

множество строк $\tilde{T} \subset T$ определено как

$$\tilde{T} \stackrel{\text{def}}{=} \{Q \in T \,|\, \neg I(\Pr_s(Q)) \wedge \tilde{I}(\Pr_s(Q))\}.$$

Тогда справедлива следующая оценка:

$$|\Delta(T,\tilde{I}) - \Delta(T,I)| \leqslant CL(\tilde{T}) \equiv C \sum_{Q \in \tilde{T}} |Q|,$$

где $C > 0$ — определенное число, не зависящее от предикатов I и $\tilde{I}$.

Доказательство.

Введем обозначения

$$N^0(x) \stackrel{\text{def}}{=} N_{st}(T,x),\ N^+(x) \stackrel{\text{def}}{=} N_{st}(T,x),\ \tilde{N}^+(x) \stackrel{\text{def}}{=} N_{st}(\tilde{T}^+,x),$$

где множества строк T^+ и $\tilde{T}^+$ определены как

$$T^+ = \{Q \in T \,|\, I(\Pr_s(Q))\}, \quad \tilde{T}^+ = \{Q \in T \,|\, \tilde{I}(\Pr_s(Q))\}.$$

В силу определения предикатов I и $\tilde{I}$ выполняются тождества

$$\forall S \in R_s^* \quad I(S) \Rightarrow \tilde{I}(S), \neg\tilde{I}(S) \Rightarrow \neg I(S).$$

Таким образом, множество $\tilde{T}^+$ включает в себя все строки из T^+; более того, $\tilde{T}^+ \setminus T^+ = \tilde{T}$.

Разность функционалов качества разбиений, порожденных предикатами I и $\tilde{I}$, состоит из четырех частей, соответствующих четырем слагаемым из формулы (3.21):

$$\Delta(T,\tilde{I}) - \Delta(T,I) = \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4;$$

$$|\Delta(T,\tilde{I}) - \Delta(T,I)| \leqslant |\Delta_1| + |\Delta_2| + |\Delta_3| + |\Delta_4|.$$

Оценим первую из частей

$$\Delta_1 \stackrel{\text{def}}{=} \sum_{|x|=l} \Psi(N^0(x), \tilde{N}^+(x)) - \Psi(N^0(x), N^+(x)).$$

Функции Ψ и ψ из доказательства теоремы 3.2 связаны соотношением

$$\Psi(a,b) = a\left(\frac{b}{a}\log b + \frac{a-b}{a}\log(a-b) - \log(a)\right) = a\psi(b/a).$$

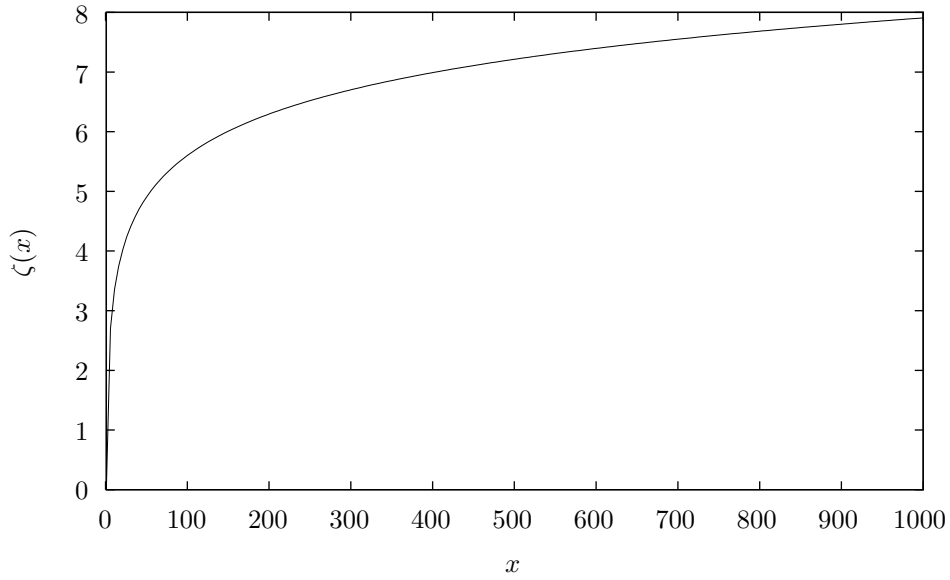


Рис. 3.2. График функции $\zeta(x) = -x\psi(1/x)$

Таким образом,

$$\Delta_1 = \sum_{|x|=l} N(x) \left(\psi\left(\frac{\tilde{N}^+(x)}{N^0(x)} \right) - \psi\left(\frac{N^+(x)}{N^0(x)} \right) \right).$$

Несмотря на то, что производная функции $\psi(r)$ обращается в бесконечность в точках $r = 0$ и $r = 1$, вследствие дискретности рассматриваемых аргументов функции для нее справедливо условие Липшица:

$$|\psi(b/a) - \psi(\tilde{b}/a)| \leqslant |a\psi(1/a)| \cdot |\tilde{b}/a - b/a| = \zeta(a)|\tilde{b}/a - b/a|, \tag{3.27}$$

где $0 \leqslant b \leqslant \tilde{b} \leqslant a$, $\zeta(a) \stackrel{\text{def}}{=} -a\psi(1/a)$. Следовательно,

$$|\Delta_1| \leqslant \sum_{|x|=l} N^0(x)\zeta(N^0(x)) \frac{|\tilde{N}^+(x) - N^+(x)|}{N^0(x)} = \sum_{|x|=l} \zeta(N^0(x))|\tilde{N}^+(x) - N^+(x)|. \tag{3.28}$$

Функция $\zeta(a)$ является возрастающей на множестве натуральных чисел (рис. 3.2), поскольку при $a > 1$ справедливо неравенство

$$\zeta'(a) = (a\log a - (a-1)\log(a-1))' = \log a - \log(a-1) > 0;$$

таким образом,

$$\forall x \in R_q^l \quad \zeta(N^0(x)) \leqslant \zeta\Big(\max_{|x|=l} N^0(x)\Big) \stackrel{\text{def}}{=} \zeta_{st}.$$

Поскольку разность между величинами $\tilde{N}^+(x)$ и $N^+(x)$ равна количеству строк из множества $\tilde{T}$, начинающихся с последовательности x:

$$\tilde{N}^+(x) - N^+(x) = N_{st}(\tilde{T}, x),$$

из неравенства (3.28) получаем оценку

$$|\Delta_1| \leqslant \zeta_{st} \sum_{|x|=l} |\tilde{N}^+(x) - N^+(x)| = \zeta_{st} \sum_{|x|=l} N_{st}(\tilde{T}, x) = \zeta_{st}|\tilde{T}|.$$

Аналогично оцениваются сверху другие части разности мер качества:

$$|\Delta_2| = \left|\Psi(|T|, |\tilde{T}^+|) - \Psi(|T|, |T^+|)\right| \leqslant \zeta(|T|)|\tilde{T}|;$$

$$|\Delta_3| = \Big| \sum_{|z|=l+1} \Psi(N(T,z), N(\tilde{T}^+, z)) - \Psi(N(T,z), N(T^+,z))\Big| \leqslant$$

$$\leqslant \zeta_{l+1} \sum_{|z|=l+1} |N(\tilde{T}^+, z) - N(T^+, z)| = \zeta_{l+1} \sum_{|z|=l+1} N(\tilde{T}, z) < \zeta_{l+1} L(\tilde{T});$$

$$|\Delta_4| = \Big| \sum_{|x|=l} \Psi(N(T,x), N(\tilde{T}^+, x)) - \Psi(N(T,x), N(T^+,x))\Big| \leqslant$$

$$\leqslant \zeta_l \sum_{|x|=l} |N(\tilde{T}^+, x) - N(T^+, x)| = \zeta_l \sum_{|z|=l} N(\tilde{T}, x) < \zeta_l L(\tilde{T}),$$

где

$$\zeta_l = \zeta\Big(\max_{|x|=l} N(T,x)\Big), \quad \zeta_{l+1} = \zeta\Big(\max_{|z|=l+1} N(T,z)\Big).$$

Окончательно,

$$|\Delta(T,\tilde{I}) - \Delta(T,I)| \leqslant (\zeta_{st} + \zeta(|T|))|\tilde{T}| + (\zeta_l + \zeta_{l+1})L(\tilde{T}),$$

или, с учетом неравенства $|\tilde{T}| < L(T)$,

$$|\Delta(T,\tilde{I}) - \Delta(T,I)| \leqslant CL(\tilde{T})$$

при $C = \zeta_{st} + \zeta(|T|) + \zeta_l + \zeta_{l+1}$. ∎

Замечание. Полученное значение для величины C зависит от обучающей выборки T. Это значение ограничено сверху числом, зависящим исключительно от размера выборки и совокупной длины ее строк:

$$C \leqslant 2\zeta(|T|) + 2\zeta(L(T)) < 4\zeta(L(T)).$$

Поскольку ζ является функций медленного роста (рис. 3.2), зависимость от вида выборки T является слабой. На практике полученная оценка является сильно завышенной, как вследствие использования грубого приближения (3.27), так и за счет «нейтрализации» слагаемых, входящих в суммы Δ_3 и Δ_4:

$$\Delta_3 + \Delta_4 = \sum_{|x|=l} \Big(\sum_{|y|=1} N(xy)\delta\psi(xy) - N(x)\delta\psi(x)\Big) = \sum_{|x|=l} \sum_{|y|=1} N(xy)(\delta\psi(xy) - \delta\psi(x)),$$

где введено обозначение

$$\delta\psi(x) \stackrel{\text{def}}{=} \psi\left(\frac{N(\tilde{T}^+, x)}{N(T,x)}\right) - \psi\left(\frac{N(T^+,x)}{N(T,x)}\right), \quad x \in R_q^l \text{ или } x \in R_q^{l+1}.$$

Для большинства цепочек $\delta\psi(xy) \approx \delta\psi(x)$, откуда следует

$$|\Delta_3 + \Delta_4| \ll |\Delta_3| + |\Delta_4|.$$

Вследствие доказанной теоремы, для каждого набора цепочек X имеет смысл рассматривать сравнительно небольшое количество порогов, выбранных с целью покрыть все множество $\Omega(T, X)$, определенное согласно (3.26). В любом случае $\Omega(T, X) \subset (0, 1)$, однако внутренний вид этого множества зависит от аргументов T и X. Для того чтобы избавиться от такой зависимости, введем следующее понятие:

Определение 3.2. *Пусть задано множество строк $T \subset R_q^*$, набор цепочек $X \subset R_s^u$, число $r \in (0, 1)$. Тогда $\omega(T, X, r) \in \Omega(T, X)$ — пороговая концентрация, выбранная таким образом, чтобы соответствующий ей предикат делил выборку T в отношении $r : (1 - r)$:*

$$\forall \omega < \omega(T, X, r) \quad |\{Q \in T \,|\, I_{X,\omega}(\mathrm{Pr}_s(Q))\}| < r|T|;$$

$$|\{Q \in T \,|\, I_{X,\omega(T,x,r)}(\mathrm{Pr}_s(Q))\}| \geqslant r|T|.$$

Отображение $r \to \omega(T, X, r)$ при фиксированных параметрах T и X позволяет нормализовать множество $\Omega(T, X)$ на отрезок $[0, 1]$; таким образом, становится возможным дать определение понятию покрытия:

Определение 3.3. *Множество*

$$\Omega_\varepsilon(T, X) = \{\omega(T, X, r) \,|\, r = i\varepsilon, i = 1, 2, \ldots, \lfloor 1/\varepsilon \rfloor\}$$

называется ε-покрытием множества $\Omega(T, X)$.

Как утверждает теорема 3.3, при достаточно малом значении ε множество $\Omega_\varepsilon(T, X)$ содержит в себе пороговую концентрацию, близкую к локальной точке экстремума

$$\omega^* \stackrel{\text{def}}{=} \arg\max_\omega \Delta(T, I_{X,\omega}).$$

Этот набор порогов можно дополнительно сократить, если учесть, что отношение, в котором оптимальный предикат делит выборку, с высокой вероятностью не является чересчур большим или малым, т. е. лежит в определенном интервале $[r_{min}, 1 - r_{min}]$, $r_{min} < 1/2$. С учетом этого замечания отбор порогов следует производить согласно алгоритму 3.2. Сложность шагов 1 и 2 алгоритма составляет $\Theta(u|X|L(T))$ и $O(|T| \log |T|)$ соответственно. Поскольку $|T| \ll L(T)$, общая оценка сложности равна $\Theta(u|X|L(T))$.

§ 3.6. Построение множеств цепочек для использования в предикатах

Как было показано в разделе 3.4, для нахождения оптимального предиката во множестве $\mathfrak{I}_u \stackrel{\text{def}}{=} \{I_{X,\omega} \,|\, X \subset R_s^u, \omega \in (0, 1)\}$ достаточно рассмотреть $2^{a(u)-1} - 1$ различных множеств цепочек X ($a(u) = |R_s|^u$). Для обеих рассматриваемых задач перебор такого количества вариантов в общем случае нереален:

Алгоритм 3.2. Алгоритм определения множества пороговых концентраций

Вход: выборка $T \subset R_q^*$; множество цепочек $X \subset R_s^u$; числа ε, r_{min}

Выход: набор пороговых концентраций Ω

1: $A := \{n(X, \Pr_s(Q)) \mid Q \in T\}$ — концентрации для всех строк выборки
2: Отсортировать массив A по возрастанию.
3: $\Omega := \varnothing$;
4: **для** $j = \lceil r_{min}/\varepsilon \rceil, \ldots, \lfloor (1 - r_{min})/\varepsilon \rfloor$
5: $\quad i := \varepsilon j |T|$;
6: $\quad \Omega := \Omega \cup \{A_i\}$.

- Для задачи распознавания фрагментов генов ($|R_s| = 4$) полный перебор возможен при длине цепочек $u = 1$, что дает $2^3 - 1 = 7$ вариантов. Уже при $u = 2$ число вариантов равно $2^{15} - 1 = 32767$.
- Для задачи определения вторичной структуры белка ($|R_s| = 21$) количество возможных множеств аминокислот ($2^{20} - 1 \approx 10^6$) недопустимо велико уже при $u = 1$.

Таким образом, необходимо сокращение набора рассматриваемых множеств цепочек. Для решения этой задачи введем оценку качества предикатов, которые могут быть порождены на основе заданного множества цепочек X. В роли ее основы возьмем меру качества разбиения обучающей выборки T предикатом вида $I_{X,\omega}$ в отношении $1 : 1$ — в большинстве случаев это отношение близко к разбиению, порожденном оптимальным предикатом

$$I^*(X) = I_{X,\omega^*}, \quad \omega^* = \arg\max_{\omega} \Delta(T, I_{X,\omega}).$$

Более формально:

Определение 3.4. *Мерой качества набора цепочек $X \subset R_s^u$, вычисленной на основании обучающей выборки $T \subset R_q^*$, является величина*

$$\mathcal{L}(T, X) \stackrel{\text{def}}{=} \Delta(T, I_{X,\tilde{\omega}}),$$

где пороговая концентрация задана как

$$\tilde{\omega} = \omega(T, X, 1/2) = \min_{\omega} \Big(|\{Q \in T \mid I_{X,\omega}(\Pr_s(Q))\}| \geqslant \frac{1}{2}|T| \Big).$$

Сложность вычисления величины $\mathcal{L}$ зависит от алгоритмической сложности нахождения трех чисел:

- множества пороговых концентраций для всех строк выборки $\Omega(T, X)$ согласно формуле (3.26);
- концентрации $\tilde{\omega}$, которая, как следует из определения 3.4, является медианой множества $\Omega(T, X)$;
- меры качества разбиения $\Delta(T, I_{X,\tilde{\omega}})$.

Суммарная сложность нахождения $\mathcal{L}$, таким образом, составляет

$$O(u|X|L(T) + |T| + lL(T)) = O((u|X| + l)L(T)).$$

Алгоритм 3.3. Отбор цепочек X методом последовательного наращивания

Вход: выборка $T \subset R_q^*$; длина цепочек u; число сохраняемых цепочек n_{max}
Выход: множество наборов $\mathbf{X} \subset 2^{R_s^u}$

1: $\mathbf{X} := \{\varnothing\}$;
2: **для** $t = 1, 2, \ldots, [|R_s|^u/2]$ — текущий размер множеств X
3: $\quad \mathbf{Z} := \varnothing$; — множество для хранения наборов текущей длины
4: $\quad$ **для всех** $X \in \mathbf{X},\ |X| = t - 1$
5: $\quad\quad$ **для всех** $z \in R_s^u,\ z \notin X$
6: $\quad\quad\quad \mathbf{Z} := \mathbf{Z} \cup \{X \cup \{z\}\}$;
7: $\quad$ **если** $|\mathbf{Z}| > n_{max}$, **то**
8: $\quad\quad$ отсортировать $X \in \mathbf{Z}$ по убыванию величины $\mathcal{L}(T, X)$;
9: $\quad\quad$ оставить в $\mathbf{Z}$ первые n_{max} элементов;
10: $\quad \mathbf{X} := \mathbf{X} \cup \mathbf{Z}$.

С учетом введенной меры качества, нахождение ограниченного количества оптимальных множеств цепочек аналогично отбору информативных признаков в задачах машинного обучения; отдельным признакам при этом соответствуют элементы множества R_s^u. Приведем два метода отбора, основанные на известных алгоритмах, адаптированных для рассматриваемой задачи.

¶ **3.6.1. Последовательное добавление цепочек.** Один из простых способов отбора заключается в последовательном наращивании размера множеств цепочек X; согласно замечанию из раздела 3.4, достаточно рассмотреть наборы размером 1, 2,..., $[|R_s|^u/2]$. На каждой итерации будем сохранять определенное количество лучших множеств n_{max} (листинг 3.3).

Число наборов X, рассматриваемых алгоритмом, составляет, с учетом обозначения $a(u) = |R_s|^u$, $O(n_{max}a^2(u))$ — заметное улучшение по сравнению с полным перебором $O(2^{a(u)})$ подмножеств множества R_s^u.

¶ **3.6.2. Использование генетического алгоритма.** Для отбора информативных наборов цепочек может использоваться адаптированный генетический алгоритм, идея оптимизации в котором основана на принципах эволюции живых организмов. Для применения алгоритма введем на пространстве наборов $\Lambda_u \stackrel{\text{def}}{=} 2^{R_s^u}$ операции мутации и скрещивания.

Определение 3.5. *Мутацией набора цепочек называется вероятностная функция* $\text{mut} : \Lambda_u \times (0,1) \to \Lambda_u$*, вычисляемая как*

$$\text{mut}(X, p_m) = \bigcup_{x \in R_s^u} \text{mut}_0(x, \xi_x[x \notin X] + (1 - \xi_x)[x \in X]),$$

где ξ_x*,* $x \in R_s^u$ *— значения некоррелированных случайных величин, равномерно распределенных на отрезке* $[0, 1]$*,*

$$\text{mut}_0(x, \xi) = \begin{cases} \{x\}, & \text{если } \xi \leqslant p_m, \\ \varnothing, & \text{если } \xi > p_m; \end{cases}$$

$p_m \in (0,1)$ — *вероятность мутации отдельной цепочки.*

Согласно определению, каждый элемент множества $X \subset R_s^u$ с определенной вероятностью p_m может исчезнуть из подвергшегося мутации набора; аналогично, каждая из $|R_s|^u - |X|$ цепочек, отсутствовавших в наборе, может с этой же вероятностью в нем появиться.

Определение 3.6. *Скрещиванием двух наборов цепочек называется вероятностная функция* $\mathrm{cr} : \Lambda_u \times \Lambda_u \to \Lambda_u$*, вычисляемая по формуле*

$$\mathrm{cr}(X,Y) = \bigcup_{x \in R_s^u} \mathrm{cr}_0(x, \xi_x[x \in X] + (1-\xi_x)[x \in Y]),$$

где ξ_x*,* $x \in R_s^u$ *— значения некоррелированных случайных величин, равномерно распределенных на отрезке* $[0,1]$*,*

$$\mathrm{cr}_0(x,\xi) = \begin{cases} \varnothing, & \text{если } \xi \leqslant 1/2, \\ \{x\}, & \text{если } \xi > 1/2. \end{cases}$$

Как следует из определения, цепочки, присутствующие в обоих множествах X, Y, обязательно присутствуют в результате скрещивания $\mathrm{cr}(X,Y)$; цепочки, отсутствующие в обоих наборах, отсутствуют и в $\mathrm{cr}(X,Y)$. Для цепочек, принадлежащих разностям $X \setminus Y$ и $Y \setminus X$, вероятность попадания в результат скрещивания равна $1/2$.

Введенные выше понятия будем использовать в генетическом алгоритме для перехода между последовательными поколениями «организмов» (в данном случае множеств цепочек длины u): для каждого организма в текущем поколении произведем определенное количество мутаций (b) и скрещиваний с другими организмами (c), после чего отберем в следующее поколение не более n_{max} наборов с максимальным показателем качества (листинг 3.4). Общее количество наборов, рассматриваемых алгоритмом, составляет $O(Mn_{max}(b+c))$, где M — число сгенерированных поколений.

¶ **3.6.3. Дальнейшее сокращение множества наборов.** По аналогии с утверждением теоремы 3.3, для мало отличающихся наборов $X, Y \subset R_s^u$ близкими будут и значения меры качества $\mathcal{L}(T,X)$ и $\mathcal{L}(T,Y)$. В качестве меры близости в пространстве Λ_u будем использовать расстояние Хэмминга

$$d(X,Y) = \sum_{x \in R_s^u} ([x \in X] + [x \in Y]) \bmod 2.$$

Для сокращения количества рассматриваемых наборов цепочек подвергнем множество $\mathbf{X} \subset \Lambda_u$ преобразованию $\mathrm{Reduce}(\mathbf{X}, d_{min})$, в результате которого из него удаляются элементы, близкие по введенному расстоянию к другим элементам множества:

$$\forall X, Y \in \mathrm{Reduce}(\mathbf{X}, d_{min}) \quad d(X,Y) \geqslant d_{min}.$$

Для построения множества $\mathrm{Reduce}(\mathbf{X}, d_{min})$ будем последовательно перебирать элементы $\mathbf{X}$, удаляя для каждого из них окрестность радиуса $d_{min} - 1$. Результат при этом зависит от порядка рассмотрения элементов, что для рассматриваемой задачи несущественно.

Алгоритм 3.4. Генетический алгоритм отбора цепочек X

Вход: выборка $T \subset R_q^*$; длина цепочек u; размер начального поколения n_0; максимальный размер поколения n_{max}; количество мутаций b; вероятность элементарной мутации p_m; количество скрещиваний c; число поколений M

Выход: множество наборов $\mathbf{X} \subset \Lambda_u$

1: сформировать начальное поколение: $\mathbf{X} := \varnothing$;
2: **пока** $|X| < n_0$
3: Сформировать набор из случайных цепочек длины u:
$$Z := \bigcup_{x \in R_s^u} \mathrm{cr}_0(x, \xi_x), \quad \xi_x \sim [0,1];$$
4: $\mathbf{X} := \mathbf{X} \cup \{Z\}$;
5: **для** $t = 1, 2, \ldots, M$
6: **для всех** $X \in \mathbf{X}$, $i = 1, 2, \ldots, b$
7: $\mathbf{X} := \mathbf{X} \cup \{\mathrm{mut}(X, p_m)\}$;
8: **для всех** $X \in \mathbf{X}$, $i = 1, 2, \ldots, c$
9: Выбрать $Y \in \mathbf{X}$ — случайный элемент множества;
10: $\mathbf{X} := \mathbf{X} \cup \{\mathrm{cr}(X, Y)\}$;
11: **если** $|\mathbf{X}| > n_{max}$, **то**
12: Отсортировать массив $\mathbf{X}$ по убыванию величины $\mathcal{L}(T, X)$;
13: Оставить в $\mathbf{X}$ первые n_{max} элементов.

Сокращение множества рассматриваемых элементов будем проводить в ходе выполнения алгоритмов 3.3 и 3.4:

- для алгоритма последовательного наращивания наборов добавим преобразование $\mathbf{Z} := \mathrm{Reduce}(\mathbf{Z}, d_{min})$ между шагами 6 и 7;
- для генетического алгоритма добавим между шагами 10 и 11 сокращение текущего поколения $\mathbf{X} := \mathrm{Reduce}(\mathbf{X}, d_{min})$.

§ 3.7. Построение оптимального разбиения на области компетентности

Пусть $\mathrm{Pred}(T)$ — алгоритм построения определенного множества предикатов, составленный в соответствии с принципами, описанными в разделах 3.4–3.6. Результатом работы алгоритма является конечный набор предикатов $\{I_j\} \subset \mathfrak{I}_u$, близких к оптимальному по мере качества разбиения $\Delta(T, I)$. Согласно принципам построения оптимального разбиения, изложенным в разделе 3.1, на каждом из $k-1$ этапов нахождения оптимального разбиения обучающей выборки $\{T_i\}_{i=1}^k$ следует искать множество $T_r \subset T$ из текущего разбиения, разделение которого с помощью предикатов из $\mathrm{Pred}(T_r)$ позволяет добиться наибольшего увеличения совместной вероятности (3.21).

Разбиение пространства наблюдаемых строк $\{G_i\}_{i=1}^k$ в силу бесконечности составляющих его множеств будем хранить в виде бинарного дерева предикатов. Для описания этого дерева

$$D = \begin{pmatrix}(1, n(A) \gtrqless 29{,}2\%),\\ (1, n(A,T) \gtrqless 57{,}7\%),\\ (2, n(G) \gtrqless 19{,}9\%),\\ (3, n(G) \gtrqless 20{,}9\%),\\ (1, n(C) \gtrqless 21{,}4\%)\end{pmatrix}$$

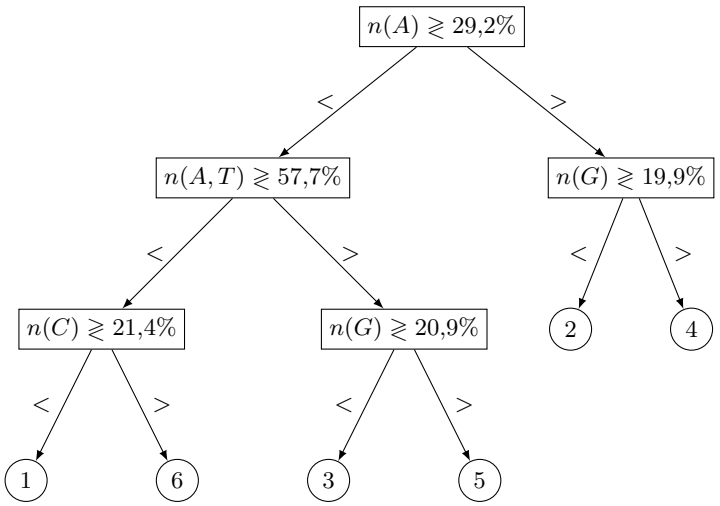


Рис. 3.3. Пример бинарного дерева разбиения и его описания для задачи распознавания фрагментов генов

достаточно на каждом шаге алгоритма запоминать индекс разделяемой области, а также сам оптимальный предикат (рис. 3.3).

С учетом вышесказанного, алгоритм построения оптимального разбиения формулируется в виде листинга 3.5; соответствующий ему алгоритм классификации, использующий композиции алгоритмов, приведен в следующем листинге. Метод выделения новых множеств в разбиении, соответствующий шагам 7 и 8 алгоритма 3.5, отображен в методе определения области G_i, которая содержит наблюдаемую строку, на шагах 1–5 алгоритма 3.6. Таким образом, любая строка из обучающей выборки будет классифицироваться алгоритмом, обученным на части выборки, содержащей эту строку:

$$\forall i = 1, \ldots, k,\ Q \in T_i \quad \mathcal{A}_c(\Pr_s(Q)) \equiv \mathcal{A}_i(\Pr_s(Q)).$$

Сложность алгоритма 3.5 определяется сложностью выполнения двух действий:

- определение множества предикатов $\mathbf{I}$ (шаг 4);
- подсчет величин $\{\Delta(T_i, I) \mid I \in \mathbf{I}\}$ (шаг 5).

Поскольку после каждого разбиения меняются только две области из множеств T_1,..., T_k, каждый из этих шагов выполняется $O(k)$ раз.

Так как нахождение множества наборов цепочек $\mathbf{X} \subset \Lambda_u$ — наиболее затратный этап вычисления $\text{Pred}(T_i)$, для ускорения алгоритма будем использовать оптимальные цепочки для всей выборки T, определенные до начала разбиения. При использовании алгоритма 3.3 сложность нахождения наборов составляет, с учетом оценок, полученных в разделе 3.6.1,

$$\Theta_{add} = O(n_{max} a^2(u)(ua(u) + l)L(T)),$$

где $a(u) = |R_s|^u$, n_{max} — число множеств цепочек, сохраняемых на каждой итерации. Если использовать генетический алгоритм 3.4, сложность составит

$$\Theta_{gen} = O(M n_{max}(b + c)(ua(u) + l)L(T)),$$

где M — количество поколений, n_{max} — максимальный размер поколения, b и c — число мутаций

Алгоритм 3.5. Построение оптимального разбиения

Вход: обучающая выборка $T \subset R_q^*$; порядок модели l;
количество областей в разбиении k

Выход: описание дерева предикатов $D \equiv (D_1, \ldots, D_{k-1})$;
алгоритмы $\mathcal{A}_1, \ldots, \mathcal{A}_k$

$T_1 := T$; — текущее разбиение обучающей выборки
для $t = 1, 2, \ldots, k-1$
    для $i = 1, 2 \ldots, t$
        $\mathbf{I} := \text{Pred}(T_i)$;
        $I_i^* := \arg\max_{I \in \mathbf{I}} \Delta(T_i, I)$; — оптимальный предикат для i-й части разбиения
    $r := \arg\max_{i=1,\ldots,t} \Delta(T_i, I_i^*)$; — часть выборки, которую следует разделить
    $T_{t+1} := \{Q \in T_r \mid \neg I_r^*(\text{Pr}_s(Q))\}$;
    $T_r := \{Q \in T_r \mid I_r^*(\text{Pr}_s(Q))\}$;
    $D_t := (r, I_r^*)$;
для $t = 1, 2, \ldots, k$
    Обучить алгоритм $\mathcal{A}_t$ на выборке T_t.

Алгоритм 3.6. Вычисление алгоритмической композиции

Вход: описание дерева предикатов $D \equiv (D_1, \ldots, D_{k-1})$; алгоритмы $\mathcal{A}_1, \ldots, \mathcal{A}_k$;
наблюдаемая строка S

Выход: цепочка скрытых состояний H, соответствующая строке S

$i := 1$; — индекс области G_i, которой принадлежит строка S
для $t = 1, 2, \ldots, k-1$
    $(r, I) := D_t$;
    если $(r = i)$ **и** $\neg I(S)$, **то**
        $i := t + 1$;
$H := \text{Pr}_h(\mathcal{A}_i(S))$.

и скрещиваний соответственно для каждого множества цепочек в популяции.

Шаг 4 алгоритма 3.5 при описанном выше подходе сводится к выполнению алгоритма 3.2 для $O(n_{max})$ наборов цепочек; его сложность равна

$$\Theta_{pred} = O(n_{max} ua(u) L(T)).$$

Подсчет качества разбиения, порождаемого всеми $O(n_{max}/\varepsilon)$ найденными предикатами, на 5-м шаге алгоритма имеет сложность

$$\Theta_{\Delta} = O(n_{max}(ua(u) + l)L(T)/\varepsilon).$$

Таким образом, общая сложность поиска оптимального разбиения составляет

$$\Theta_{add} + k(\Theta_{pred} + \Theta_{\Delta}) = O(n_{max}(ua(u) + l)(a^2(u) + k/\varepsilon)L(T))$$

при использовании алгоритма 3.3, и

$$\Theta_{gen} + k(\Theta_{pred} + \Theta_{\Delta}) = O(n_{max}(ua(u) + l)(M(b + c) + k/\varepsilon)L(T))$$

при использовании алгоритма 3.4.

§ 3.8. Выводы к разделу 3

Рассмотрены алгоритмические композиции описанных в разделе 2 вероятностных моделей с эксклюзивными областями компетентности составляющих алгоритмов; обоснована возможность их применения при условиях, заданных гипотезой 3.2. На основе принципа максимума правдоподобия получен алгоритм построения оптимального разбиения для композиции, использующий предикаты простого вида, зависящие от концентраций отдельных наблюдаемых символов или их коротких последовательностей. С целью ускорения построения оптимального разбиения предложены методы по отбору предикатов.

РАЗДЕЛ 4

ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ ПО ОЦЕНКЕ КАЧЕСТВА ПРЕДЛОЖЕННЫХ АЛГОРИТМОВ

§ 4.1. Метрики качества распознавания скрытых последовательностей

Для оценки качества работы алгоритма $\mathcal{A} : R_s^* \to R_q^*$, являющегося решением задачи 2.2, будем использовать принятые в литературе по биоинформатике метрики, вычисляемые на основе выборки, состоящей из пар строк наблюдаемых и соответствующих им скрытых состояний [65]. С помощью этих метрик оценивается качество предсказания как отдельных скрытых символов, так и непрерывных последовательностей, состоящих из символов одного типа (т. е., экзонов и интронов для задачи определения фрагментов генов; спиралей, β-листов и нерегулярных фрагментов для задачи предсказания вторичной структуры белка). Использование обоих классов оценок качества актуально с точки зрения выбранной предметной области: при анализе генома важно определение точных границ между экзонами и интронами; для прикладных задач, связанных с нахождением вторичной структуры белка, имеет смысл определение мест переходов между пространственными образованиями.

¶ 4.1.1. Символьные метрики.

¶ 4.1.1.1. Базовые метрики. Для анализа обобщающей способности алгоритма применим стандартные метрики из теории обработки сигналов — специфичность и чувствительность.

Определение 4.1. *Символьной специфичностью (specificity) распознавания класса скрытых символов $h \in R_h$ алгоритмом $\mathcal{A} : R_s^* \to R_q^*$, измеренной при помощи выборки $T \subset R_q^*$, называется отношение*

$$SSp_h = \frac{TP_h}{TP_h + FP_h},$$

где TP_h — количество символов из класса h, корректно отнесенных алгоритмом к своему классу:

$$TP_h \stackrel{\text{def}}{=} \sum_{Q \in T} \sum_{i=1}^{|Q|} [\mathrm{Pr}_h(Q_i) = h][\mathrm{Pr}_h(\mathcal{A}(\mathrm{Pr}_s(Q))_i) = h],$$

FP_h — количество символов из других классов, ошибочно отнесенных алгоритмом к классу h:

$$FP_h \stackrel{\text{def}}{=} \sum_{Q \in T} \sum_{i=1}^{|Q|} [\mathrm{Pr}_h(Q_i) \neq h][\mathrm{Pr}_h(\mathcal{A}(\mathrm{Pr}_s(Q))_i) = h].$$

Другими словами, символьная специфичность является долей правильно распознанных алгоритмом $\mathcal{A}$ символов h среди символов, которые алгоритм относит к этому классу.

Определение 4.2. *Символьная чувствительность (sensitivity) распознавания класса скрытых символов $h \in R_h$ алгоритмом $\mathcal{A} : R_s^* \to R_q^*$, измеренная с помощью выборки $T \subset R_q^*$, — это*

отношение

$$SSn_h = \frac{TP_h}{TP_h + FN_h}$$

где число TP_h определено так же, как и для предыдущего определения, FN_h — количество символов класса h, ошибочно отнесенных алгоритмом к другим классам скрытых символов:

$$FN_h \stackrel{\text{def}}{=} \sum_{Q \in T} \sum_{i=1}^{|Q|} [\mathbf{Pr}_h(Q_i) = h][\mathbf{Pr}_h(\mathcal{A}(\mathbf{Pr}_s(Q))_i) \neq h].$$

Иначе: символьная чувствительность по отношению к классу скрытых символов h является долей корректно распознанных алгоритмом $\mathcal{A}$ символов этого типа среди общего числа этих символов в выборке T.

Согласно определениям 4.1 и 4.2, значения символьной специфичности и чувствительности находятся в отрезке $[0, 1]$; чем выше значение этих величин, тем лучше исследуемый алгоритм. В то же время, значения SSp_h и SSn_h по отдельности плохо характеризуют качество распознавания [66]: так, для тривиального алгоритма $\mathcal{A}(S) = S \times h^{|S|}$, относящего все скрытые символы к одному классу, выполняется равенство $FN_h = 0$; соответственно, $SSn_h = 1$. Для устранения этого недостатка вводятся метрики, основанные на двух основных идеях:

- метрики, являющиеся агрегацией чувствительности и специфичности для конкретного класса скрытых состояний;
- меры качества, объединяющие две рассмотренных выше метрики для всех классов $h \in R_h$.

¶ 4.1.1.2. Агрегация специфичности и чувствительности.

Определение 4.3. *Коэффициентом корреляции алгоритма $\mathcal{A} : R_s^* \to R_q^*$ для класса $h \in R_h$, вычисленным по выборке $T \subset R_q^*$, называется отношение*

$$CC_h = \frac{TP_h \cdot TN_h - FP_h \cdot FN_h}{\sqrt{(TP_h + FN_h)(TP_h + FP_h)(TN_h + FP_h)(TN_h + FN_h)}},$$

где, помимо введенных в предыдущих определениях обозначений TP_h, FP_h и FN_h, используется выражение

$$TN_h \stackrel{\text{def}}{=} \sum_{Q \in T} \sum_{i=1}^{|Q|} [\mathbf{Pr}_h(Q_i) \neq h][\mathbf{Pr}_h(\mathcal{A}(\mathbf{Pr}_s(Q))_i) \neq h],$$

обозначающее количество символов, принадлежащих классам из множества $R_h \setminus h$, которые не были отнесены алгоритмом к типу h.

Значение CC_h лежит в пределах от -1 до 1; чем ближе это число к единице, тем сильнее связь между принадлежностью скрытого состояния к определенному классу и его классификацией в качестве такового алгоритмом $\mathcal{A}$. Наоборот, для алгоритма, подбирающего скрытые символы случайным образом, коэффициент корреляции близок к нулю.

Определение 4.4. *Средней условной вероятностью алгоритма $\mathcal{A} : R_s^* \to R_q^*$ по классу скрытых символов $h \in R_h$, вычисленной согласно выборке $T \subset R_q^*$, называется величина*

$$ACP_h = \frac{1}{4}\Big(\frac{TP_h}{TP_h + FN_h} + \frac{TP_h}{TP_h + FP_h} + \frac{TN_h}{TN_h + FP_h} + \frac{TN_h}{TN_h + FN_h}\Big).$$

ACP_h является арифметическим средним между специфичностью и чувствительностью для класса h, а также двумя этими характеристиками, вычисленными для совокупности остальных классов $R_h \setminus h$. Соответственно, значения средней условной вероятности находятся в отрезке $[0, 1]$; чем выше величина ACP_h, тем эффективнее соответствующий алгоритм.

Для задачи распознавания фрагментов генов (как и в произвольном случае при выполнении условия $|R_h| = 2$) значения коэффициента корреляции и средней условной вероятности совпадают для обоих классов скрытых состояний:

$$CC_{ex} = CC_{in}; \quad ACP_{ex} = ACP_{in}.$$

В самом деле, в силу наличия двух скрытых символов в алфавите R_h, справедливы равенства

$$TP_{ex} = TN_{in};\ TN_{ex} = TP_{in};\ FP_{ex} = FN_{in};\ FN_{ex} = FP_{in}. \tag{4.1}$$

Поскольку формулы для вычисления средней условной вероятности и коэффициента корреляции симметричны по отношению к парам переменных $\{TP_h, TN_h\}$ и $\{FP_h, FN_h\}$, из (4.1) следует, что значения этих метрик для обоих классов скрытых состояний совпадают. Более того, из равенств (4.1) следует, что для задачи определения экзонов и интронов выполняется тождество

$$ACP_{ex} = ACP_{in} = \frac{1}{4}\big(SSp_{ex} + SSp_{in} + SSn_{ex} + SSn_{in}\big),$$

то есть средняя условная вероятность может быть отнесена к обоим описанным выше типам агрегационных метрик.

¶ **4.1.1.3. Агрегация по классам скрытых состояний.** Ко второму типу агрегационных метрик, то есть к мерам качества, не зависящим от класса скрытых символов, относится доля правильно предсказанных состояний, определяемая следующим образом:

Определение 4.5. *Долей корректно распознанных алгоритмом $\mathcal{A} : R_s^* \to R_q^*$ скрытых символов, вычисленной на основе выборки $T \subset R_q^*$, называется отношение*

$$SC = \frac{1}{L(T)} \sum_{Q \in T} \sum_{i=1}^{|Q|} [Q_i = \mathcal{A}(\mathrm{Pr}_s(Q))_i],$$

где $L(T)$ — суммарная длина последовательностей в выборке.

Из определения следует $0 \leqslant SC \leqslant 1$; чем выше значение метрики, тем эффективнее оцениваемый ей алгоритм. В силу выполнения равенств

$$SC = \sum_{h \in R_h} SSp_h \frac{TP_h + FP_h}{L(T)} = \sum_{h \in R_h} SSn_h \frac{TP_h + FN_h}{L(T)},$$

меру SC можно рассматривать как взвешенную сумму специфичностей или чувствительностей по отдельным классам, причем веса отдельных компонент равны долям соответствующих скрытых состояний в выборке (для чувствительностей) либо в генерируемом алгоритмом выходе (для специфичностей). Как следствие, выполняются неравенства

$$\max(\min_h SSn_h, \min_h SSp_h) \leqslant SC \leqslant \min(\max_h SSn_h, \max_h SSp_h).$$

¶ 4.1.2. Фрагментные метрики.

Определение 4.6. *Фрагментом класса скрытых символов $h \in R_h$ для строки $Q \in R_q^*$ называется ее подстрока $Q_{i\ldots j}$, скрытая часть которой состоит исключительно из этих символов и обособлена от соседних состояний:*

$$\Pr_h(Q_{i\ldots j}) = h^{j-i+1};\ (i=1) \vee (\Pr_h(Q_{i-1}) \neq h);\ (j=|Q|) \vee (\Pr_h(Q_{j+1}) \neq h).$$

Каждая цепочка полных состояний $Q \in R_q^*$, в соответствии с определением 4.6, однозначно соответствует множеству входящих в нее фрагментов, задающихся тремя величинами — классом скрытых символов $h \in R_h$, а также начальным и конечным положениями:

$$Q \sim R(Q) = \{(h_1, i_1, i_2 - 1), (h_2, i_2, i_3 - 1), \ldots, (h_k, i_k, i_{k+1} - 1)\},$$

где

$$1 = i_1 < i_2 < \cdots < i_k < i_{k+1} = |Q| + 1;\ \forall t = 2, \ldots, k \quad h_t \neq h_{t-1}.$$

В случае задачи распознавания фрагментов генов фрагментами являются нуклеотидные последовательности, соответствующие экзонам и интронам; для задачи определения вторичной структуры белков фрагменты — цепочки аминокислот, соответствующие непрерывным вторичным структурам. Метрики, связанные с фрагментами, оценивают качество восстановления фрагментов различных классов конкретным алгоритмом.

Определение 4.7. *Фрагмент (h, i, j), соответствующий строке Q, будем считать распознанным алгоритмом $\mathcal{A} : R_s^* \to R_q^*$ тогда и только тогда, когда в выходе алгоритма $\tilde{Q} = \mathcal{A}(\Pr_s(Q))$ содержится фрагмент с теми же параметрами, т. е. $(h, i, j) \in R(\tilde{Q})$.*

Определение 4.8. *Фрагментной специфичностью алгоритма $\mathcal{A} : R_s^* \to R_q^*$ для класса скрытых символов $h \in R_h$, вычисленной на основе выборки $T \subset R_q^*$, является отношение количества корректно распознанных алгоритмом фрагментов этого класса к общему количеству фрагментов типа h, содержащемуся в его выводе:*

$$RSp_h = TR_h / PR_h;$$

$$TR_h = \sum_{Q \in T} \sum_{i,j} [(h, i, j) \in R(Q)][(h, i, j) \in R(\tilde{Q})];$$

$$TR_h = \sum_{Q \in T} \sum_{i,j} [(h, i, j) \in R(\tilde{Q})],$$

где использовано сокращение $\tilde{Q} = \mathcal{A}(\Pr_s(Q))$.

Определение 4.9. *Фрагментной чувствительностью алгоритма $\mathcal{A} : R_s^* \to R_q^*$ для класса $h \in R_h$, вычисленной по выборке $T \subset R_q^*$, называется отношение числа правильно определенных алгоритмом фрагментов класса h к суммарному количеству этих фрагментов в выборке:*

$$RSp_h = TR_h / AR_h;$$

$$AR_h = \sum_{Q \in T} \sum_{i,j} [(h, i, j) \in R(Q)].$$

Согласно определениям 4.8 и 4.9, фрагментная чувствительность и специфичность находятся в пределах от 0 до 1, причем эффективным алгоритмам соответствуют более высокие значения. При разумных предположениях о длине фрагментов в выборке и выводе алгоритма $\mathcal{A}$ соблюдается неравенство

$$RSp_h < SSp_h; \quad RSn_h < SSn_h;$$

т. е. из высокого значения фрагментных метрик следует высокое качество алгоритма по отношению к символьным мерам. Обратное неверно: даже при $SSp_h \approx SSn_h \approx 1$ значения RSp_h и RSn_h может быть близкими к нулю.

§ 4.2. Методика вычислительного эксперимента

¶ **4.2.1. Используемые выборки.** Для определения оптимальных параметров рассматриваемых в разделе 2 вероятностных моделей и их композиций (раздел 3), а также для определения границ их применения для задачи распознавания фрагментов генов будем использовать геномы эукариотов, взятые из репозитория Национального центра биоинформатики США (NCBI) [67]. Рассмотрим ДНК следующих биологических видов (табл. 4.1):

- круглый червь *C. elegans*;
- растения — *Arabidopsis thaliana* (резуховидка Таля), *Glycine max* (соя), *Medicago truncatula* (растение семейства бобовых), *Oryza sativa* (рис посевной), *Populus trichocarpa* (тополь), *Vitis vinifera* (культурный вид винограда);
- насекомые — *Anopheles gambiae* (малярийный комар), *Apis mellifera* (медоносная пчела), *Drosophila melanogaster*, *Drosophila yakuba* (плодовые мушки), *Nasonia vitripennis* (паразитическая оса), *Tribolium castaneum* (хрущак — жук-вредитель);
- рыбы — *Danio rerio* (тропическая рыбка), *Oreochromis niloticus* (нильская тиляпия), *Oryzias latipes* (японская оризия), *Takifugu rubripes* (рыба-фугу);
- птицы — *Gallus gallus* (курица);
- млекопитающие — *Homo sapiens* (человек), *Sus scrofa* (свинья), *Mus musculus* (мышь), *Rattus norvegicus* (крыса), *Papio anubis* (павиан).

Во внимание будем принимать гены с полностью известной нуклеотидной записью; с целью ускорения работы алгоритмов для геномов человека и мыши, включающих в себя определенное количество чрезвычайно длинных последовательностей, введем ограничение сверху на их длину: $|Q| \leqslant 40000$.

Для проверки справедливости гипотез 2.2–2.4 также будем использовать ДНК семи видов бактерий (табл. 4.2).

Для задачи определения вторичной структуры белков оптимизацию параметров моделей будем проводить на основе данных, взятых из базы Центра молекулярной и биоинформатики (CMBI) [68]. Она содержит сведения о вторичной структуре для 76907 белков суммарной длиной 46293781 аминокислот. Из-за наличия в выборке дублирующихся последовательностей, а также альтернативных вариантов вторичной структуры для одного и того же белка для проверки качества классификации будем использовать модифицированный вариант выборки, из которого

Таблица 4.1

Сводные данные по используемым геномам

Вид	Число генов	Длина гена (без UTR)	Экзонов на ген	Длина экзона	Длина интрона	Плотность экзонов, %
C. elegans	23818	3100	6,43	205,8	326,8	42,7
Anopheles gambiae	12734	4593	4,03	384,3	1006,0	33,7
Apis mellifera	7980	4370	6,01	238,8	585,5	32,9
Drosophila melanogaster	22102	6405	4,95	370,1	1157,0	28,6
Drosophila yakuba	14131	3164	3,66	388,8	655,1	45,0
Nasonia vitripennis	10254	3345	6,17	250,8	348,3	46,2
Tribolium castaneum	7978	4191	5,18	297,9	634,3	36,8
Arabidopsis thaliana	35176	1954	5,60	220,4	156,7	63,1
Glycine max	43297	3934	5,70	225,6	563,5	32,7
Medicago truncatula	42344	2175	3,98	222,9	433,2	40,7
Oryza sativa	34127	2325	4,20	259,8	386,3	46,9
Populus trichocarpa	28063	2430	4,71	229,0	364,2	44,4
Vitis vinifera	20304	4983	5,44	247,4	819,7	27,0
Danio rerio	21381	11853	7,47	181,8	1623,2	11,4
Oreochromis niloticus	23249	9296	9,59	170,9	891,1	17,6
Oryzias latipes	8268	4258	6,68	181,2	536,9	28,4
Takifugu rubripes	13103	6250	10,26	161,2	496,7	26,5
Homo sapiens	22250	12362	6,92	178,3	1881,2	10,0
Gallus gallus	14301	23383	10,04	158,7	2410,3	6,8
Sus scrofa	14524	15440	6,56	181,7	2562,1	7,7
Mus musculus	20913	10874	6,65	188,1	1703,7	11,5
Rattus norvegicus	15540	10154	5,95	201,0	1808,7	11,8
Papio anubis	15127	11363	6,04	192,7	2022,6	10,2

Таблица 4.2

Общая информация по исследуемым бактериям

Вид	Число хромосом	Длина генома	Число белков	Длина белков		ρ, %
				общая	средняя	
Bifidobacterium lactis	1 к.[1)]	1938709	1567	554371	354	42,89
C. glutamicum	1 к.	3309401	2993	950323	318	43,07
Gordonia bronchialis	1 к.	5208602	4616	1500908	325	43,22
Mycobacterium gilvum	1 к.	5619607	5241	1728118	330	46,13
P. acnes	1 к.	2560265	2297	761899	332	44,64
R. erythropolis	1 к.	6516310	6030	1980605	328	45,59
S. avermitilis	1 лин.[2)]	9025608	7580	2588611	342	43,02

Примечания:

1) кольцевая хромосома;
2) линейная хромосома.

удалены последовательности с одинаковыми именами, — 26108 белков общей длиной 14070736 аминокислот.

¶ **4.2.2. Методика кросс-валидации.** Непосредственное использование целых геномов либо полного набора белков в качестве выборки, на которой производится обучение алгоритма рас-

познавания и вычисление мер качества, не позволяет найти оптимальные параметры исследуемых моделей. Действительно, вследствие того, что в этом случае критерий качества является внутренним по отношению к используемому множеству прецедентов, при увеличении сложности модели наблюдается переобучение (*overfitting*) — неустойчивость алгоритма к малым изменениям обучающей выборки [69]. Один из способов решения этой проблемы — применение внешних критериев качества, т. е. независимость обучающей выборки и набора строк полных состояний, используемых при вычислении введенных в разделе 4.1 метрик [70].

В соответствии с вышесказанным, для оценки качества предсказания будем использовать n-кратную кросс-валидацию, при которой доступная выборка делится случайным образом на n приблизительно равных частей, каждая из которых по очереди выступает в роли контроля, а оставшиеся $n-1$ частей служат обучающей выборкой для исследуемого алгоритма. Результаты, полученные для всех n запусков, усредняются, нивелируя возможные нерегулярности, возникшие при разбиении выборки на составные части. Для задачи определения фрагментов генов количество частей выборки положим равным $n = 5$ в соответствии с общими рекомендациями; в случае предсказания вторичной структуры белка из-за меньшего объема доступных данных будем применять десятикратную кросс-валидацию.

¶ **4.2.3. Среда тестирования.** Для реализации описанных в разделах 2–3 алгоритмов разработан программный комплекс на языке программирования Java, описанный в приложении A. При тестировании используется операционная среда на основе персонального компьютера с характеристиками согласно табл. 4.3. В силу больших объемов обрабатываемых данных выбраны 64-битные версии операционной системы и виртуальной машины Java.

Таблица 4.3

Характеристики среды тестирования

Характеристика	Значение
Оборудование	
Процессор	Intel Core i7 Q720; 1,6 ГГц, 8 потоков исполнения
Оперативная память	8 Гб DDR3 SDRAM
Жесткий диск	600 Гб, SATA II; 7200 оборотов в минуту
Программное обеспечение	
Операционная система	Ubuntu GNU/Linux 12.04 x86_64
Виртуальная машина Java	OpenJDK 1.6.27
Параметры виртуальной машины	`-Xms512m -Xmx5g`

§ 4.3. Проверка выполнения симметрии в ДНК и белках

Помимо семи видов бактерий, приведенных в табл. 4.2, для проверки гипотез 2.2–2.4 будем использовать геномы плоского червя *C. elegans*, медоносной пчелы *Apis mellifera*, плодовой мушки *Drosophila melanogaster*, тропической рыбки *Danio rerio* и растений *Populus trichocarpa* (тополь) и *Vitis vinifera* (виноград). Для оценки точности выполнения гипотез 2.2 и 2.3 вычислим абсолютную и относительную разность между числом встреч отдельных аминокислот и

их пар в пределах каждой из нитей (проверять симметричность для более длинных последовательностей аминокислот нецелесообразно, так как число вхождений в белки отдельных цепочек длиной 3 и выше слишком мало). Относительные разности при этом подсчитываются по формулам

$$\varepsilon_n(x) = \frac{|N_1(x) - N_2(x)|}{\max\{N_1(x), N_2(x)\}};$$
$$\varepsilon_f(x) = \frac{|f_1(x) - f_2(x)|}{\max\{f_1(x), f_2(x)\}},$$

где x — аминокислота или пара аминокислот; N_1, N_2 — число вхождений последовательности x в белки, синтезируемые из генов, расположенных на соответствующей нити ДНК; f_1, f_2 — доли x среди всех последовательностей той же длины в пределах одной нити.

Результаты для белков *C. elegans* приведены: для отдельных аминокислот — в табл. 4.4, для двадцати наиболее распространенных пар аминокислот — в табл. 4.5 [71; 72]. Для остальных организмов итоги в целом аналогичны.

Таблица 4.4

Симметрия по отдельным аминокислотам для белков *Caenorhabditis elegans*

АК	Количество			Частота, %			АК	Количество			Частота, %		
	N_1	N_2	ε_n	f_1	f_2	ε_f		N_1	N_2	ε_n	f_1	f_2	ε_f
A	340748	328970	3,46	6,39	6,38	0,13	M	139418	135306	2,95	2,61	2,62	0,39
C	106685	103890	2,62	2,00	2,02	0,73	N	258940	251406	2,91	4,86	4,88	0,43
D	288541	274893	4,73	5,41	5,33	1,45	P	265578	254727	4,09	4,98	4,94	0,78
E	355398	339859	4,37	6,66	6,59	1,08	Q	223260	216276	3,13	4,19	4,20	0,21
F	246764	241401	2,17	4,63	4,68	1,18	R	276677	268958	2,79	5,19	5,22	0,55
G	284622	278687	2,09	5,34	5,41	1,27	S	434562	416359	4,19	8,15	8,08	0,89
H	122517	119700	2,30	2,30	2,32	1,05	T	316097	301751	4,54	5,93	5,85	1,25
I	322613	313674	2,77	6,05	6,08	0,57	V	331505	322291	2,78	6,22	6,25	0,56
K	338555	326479	3,57	6,35	6,33	0,25	W	57709	56708	1,73	1,08	1,10	1,62
L	455951	441969	3,07	8,55	8,57	0,27	Y	166662	162010	2,79	3,13	3,14	0,55

Таблица 4.5

Симметрия по наиболее распространенным парам аминокислот для белков *C. elegans*

АК	Количество			Частота, %			АК	Количество			Частота, %		
	N_1	N_2	ε_n	f_1	f_2	ε_f		N_1	N_2	ε_n	f_1	f_2	ε_f
SS	47865	45880	4,15	0,90	0,89	0,85	KL	29924	29359	1,89	0,56	0,57	1,47
LL	43726	42189	3,52	0,82	0,82	0,19	LE	30190	28831	4,50	0,57	0,56	1,21
LS	35545	34013	4,31	0,67	0,66	1,02	VL	29256	28642	2,10	0,55	0,56	1,26
SL	34307	33806	1,46	0,64	0,66	1,90	IL	29194	28423	2,64	0,55	0,55	0,71
EE	32866	30560	7,02	0,62	0,59	3,82	AA	29277	27891	4,73	0,55	0,54	1,45
KK	31778	30632	3,61	0,60	0,59	0,29	TS	28635	28001	2,21	0,54	0,54	1,14
EK	31302	30580	2,31	0,59	0,59	1,05	LI	28466	27500	3,39	0,53	0,53	0,07
ST	31654	29026	8,30	0,59	0,56	5,15	AL	28325	27464	3,04	0,53	0,53	0,30
LK	30680	29949	2,38	0,58	0,58	0,97	EL	28487	27036	5,09	0,53	0,52	1,83
LA	30295	29336	3,17	0,57	0,57	0,17	SA	27828	26522	4,69	0,52	0,51	1,41

Результаты проверки гипотез 2.2 и 2.3 подытожены в табл. 4.6 и табл. 4.7; они содержат максимальные, средние и взвешенные средние значения разностей, а также среднеквадратичные отклонения. Взвешенные значения при этом считаются по формулам

$$\bar{\varepsilon}_n = \sum_x f(x)\varepsilon_n(x), \quad \bar{\varepsilon}_f = \sum_x f(x)\varepsilon_f(x),$$

где суммирование производится по всем аминокислотам или парам аминокислот, а через

$$f(x) = \frac{N_1(x) + N_2(x)}{\sum_y N_1(y) + N_2(y)}$$

обозначена частота появления последовательности x во всех белках организма. Видно, что симметрия по отдельным аминокислотам и их парам выполняется с точностью порядка 1%. Бóльшие значения максимальных разностей для пар объясняются сравнительно малым размером выборки — каждая пара аминокислот встречается в белках в среднем несколько тысяч раз.

Таблица 4.6

Выполнение соотношений комплементарности для отдельных аминокислот

Вид	Разность по количеству, %				Разность по частоте, %			
	макс.	средн.	взвеш.	откл.	макс.	средн.	взвеш.	откл.
Bifidobacterium lactis	9,99	6,01	5,89	2,08	4,34	1,84	1,70	1,18
C. glutamicum	11,56	8,44	8,31	1,93	5,59	1,67	1,44	1,21
Gordonia bronchialis	5,97	2,72	2,91	1,38	3,77	1,34	0,96	1,12
Mycobacterium gilvum	9,22	6,64	7,04	1,61	4,49	1,33	1,01	1,12
P. acnes	6,46	3,77	3,77	1,41	2,79	1,23	1,25	0,77
R. erythropolis	9,35	6,02	6,07	1,69	5,83	1,15	0,79	1,30
S. avermitilis	13,40	10,04	10,35	1,81	4,67	1,62	1,17	1,17
Apis mellifera	7,26	5,52	5,68	0,94	1,82	0,89	0,86	0,46
C. elegans	4,73	3,15	3,32	0,83	1,62	0,76	0,72	0,43
D. discoideum	5,47	2,57	2,95	1,34	2,69	1,21	1,01	0,74
Danio rerio	1,88	0,76	0,74	0,45	1,85	0,77	0,73	0,45
D. melanogaster	10,44	5,83	5,39	1,78	5,33	1,59	1,40	1,09
Populus trichocarpa	2,42	1,11	1,09	0,67	1,60	0,65	0,57	0,44
Vitis vinifera	1,49	0,81	0,76	0,35	0,85	0,30	0,29	0,24

Для оценки выполнения гипотезы 2.4 вычислим суммарное количество вхождений упомянутых в ее формулировке групп аминокислот в белки организмов и относительную разность между соответствующими друг другу группами. Так как число стоп-кодонов, которое совпадает с количеством белков организма, в несколько раз меньше числа вхождений в белки любой аминокислоты, этими кодонами можно пренебречь. Результаты вычислительного эксперимента (табл. 4.8) свидетельствуют о том, что по сравнению с двумя другими предположениями гипотеза 2.4 выполняется хуже — в некоторых случаях суммарное количество «симметричных» аминокислот различается на треть.

Таблица 4.7

Выполнение соотношений комплементарности для пар аминокислот

Вид	Разность по количеству, %				Разность по частоте, %			
	макс.	средн.	взвеш.	откл.	макс.	средн.	взвеш.	откл.
Bifidobacterium lactis	32,18	8,02	6,61	5,91	27,93	6,06	4,54	5,23
C. glutamicum	33,75	9,26	8,47	5,44	27,74	4,98	3,52	4,45
Gordonia bronchialis	26,36	4,53	3,70	3,72	24,21	3,99	2,61	3,71
Mycobacterium gilvum	22,89	7,05	7,16	3,88	18,56	3,81	2,63	3,49
P. acnes	34,51	5,99	4,69	5,14	31,94	5,15	3,52	4,74
R. erythropolis	24,60	6,62	6,20	3,83	19,72	3,51	2,28	3,35
S. avermitilis	28,50	9,79	10,32	4,42	24,44	3,84	2,55	3,48
Apis mellifera	12,58	5,54	5,73	2,49	14,29	2,22	1,85	1,88
C. elegans	10,69	3,37	3,47	2,09	12,82	2,03	1,77	1,73
D. discoideum	13,46	3,07	3,28	2,03	16,02	2,33	1,89	2,14
Danio rerio	9,58	1,77	1,55	1,45	9,60	1,77	1,55	1,45
D. melanogaster	18,44	6,44	5,47	3,38	13,79	2,99	2,54	2,53
Populus trichocarpa	9,00	1,71	1,48	1,43	8,04	1,49	1,22	1,32
Vitis vinifera	7,31	1,66	1,37	1,35	7,28	1,53	1,27	1,22

Таблица 4.8

Соотношения комплементарности для групп аминокислот

Вид	D, H, N, Y	I, M, V	ε, %	F, L	K, E, Q	ε, %	C, G, R, S, W	A, P, T	ε, %
B. lactis	83724	88945	5,87	67945	74364	8,63	121974	117419	3,73
C. glutamicum	129315	152729	15,33	126379	126003	0,30	208827	207070	0,84
G. bronchialis	197684	230905	14,39	185443	148671	19,83	363893	374312	2,78
M. gilvum	224784	260156	13,60	219693	177641	19,14	414750	431094	3,79
P. acnes	101243	124264	18,53	93195	85520	8,24	184972	172705	6,63
R. erythropolis	249848	310419	19,51	258472	217586	15,82	475091	469189	1,24
S. avermitilis	316008	342185	7,65	335527	278025	17,14	649624	667242	2,64
Apis mellifera	844598	743177	12,01	649618	927621	29,97	1121657	847396	24,45
C. elegans	1644669	1564807	4,86	1386085	1799827	22,99	2284857	1807871	20,88
D. discoideum	1529781	998502	34,73	929109	1297450	28,39	1334739	907724	31,99
Danio rerio	2020394	1845804	8,64	1822290	2428822	24,97	3320945	2405173	27,58
D. melanogaster	2087069	1761288	15,61	1659885	2338227	29,01	3102498	2531037	18,42
P. trichocarpa	1533850	1478278	3,62	1446650	1679307	13,85	2465075	1696851	31,16
Vitis vinifera	1467727	1448372	1,32	1437123	1601782	10,28	2430121	1662218	31,60

§ 4.4. Оценка качества отдельных вероятностных моделей

¶ **4.4.1. Распознавание фрагментов генов.** Для определения оптимальных длин l и m в вероятностной модели $\mathcal{M}(l, m)$, описанной в разделе 2.4, рассмотрим два генома, часто используемые в профильных исследованиях: плодовой мушки *Drosophila melanogaster* и резуховидки Таля *Arabidopsis thaliana*. Проверим значения параметров $m \in [1, 4]$ и $l \in [m, 8-m]$: при $l+m > 8$ количество скрытых параметров модели — $\Theta((l+m) \cdot 4^{l+m})$ — становится чрезмерно большим, что проявляется в сильном переобучении. Таким образом, исследовать бóльшие значения параметров не представляется целесообразным.

Усредненные показатели качества для экзонов — для более важного класса скрытых состояний в рассматриваемой задаче — приведены в табл. 4.9 и 4.10 [73; 74]. Как и следует ожидать, показатели качества классификации, полученные на обучающей выборке, растут при увеличении значений l и m, в том числе когда один из этих параметров является фиксированным. Вместе с тем, при больших значениях становится выраженным переобучение, проявляющееся в разнице меду оценками качества для контрольной и обучающей выборок. Среди моделей с одинаковой суммой $l + m$, т. е. обладающих приблизительно равным количеством скрытых параметров, наиболее высокий результат показывает модель с длиной зависимого фрагмента состояний $m = 1$.

Таблица 4.9

Качество алгоритма 2.3 с разными параметрами на геноме *Arabidopsis thaliana*

(m, l)	SSp_{ex}, %		SSn_{ex}, %		CC_{ex}, %		ACP_{ex}, %		RSp_{ex}, %		RSn_{ex}, %	
	об.[1)]	к.[2)]	об.	к.	об.	к.	об.	к.	об.	к.	об.	к.
$(1, 1)$	86,69	86,69	88,89	88,89	66,16	66,15	83,08	83,08	26,67	26,67	16,22	16,21
$(1, 2)$	92,33	92,32	92,17	92,17	79,00	79,00	89,50	89,50	52,53	52,54	43,18	43,19
$(1, 3)$	94,65	94,64	94,95	94,95	85,86	85,84	92,93	92,92	66,71	66,68	62,05	62,02
$(1, 4)$	95,60	95,57	95,79	95,77	88,29	88,86	94,15	94,43	71,08	71,03	68,33	68,25
$(1, 5)$	96,44	96,37	96,74	96,71	90,74	90,58	95,37	95,29	76,03	75,77	75,39	75,04
$(1, 6)$	96,68	96,42	97,06	96,90	91,48	90,91	95,74	95,45	77,52	76,35	77,45	75,99
$(1, 7)$	97,25	96,26	97,54	96,89	92,91	90,65	96,46	95,33	80,12	75,39	80,93	75,07
$(2, 2)$	93,62	93,60	93,55	93,55	82,61	82,58	91,31	91,29	59,43	59,41	52,44	52,40
$(2, 3)$	95,22	95,17	95,37	95,37	87,22	87,15	93,61	93,57	69,02	68,94	65,39	65,28
$(2, 4)$	96,08	96,00	96,28	96,23	89,63	89,44	94,81	94,72	73,70	73,32	72,12	71,68
$(2, 5)$	96,66	96,34	96,96	96,74	91,31	90,58	95,66	95,29	77,04	75,56	76,86	75,01
$(2, 6)$	97,21	95,84	97,54	96,74	92,86	89,85	96,43	94,93	79,82	73,68	80,64	73,07
$(3, 3)$	95,96	95,87	95,41	95,35	88,35	88,15	94,18	94,07	71,47	71,05	68,76	68,27
$(3, 4)$	96,61	96,29	96,31	96,05	90,42	89,63	95,21	94,82	75,30	73,65	74,25	72,24
$(3, 5)$	97,31	95,87	97,21	96,28	92,58	89,30	96,29	94,65	79,30	72,83	79,76	71,83
$(4, 4)$	97,00	95,35	97,19	96,31	92,10	88,60	96,05	94,30	77,97	70,93	78,31	69,87

Примечания:

1) обучающая выборка;
2) контрольная выборка.

В соответствии с результатами, полученными на первой стадии эксперимента, для остальных исследуемых видов будем проверять модели вида $\mathcal{M}(l, 1)$ как наиболее точно описывающие структуру генов [75]. Результаты по всем организмам подытожены в табл. 4.11. Качество алгоритмов на контрольной выборке свидетельствует о применимости моделей $\mathcal{M}(l, 1)$ к поставленной задаче для генома растений и насекомых; так, в [21] указывается порядок значений метрик для современных алгоритмов

$$SSp_{ex} \approx SSn_{ex} \approx 90\%; \quad RSp_{ex} \approx RSn_{ex} \approx 45\text{–}75\%.$$

В то же время, для рыб, птиц и млекопитающих в силу более сложного устройства генов качество классификации исследуемых алгоритмов является недостаточно высоким.

Для определения границ применимости моделей проверим возможность классификации

Таблица 4.10

Качество алгоритма 2.3 с разными параметрами на геноме *Drosophila melanogaster*

(m,l)	SSp_{ex}, %		SSn_{ex}, %		CC_{ex}, %		ACP_{ex}, %		RSp_{ex}, %		RSn_{ex}, %	
	об.[1)]	к.[2)]	об.	к.	об.	к.	об.	к.	об.	к.	об.	к.
$(1,1)$	68,73	68,72	83,43	83,43	64,75	64,75	82,42	82,42	11,76	11,74	9,18	9,17
$(1,2)$	75,45	75,57	86,37	86,38	71,68	71,79	85,86	85,92	27,40	27,44	23,92	23,94
$(1,3)$	78,73	78,33	87,62	87,66	75,23	74,92	87,63	87,48	38,59	38,48	35,41	35,27
$(1,4)$	80,95	80,37	88,87	88,95	77,87	77,50	88,95	88,77	48,37	48,36	46,84	46,81
$(1,5)$	84,02	83,56	90,64	90,74	81,51	81,23	90,76	90,62	61,76	61,57	62,01	61,61
$(1,6)$	86,86	86,03	92,14	91,74	84,75	83,82	92,38	91,92	69,62	68,04	69,00	66,80
$(1,7)$	89,53	87,92	93,87	92,60	87,98	85,87	93,99	92,94	77,11	70,46	77,21	67,80
$(2,2)$	77,17	77,12	86,90	86,89	73,48	73,42	86,76	86,73	32,56	32,55	29,12	29,10
$(2,3)$	79,98	79,56	88,22	88,19	76,66	76,30	88,34	88,16	44,06	43,98	41,49	41,37
$(2,4)$	82,69	82,16	89,84	89,73	79,92	79,42	89,97	89,72	55,33	54,86	54,62	53,99
$(2,5)$	86,06	85,19	91,67	91,25	83,80	82,84	91,91	91,43	66,81	64,79	66,69	63,96
$(2,6)$	89,53	87,74	93,80	92,26	87,93	85,49	93,97	92,75	76,68	68,39	76,60	65,23
$(3,3)$	82,23	81,74	88,76	88,65	78,83	78,37	89,43	89,19	51,40	50,91	49,15	48,53
$(3,4)$	85,14	84,39	90,61	90,18	82,37	81,49	91,19	90,75	62,30	60,23	61,33	58,72
$(3,5)$	89,17	87,33	92,98	91,30	87,09	84,54	93,55	92,27	74,28	65,88	74,07	62,85
$(4,4)$	87,72	86,02	92,79	91,19	85,86	83,45	92,93	91,73	70,29	61,79	70,21	59,39

Примечания:

1) обучающая выборка;
2) контрольная выборка.

участков генов, принадлежащих некоторому виду, моделью, обученной на другом, родственном, организме [76]. Для этого переберем всевозможные пары среди трех групп, к которым принадлежат выбранные для исследования виды (растения, насекомые и рыбы), для моделей данных $\mathcal{M}(6,1)$ и $\mathcal{M}(7,1)$. Этот этап эксперимента соответствует $2 \cdot 6^2 = 72$ запускам программы для насекомых и растений и $2 \cdot 4^2 = 32$ запускам для геномов рыб.

Наиболее точно отражающие качество классификации метрики — специфичность и чувствительность по экзонам — для лучшей из двух моделей для каждой пары родственных организмов приведены в таблицах 4.12–4.14. Как показывает эксперимент, достаточно высокое качество классификации ($RSp_{ex} \geqslant 50$%, $RSn_{ex} \geqslant 50$%) достигается для насекомых в восьми из 30 случаев перекрестного тестирования, когда обучающая и контрольная выборки не совпадают между собой. Для растений это качество наблюдается для 18 пар геномов; для рыб — в четырех из 12 случаев. При обучении на геноме двух организмов — сои *Glycine max* и тополя *Populus trichocarpa* — приемлемо классифицируются гены всех пяти родственных видов.

¶ **4.4.2. Распознавание вторичной структуры белка.** Результаты работы алгоритма 2.3 на описанной в разделе 4.2 выборке белков приведены в табл. 4.15 [77]. Серьезным недостатком алгоритма при использовании модели порядка $l = 3$ является большое количество белков, для которых классификация невозможна из-за отсутствия сведений об определенных переходных вероятностях. В связи с этим, сведения о качестве распознавания вторичной структуры для контрольных выборок приведены только для той части белков, которая была успешно классифицирована.

Таблица 4.11

Качество классификации оптимальной модели $\mathcal{M}(l,1)$ на контрольной выборке для класса скрытых символов, соответствующих экзонам

Вид	l	SSp_{ex}, %	SSn_{ex}, %	CC_{ex}, %	ACP_{ex}, %	RSp_{ex}, %	RSn_{ex}, %
Anopheles gambiae	6	77,52	90,38	74,55	87,30	63,96	55,77
Apis mellifera	6	89,93	94,15	87,98	93,99	70,31	71,82
D. melanogaster	7	87,92	92,60	85,87	92,94	70,46	67,80
Drosophila yakuba	6	93,46	93,27	87,89	93,95	69,92	66,08
Nasonia vitripennis	6	93,52	92,39	86,97	93,48	71,83	69,34
Tribolium castaneum	6	86,95	91,77	82,87	91,44	57,77	48,85
Arabidopsis thaliana	6	96,42	96,90	90,91	95,45	76,35	75,99
Glycine max	7	82,66	90,81	79,74	89,88	69,24	65,35
Medicago truncatula	6	86,50	92,73	82,00	91,00	61,95	55,90
Oryza sativa	7	90,84	81,16	74,78	87,39	61,29	49,91
Populus trichocarpa	6	87,94	93,26	82,67	91,33	69,59	61,36
Vitis vinifera	6	80,28	87,38	77,42	88,72	64,42	55,37
Danio rerio	7	67,62	91,07	75,32	87,96	55,51	61,03
Oreochromis niloticus	7	85,37	86,92	83,14	91,57	68,32	65,81
Oryzias latipes	6	81,03	88,64	78,35	89,19	57,13	51,56
Takifugu rubripes	7	83,94	84,79	78,69	89,34	58,17	52,15
Homo sapiens	7	35,58	89,56	49,48	76,45	27,64	31,11
Gallus gallus	7	54,41	64,65	56,06	78,12	47,39	32,03
Sus scrofa	7	33,24	85,66	47,60	75,75	24,87	26,64
Mus musculus	7	59,97	85,32	67,20	83,97	42,22	40,39
Rattus norvegicus	7	61,73	83,59	67,47	84,01	40,75	36,49
Papio anubis	7	39,91	86,65	52,28	77,45	30,41	31,42

¶ **4.4.3. Оценка отсутствующих данных.** Для устранения отказов от классификации проверим предложенные в разделе 2.6 четыре методики аппроксимации отсутствующих данных:

- композиции моделей $\mathcal{M}_c$;
- использование априорных оценок вероятностей — модель $\mathcal{M}_\varepsilon$;
- оценка вероятностей с использованием более общих моделей формулами вида (2.26) или (2.27) — модели $\mathcal{M}_f$ и $\mathcal{M}_s$ соответственно.

Согласно результатам тестирования (табл. 4.16), лучшие результаты демонстрирует метод аппроксимации, основанный на усреднении вероятностей более общих моделей: для модели $\mathcal{M}_s(2,5)$ показатели качества распознавания всего на 5% хуже, чем для рассмотренной ранее модели $\mathcal{M}(3,1)$ на классифицируемой ей части выборки.

¶ **4.4.4. Оценки вычислительной сложности.** Проверим точность верхней оценки сложности алгоритма 2.3 $O(|S|\cdot|R_h|^l)$. Для этого замерим суммарное время работы алгоритма на обучающей и соответствующей ей контрольной выборке T_f при различных значениях порядка модели $l = 1,\ldots,7$ и использовании семи вычислительных потоков (рис. 4.1).

Из полученных данных следует, что оценка времени исполнения достаточно точна — оно зависит от параметра l практически экспоненциально,

$$\frac{T_f(l)}{T_f(l-1)} \approx 1{,}7\text{–}2{,}0. \tag{4.2}$$

Таблица 4.12

Тестирование алгоритма на родственных видах — насекомые

Обучение	Контроль											
	A. gambiae 1		*A. mellifera* 2		*D. melanogaster* 3		*D. yakuba* 4		*N. vitripennis* 5		*T. castaneum* 6	
	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn
1	71,26	68,42	35,47	14,67	57,73	50,87	60,32	52,63	51,28	27,51	35,63	18,22
2	26,40	32,27	80,14	82,41	21,66	28,18	29,93	27,60	54,98	49,00	29,68	30,27
3	57,97	57,35	45,90	21,65	75,81	76,30	71,46	65,00	60,49	37,64	48,78	29,30
4	55,67	59,37	43,26	21,59	64,94	67,70	80,04	78,21	59,06	38,83	46,88	30,23
5	36,16	51,97	58,75	55,20	37,04	55,42	46,62	53,75	79,75	79,76	50,03	50,32
6	30,82	34,31	43,66	30,41	35,89	48,90	46,28	48,96	56,51	46,24	70,31	65,66

Таблица 4.13

Тестирование алгоритма на родственных видах — растения

Обучение	Контроль											
	A. thaliana 1		*Glycine max* 2		*M. truncatula* 3		*O. sativa* 4		*P. trichocarpa* 5		*V. vinifera* 6	
	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn
1	79,89	80,64	57,36	72,14	49,52	56,72	54,16	59,74	59,87	69,11	45,15	70,83
2	74,05	60,22	72,67	69,89	59,20	51,14	59,60	50,96	68,77	61,16	58,83	61,89
3	65,04	48,79	64,03	63,49	65,39	60,85	50,87	42,46	59,90	53,81	46,84	53,15
4	56,02	40,65	48,28	33,95	37,74	22,40	66,80	56,44	49,51	39,23	48,44	42,43
5	75,25	63,06	67,49	64,52	58,04	51,80	61,13	54,12	74,48	70,15	58,42	64,25
6	72,62	53,59	67,20	52,95	54,85	42,04	58,76	45,54	69,20	54,56	70,10	64,02

Таблица 4.14

Тестирование алгоритма на родственных видах — рыбы

Обучение	Контроль							
	Danio rerio 1		*O. niloticus* 2		*O. latipes* 3		*T. rubripes* 4	
	RSp	RSn	RSp	RSn	RSp	RSn	RSp	RSn
1	59,86	67,37	46,14	52,81	47,84	41,27	33,71	28,35
2	55,38	53,56	71,36	70,08	59,31	49,39	49,56	42,65
3	48,80	54,89	55,17	60,33	68,04	65,60	45,44	43,81
4	51,49	48,14	60,93	57,52	61,02	53,16	64,92	60,67

Таблица 4.15

Результаты работы модели $\mathcal{M}(l, 1)$ на выборке, состоящей из уникальных по названию белков

Выборка	SC, %	Структура	SSp, %	SSn, %	CC, %	ACP, %	RSp, %	RSn, %
			$l = 2$					
Обучение	61,71	–	70,62	53,59	40,44	70,25	16,23	7,77
		спираль	55,67	84,64	45,39	72,71	7,65	4,43
		слой	64,86	40,57	40,71	70,69	21,85	8,66
Контроль	60,35	–	69,30	52,42	38,55	69,31	14,35	6,76
		спираль	54,67	83,71	43,31	71,67	6,25	3,60
		слой	62,79	38,24	38,08	69,39	19,42	7,46
			$l = 3$					
Обучение	87,00	–	87,95	84,51	76,95	88,48	56,34	55,24
		спираль	86,34	91,13	82,15	91,08	50,77	51,51
		слой	86,45	85,03	81,49	90,75	61,69	60,76
Контроль (59,51% пропусков)	81,59	–	81,82	80,13	68,06	84,03	45,62	43,73
		спираль	81,12	85,69	73,91	86,96	39,07	39,35
		слой	81,97	78,00	73,92	86,97	51,98	49,40

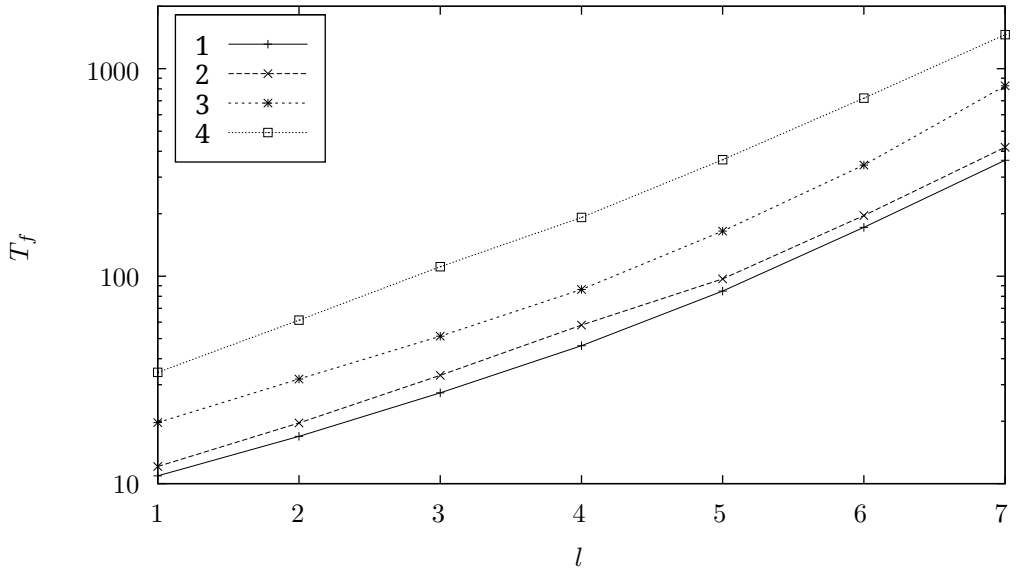


Рис. 4.1. Зависимость времени работы алгоритма 2.3 от порядка модели для геномов:

1 — *Arabidopsis thaliana*;

2 — *Oryza sativa*;

3 — *Drosophila melanogaster*;

4 — *Danio rerio*.

При увеличении l отношение (4.2) возрастает из-за увеличения времени доступа к отдельным переходным вероятностям. Справедлива и линейная зависимость времени выполнения алгоритма от суммарной длины генома. Так, поскольку длина генов организма *Danio rerio* составляет около $2{,}5 \cdot 10^7$ оснований, а генов *Drosophila melanogaster* — около $1{,}4 \cdot 10^7$ оснований, время работы алгоритма для первого из геномов при любом значении l в $\approx 1{,}8$ раз больше, чем для второго.

Аналогично, замерим время работы алгоритма 2.3 для задачи распознавания белков

Таблица 4.16

Усредненные меры качества моделей с аппроксимацией на контрольной выборке при кросс-валидации для задачи распознавания вторичной структуры белка

Модель	l	SC, %	Структура	SSp, %	SSn, %	CC, %	ACP, %	RSp, %	RSn, %
$\mathcal{M}_c(2,l)$	3	66,56	–	73,49	60,84	47,32	73,68	29,15	18,31
			спираль	60,67	83,88	51,31	75,68	20,67	14,76
			слой	69,99	49,86	49,36	74,88	36,36	20,53
	4	67,07	–	74,04	61,46	48,22	74,13	31,29	19,76
			спираль	61,03	84,17	51,93	75,99	22,61	16,19
			слой	70,80	50,53	50,27	75,34	38,57	21,89
$\mathcal{M}_\varepsilon(l,\cdot)$, $\varepsilon_\pi = 10^{-4}$, $\varepsilon_p = 10^{-3}$	3	63,19	–	68,16	60,27	41,60	70,80	17,74	22,03
			спираль	63,99	68,94	46,56	73,29	11,43	17,44
			слой	54,75	59,45	43,44	71,73	17,92	25,63
	4	72,48	–	71,99	75,51	54,69	77,35	30,22	35,42
			спираль	75,02	72,19	59,29	79,65	23,39	30,29
			слой	69,53	67,57	59,26	79,63	33,80	39,43
	5	71,67	–	63,99	86,42	52,02	76,01	35,71	34,65
			спираль	82,13	62,40	59,06	79,61	26,93	29,20
			слой	79,78	59,83	61,55	80,95	43,92	38,94
$\mathcal{M}_f(l)$	3	64,70	–	68,25	62,50	42,91	71,46	19,72	22,76
			спираль	67,47	68,89	50,12	75,06	15,97	18,71
			слой	55,60	62,10	45,46	72,74	20,84	26,14
	4	69,91	–	70,46	73,12	51,51	75,75	29,14	34,55
			спираль	75,40	65,85	55,69	77,86	25,23	29,01
			слой	62,43	70,53	55,44	77,75	31,04	38,91
	5	72,30	–	72,45	75,15	54,97	77,49	32,13	36,10
			спираль	76,84	69,90	59,47	79,74	28,24	31,21
			слой	66,08	70,96	58,56	79,29	34,61	40,25
$\mathcal{M}_s(2,l)$	3	70,19	–	74,71	65,81	51,50	75,76	29,21	22,58
			спираль	66,82	82,14	57,34	78,69	21,42	18,23
			слой	69,41	59,43	54,55	77,32	34,56	25,88
	4	75,94	–	77,16	75,35	60,04	80,02	38,87	35,75
			спираль	75,26	80,83	65,04	82,52	32,62	31,39
			слой	74,86	69,41	64,10	82,06	44,19	39,87
	5	76,47	–	77,19	76,48	60,83	80,41	39,75	36,68
			спираль	76,11	80,68	65,73	82,87	33,46	32,20
			слой	75,72	69,92	65,01	82,52	45,42	40,82

(табл. 4.17). Как и в предыдущем случае, время работы растет при увеличении l приблизительно экспоненциально, как и предсказывает теоретическая оценка. Модели $\mathcal{M}_f$ и $\mathcal{M}_s$ работают в $\sim l$ раз медленнее базовой модели, что соответствует данной в разделе 2.6 оценке их сложности.

Таблица 4.17

Зависимость времени работы алгоритма 2.3 от порядка модели и метода аппроксимации данных для белков

Модель	l				
	1	2	3	4	5
$\mathcal{M}(l,1)$	4,83	10,93	47,31	233,82	348,50
$\mathcal{M}_c(2,l)$	–	10,92	46,02	236,37	360,44
$\mathcal{M}_f(l)$	–	–	80,54	466,48	1672,98
$\mathcal{M}_s(2,l)$	–	–	83,93	478,48	1766,19

Также проверим эффективность предложенной модели организации параллельных вычислений (приложение A). При увеличении числа используемых нитей исполнения время классификации для всех рассмотренных выборок снижается по меньшей мере в четыре раза (рис. 4.2–4.3).

§ 4.5. Оценка качества композиций моделей

Для повышения качества классификации фрагментов генов для организмов животных будем использовать предложенные в разделе 3 композиции моделей на основе бинарных деревьев разбиений.

¶ **4.5.1. Предикаты на основе концентрации отдельных нуклеотидов.** Рассмотрим в качестве предикатов, образующих дерево разбиения, простейшие утверждения, касающиеся концентрации отдельных нуклеотидов:

$$I_{X,\omega} = [n(S,X) > \omega], \quad X \subset R_{nt},\ \omega \in (0,1).$$

Такой выбор позволяет избавиться от трудоемкого этапа отсеивания множеств X, описанного в разделе 3.6; в самом деле, согласно примеру из раздела 3.4, в качестве X достаточно рассмотреть семь наборов нуклеотидов. Множества возможных пороговых концентраций ω будем строить в соответствии с алгоритмом 3.2 с параметрами $r_{min} = 0{,}3$, $\varepsilon = 0{,}05$. Таким образом, количество кандидатов в оптимальный предикат для произвольной выборки составляет

$$((1 - 0{,}3 - 0{,}3)/0{,}05 + 1) \cdot 7 = 9 \cdot 7 = 63.$$

Вычислительный эксперимент показывает, что построенные алгоритмом 3.5 разбиения при значениях $5 \leqslant l \leqslant 7$ мало отличаются друг от друга, как и разбиения для обучающих выборок и для полной выборки [78; 79]. В связи с этим, в дальнейшем для разбиения будем использовать деревья, полученные на полных геномах соответствующих организмов при $l = 6$ (рис. 4.4–4.6).

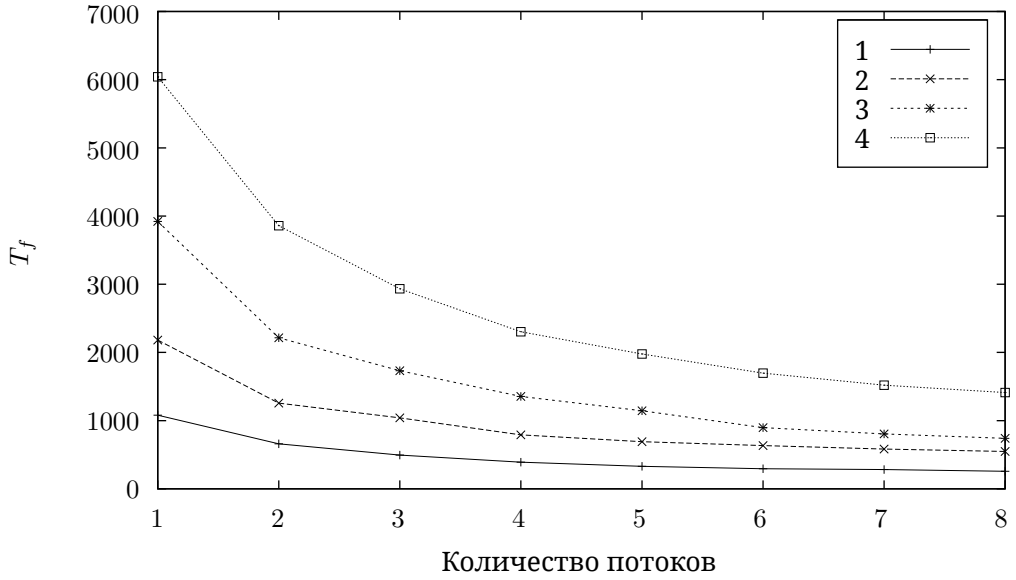


Рис. 4.2. Зависимость времени работы алгоритма 2.3 от количества нитей исполнения для геномов:

1 — *Arabidopsis thaliana*;

2 — *Oryza sativa*;

3 — *Drosophila melanogaster*;

4 — *Danio rerio*.

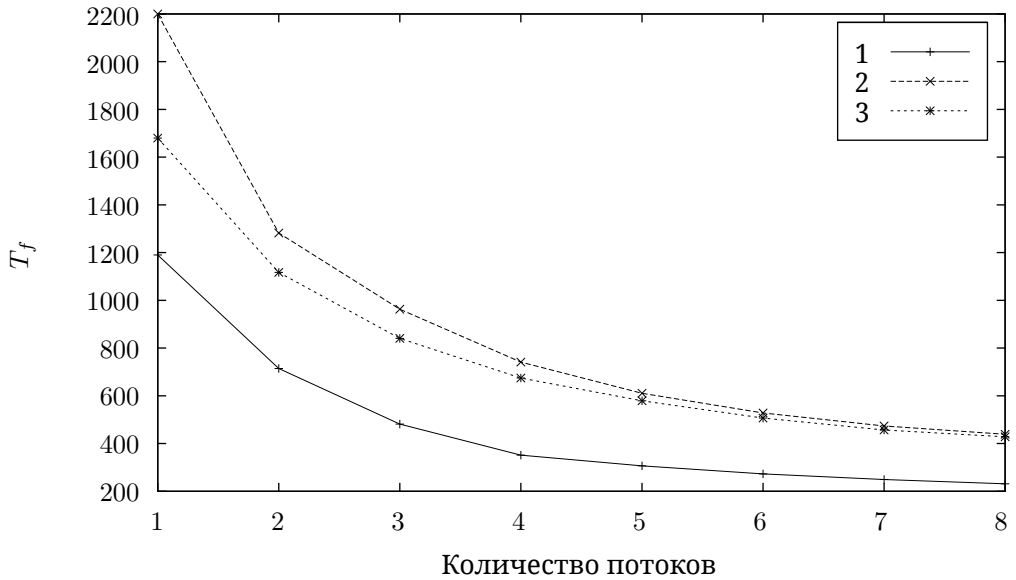


Рис. 4.3. Зависимость времени работы алгоритма 2.3 от количества нитей исполнения при классификации белков для моделей:

1 — $\mathcal{M}(4, 1)$;

2 — $\mathcal{M}_s(2, 4)$.

3 — $\mathcal{M}_f(4)$;

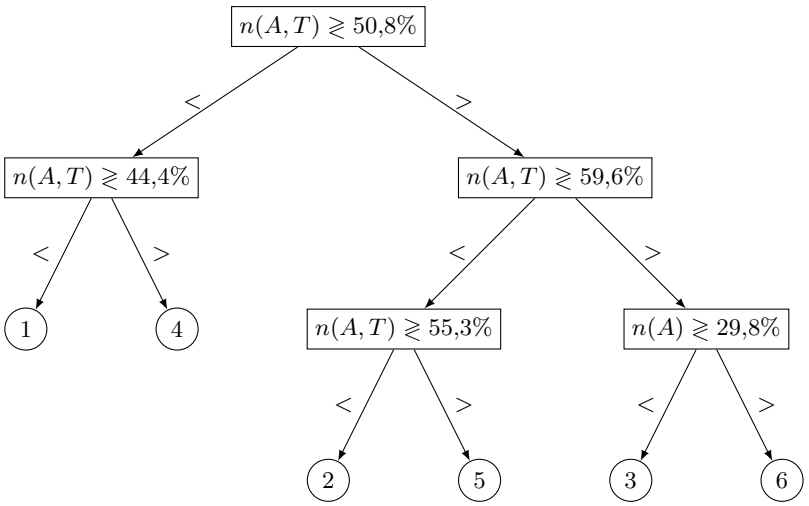


Рис. 4.4. Дерево разбиения, построенное для генов человека при $l = 6$

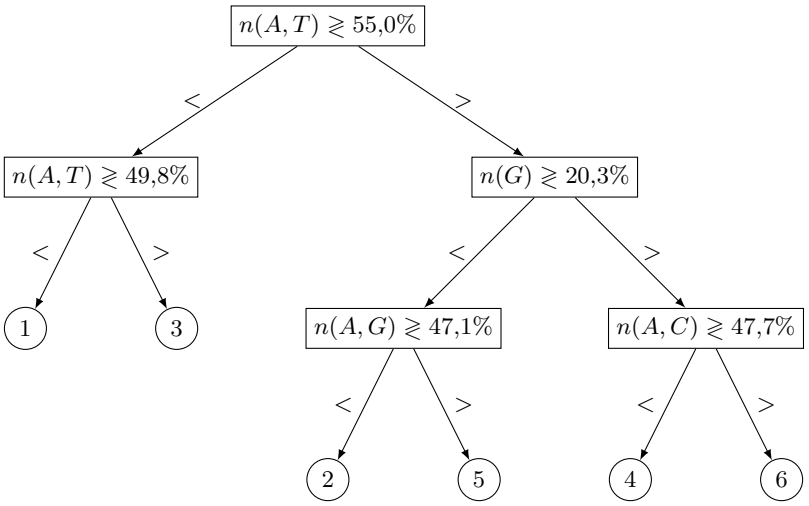


Рис. 4.5. Дерево разбиения, построенное для генов *Mus musculus* при $l = 6$

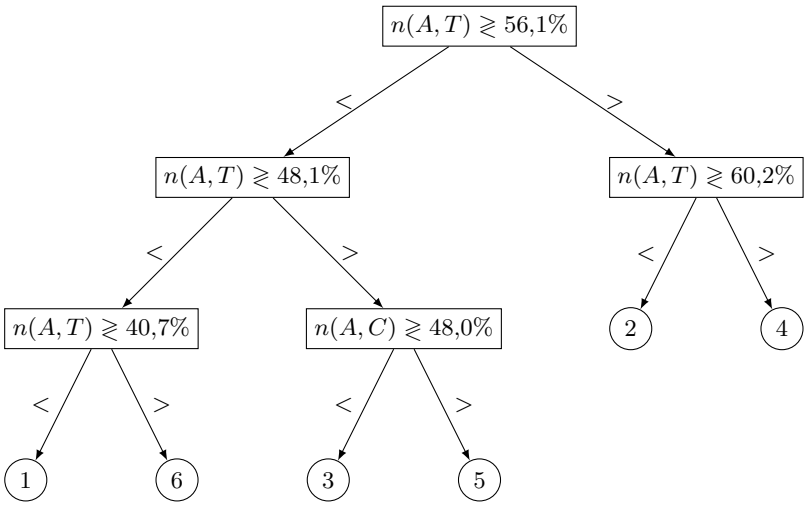


Рис. 4.6. Дерево разбиения, построенное для генов *Gallus gallus* при $l = 6$

Построенные деревья разбиений демонстрируют превалирование предикатов, использующих суммарную концентрацию нуклеотидов A и T (табл. 4.18). Важная роль этой концентрации (или эквивалентной ей концентрации $C+G$) указывается во многих работах по биохимии и биоинформатике [80; 81]. Деревья разбиения, принадлежащие близким видам (мыши и крысе, человеку и бабуину), имеют схожую структуру.

Таблица 4.18

Статистика по типам предикатов, используемых в деревьях разбиений

Вид	Число предикатов, использующих концентрацию						
	A	C	G	T	$A+C$	$A+G$	$A+T$
Homo sapiens	1	0	0	0	0	0	4
Gallus gallus	0	0	0	0	1	0	4
Sus scrofa	0	0	0	0	1	0	4
Mus musculus	0	0	1	0	1	1	2
Rattus norvegicus	0	0	1	0	1	1	2
Papio anubis	1	0	0	0	0	0	4
Всего	2	0	2	0	4	2	20

Использование композиций позволяет существенно повысить метрики качества распознавания, прежде всего нуклеотидную специфичность SSp, а также фрагментные меры RSp и RSn (табл. 4.19–4.20). Полученные данные также свидетельствуют, что при увеличении количества алгоритмов в композиции выше определенного предела она становится менее продуктивной из-за усиления эффекта переобучения — базовым алгоритмам композиции начинает соответствовать слишком мало генов из обучающей выборки. При использовании цепей Маркова седьмого порядка композиции позволяют повысить метрики качества распознавания на 10–15% (табл. 4.21).

¶ **4.5.2. Предикаты на основе концентраций пар нуклеотидов.** Для проверки работоспособности алгоритмов 3.3 и 3.4 задействуем их для отбора множеств цепочек X, состоящих из двух нуклеотидов, для шести геномов, рассмотренных в предыдущем параграфе. При этом для алгоритма 3.3 число цепочек, сохраняемых на каждой итерации, полагается равным $n_{max}=50$; для алгоритма 3.4 используются следующие значения параметров:

- размер начального поколения $n_0=20$;
- максимальный размер поколения $n_{max}=200$;
- количество мутаций для каждого набора в каждом поколении $b=2$;
- вероятность элементарной мутации $p_m=0{,}1$;
- количество скрещиваний для каждого набора в каждом поколении $c=4$;
- число поколений $M=20$.

В обоих случаях будем дополнительно отсеивать наборы цепочек в соответствии с параграфом 3.6.3; при этом минимальное расстояние Хэмминга между цепочками положим равным

Таблица 4.19

Результаты использования композиций алгоритмов для генома человека при $l = 6$

Мера	Выборка	Число алгоритмов					
		1	2	3	4	5	6
SSp_{ex}	обучение	34,39	37,51	39,14	39,02	40,57	41,70
	контроль	34,16	37,02	38,39	38,07	39,38	40,24
SSn_{ex}	обучение	88,72	91,70	92,24	92,32	92,55	92,77
	контроль	88,22	90,78	90,81	90,49	90,43	90,07
CC_{ex}	обучение	47,96	52,18	53,95	53,87	55,42	56,53
	контроль	47,55	51,36	52,70	52,27	53,49	54,15
ACP_{ex}	обучение	75,71	77,80	78,62	78,59	79,29	79,79
	контроль	75,49	77,37	77,97	77,76	78,29	78,56
RSp_{ex}	обучение	25,70	31,59	33,09	34,45	35,64	36,79
	контроль	24,91	29,92	30,58	31,26	31,92	32,35
RSn_{ex}	обучение	27,93	39,93	41,78	44,33	45,94	47,12
	контроль	26,93	37,49	38,14	39,56	40,37	40,49

Таблица 4.20

Результаты использования композиций алгоритмов для генома человека при $l = 7$

Мера	Выборка	Число алгоритмов					
		1	2	3	4	5	6
SSp_{ex}	обучение	36,46	43,24	48,76	50,75	57,93	60,64
	контроль	35,58	41,33	45,75	46,72	52,63	54,43
SSn_{ex}	обучение	91,32	94,52	95,26	95,48	95,75	96,03
	контроль	89,56	91,42	90,57	89,02	87,62	87,53
CC_{ex}	обучение	51,01	58,62	63,68	65,40	71,10	73,23
	контроль	49,48	55,67	59,17	59,38	63,49	64,71
ACP_{ex}	обучение	77,25	80,83	83,09	83,85	86,37	87,32
	контроль	76,45	79,32	80,78	80,77	82,49	83,02
RSp_{ex}	обучение	30,68	40,57	45,21	48,94	53,95	56,36
	контроль	27,64	34,81	37,29	38,87	42,27	43,84
RSn_{ex}	обучение	35,23	52,16	56,78	61,82	66,43	67,99
	контроль	31,11	43,46	44,93	46,64	48,83	49,71

Таблица 4.21

Сравнение качества работы одного алгоритма с максимальным качеством композиции алгоритмов для моделей Маркова порядка $l = 7$

Вид	Число алгоритмов	SSp, %	SSn, %	CC, %	ACP, %	RSp, %	RSn, %
Homo sapiens	1	35,58	89,56	49,48	76,45	27,64	31,11
(человек)	6	54,43	87,53	64,71	83,02	43,84	49,71
Gallus gallus	1	54,41	64,65	56,06	78,12	47,39	32,03
(курица)	3	68,29	64,14	63,81	81,92	52,90	37,70
Sus scrofa	1	33,24	85,66	47,60	75,75	24,87	26,64
(свинья)	4	47,54	81,02	58,04	79,83	36,21	40,86
Mus musculus	1	59,97	85,32	67,20	83,97	42,22	40,39
(мышь)	4	71,96	83,03	74,08	87,11	50,90	47,73
Rattus norvegicus	1	61,73	83,59	67,47	84,01	40,75	36,49
(крыса)	6	76,23	75,32	72,22	86,11	47,34	38,78
Papio anubis	1	39,91	86,65	52,28	77,45	30,41	31,42
(павиан)	6	65,00	79,75	68,56	84,42	50,54	51,18

двум.

Результаты эксперимента показывают, что генетический алгоритм находит наборы цепочек X с более высокой мерой качества, чем алгоритм 3.3 (табл. 4.22). Пары нуклеотидов входят в отобранные наборы с различной частотой: наиболее частыми являются комбинации AG, CC, GC и GG, пары AA и AT — наиболее редкими.

Таблица 4.22

Наборы пар нуклеотидов с наилучшей мерой $\mathcal{L}(T, X)$ для генома *Papio anubis* при использовании двух методик поиска

Генетический алгоритм			Наращивание размера множеств X		
Набор X	$\tilde{\omega}$[1], %	$\mathcal{L}$[2], 10^3	Набор X	$\tilde{\omega}$, %	$\mathcal{L}$, 10^3
GC,TG,AG,CC,CT,CA,GG,TA	55,37	1202,0	GC,GT,AG,CT,CC,CA,TA,GG	52,67	1203,8
GC,GT,CC,CA,GG,TA	37,17	1201,2	TG,GC,AG,CC,TC,GG,TA	46,59	1198,1
GC,GT,AG,CC,CT,CA,GG	48,72	1198,0	AC,GT,AG,CT,CC,GG	40,23	1197,7
AC,GC,GT,AG,CC,TC,TA,GG	48,81	1196,2	GT,CC,GG	19,83	1197,6
GT,AG,CC,GG	27,47	1195,8	GC,TG,AC,AG,CT,CC,TA,GG	52,97	1196,6
TG,CC,GG	22,59	1194,5	GC,GT,AG,CC,TC,GG	39,82	1196,4
AC,GC,TG,GT,AG,CC,CT,GG	54,27	1193,5	GC,CC,TA,GG	24,52	1196,1
GC,TG,AG,CC,TC,GG	42,65	1191,5	GC,CT,CA,GG	28,44	1194,8
AT,GC,CC,GG	25,77	1189,3	GC,AG,CC,CT,CA,TT,GG	50,58	1193,1
AC,GC,AG,CC,TT,GG	40,38	1188,7	TG,AG,CC,CT,CA,GG	45,33	1192,3
AC,GC,TG,CC,GG,TA	37,52	1188,1	GC,TG,GA,CC,CT,CA,GG,TA	54,33	1189,6
GC,AC,CT,GG	25,93	1187,5	GC,TC,GG	19,56	1189,4
GC,GT,AG,CT,TC,GG	40,32	1187,1	AC,TG,CC,TA,GG	31,55	1187,7
GC,CG,GT,AG,CC,CT,GG,TA	47,34	1187,1	AT,AA,TT	19,18	1171,3
CG,GT,AG,CC,CA,TC,GG,TA	47,34	1186,9	TG,AG,CC	22,86	1166,0

Примечания:

1) медианная концентрация набора;
2) мера качества набора согласно определению 3.4 для модели шестого порядка.

Определенные с помощью генетического алгоритма наборы применим для построения деревьев предикатов согласно алгоритму 3.5 (рис. 4.7, 4.8); параметры отсеивания пороговых концентраций возьмем такие же, как и для единичных нуклеотидов. Композиции алгоритмов на основе этих деревьев демонстрируют качество распознавания генов, близкое к композициям, изученным в предыдущем параграфе (табл. 4.23). Вследствие большей специфичности предикатов при увеличении количества базовых алгоритмов в композиции качество растет быстрее, чем в предыдущем случае; вместе с тем, быстрее наблюдается и эффект переобучения.

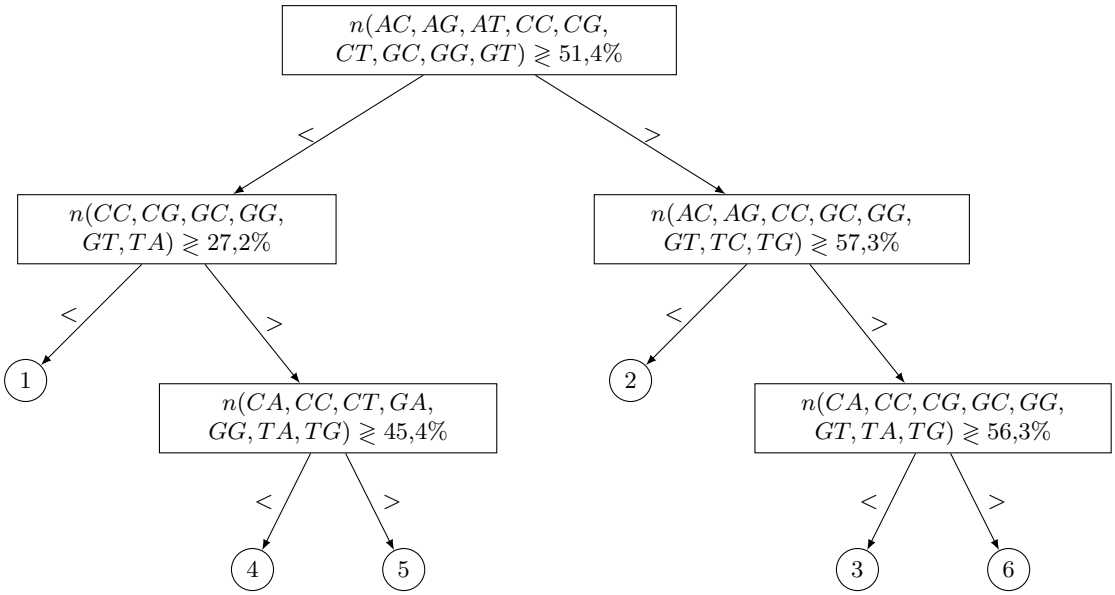


Рис. 4.7. Дерево разбиений, использующее пары нуклеотидов, для генома *Papio anubis*

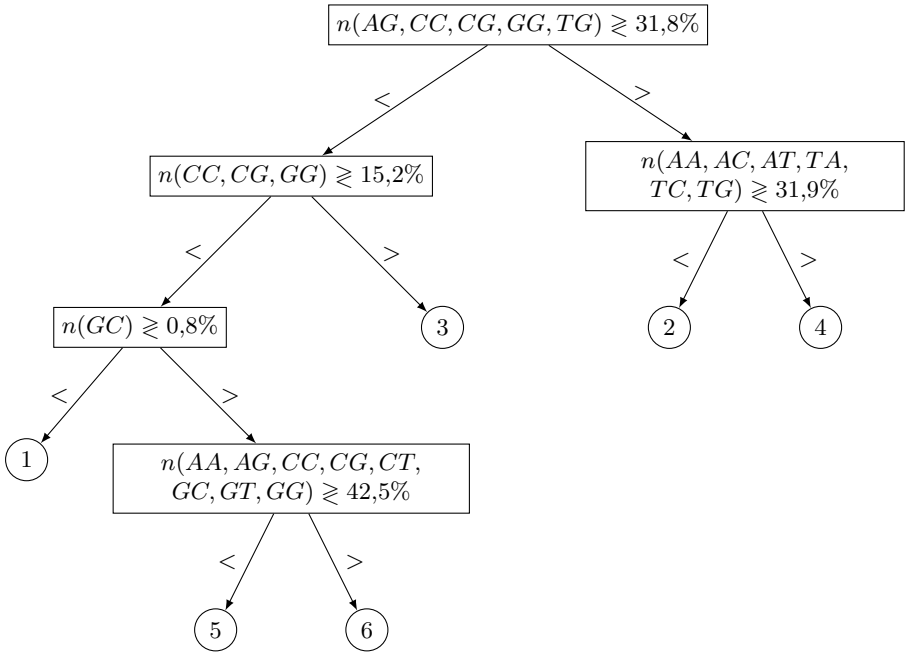


Рис. 4.8. Дерево разбиений, использующее пары нуклеотидов, для генома *Sus scrofa*

Таблица 4.23

Максимальное качество композиции алгоритмов для моделей Маркова порядка $m = 7$ при использовании концентраций пар оснований

Вид	Число алг.	SSp, %	SSn, %	CC, %	ACP, %	RSp, %	RSn, %
Homo sapiens	6	53,65	86,44	63,83	82,58	43,48	49,44
Gallus gallus	4	71,04	62,26	64,25	82,19	52,00	36,04
Sus scrofa	5	56,96	78,97	63,61	82,14	40,52	41,10
Mus musculus	5	75,17	81,25	75,12	87,58	52,33	47,61
Rattus norvegicus	3	73,57	80,24	73,40	86,72	47,33	40,67
Papio anubis	6	64,72	81,80	69,24	84,80	50,70	51,39

¶ **4.5.3. Задача определения вторичной структуры белка.** Применим аналогичный описанному в параграфе 4.5.1 подход для решения задачи 1.2. Рассмотрим предикаты на основе концентраций аминокислот для моделей четвертого порядка; как и в параграфе 4.5.2, для отсеивания наборов аминокислот будем использовать генетический алгоритм. Построенное дерево разбиения приведено на рис. 4.9. Еще одно дерево предикатов (рис. 4.10) построено с использованием ограниченного множества наборов аминокислот, полученных при помощи классификации оснований по физико-химическим характеристикам — гидрофобности и склонности к образованию спиралей [5; 82–85].

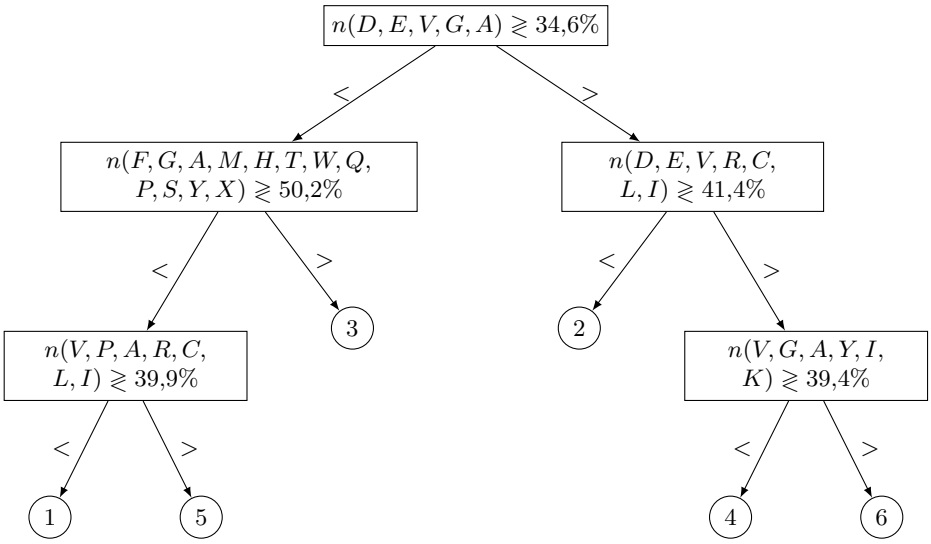


Рис. 4.9. Дерево разбиений для задачи распознавания белков при использовании генетического алгоритма отбора множеств аминокислот

Результаты вычислений (табл. 4.24, 4.25) показывают, что для рассматриваемой задачи использование композиций алгоритмов не позволяет повысить качество распознавания. Это связано с меньшим размером доступной выборки, что приводит к переобучению уже при двух алгоритмах в композиции.

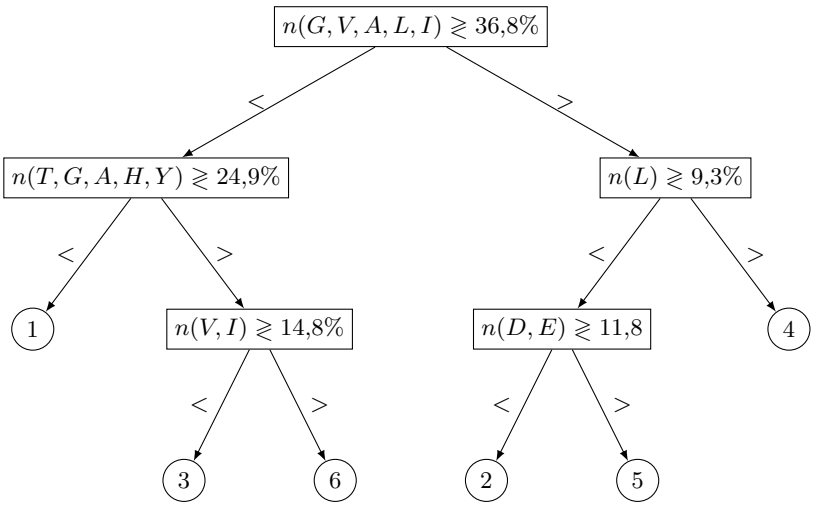


Рис. 4.10. Дерево разбиений для задачи распознавания белков при использовании априорно заданных множеств аминокислот

Таблица 4.24

Результаты использования композиций для определения вторичной структуры белков с деревом разбиения из рис. 4.9

Число алгоритмов	SC, %	Структура	SSp, %	SSn, %	CC, %	ACP, %	RSp, %	RSn, %
2	75,29	–	76,25	75,31	59,09	79,55	38,04	35,16
		спираль	74,90	79,67	63,94	81,97	31,91	30,80
		слой	74,14	68,47	63,05	81,54	43,38	39,24
3	74,92	–	75,76	75,28	58,59	79,30	37,55	34,82
		спираль	74,75	79,02	63,34	81,67	31,51	30,45
		слой	73,62	67,92	62,38	81,20	42,73	38,79
4	74,63	–	75,39	75,25	58,18	79,09	37,22	34,66
		спираль	74,51	78,53	62,83	81,42	31,20	30,31
		слой	73,35	67,47	61,91	80,97	42,38	38, 56

Таблица 4.25

Результаты использования композиций для определения вторичной структуры белков с деревом разбиения из рис. 4.10

Число алгоритмов	SC, %	Структура	SSp, %	SSn, %	CC, %	ACP, %	RSp, %	RSn, %
2	75,30	–	76,20	75,40	59,10	79,55	37,97	35,08
		спираль	74,89	79,62	63,90	81,95	31,90	30,78
		слой	74,28	68,40	63,11	81,57	43,31	39,14
3	74,89	–	75,65	75,30	58,48	79,24	37,34	34,57
		спираль	74,71	78,98	63,32	81,66	31,43	30,33
		слой	73,72	67,82	62,37	81,20	42,47	38,40
4	74,52	–	75,21	75,22	57,95	78,97	37,12	34,51
		спираль	74,33	78,44	62,66	81,33	31,15	30,28
		слой	73,53	67,26	61,89	80,96	42,29	38,36

§ 4.6. Выводы к разделу 4

Проверена симметричность в белках, синтезируемых по нитям ДНК, описанная в разделе 2.2, подтверждающая целесообразность применения для исследуемых задач моделей на основе марковских цепочек.

Приведено строгое математическое описание оценок качества алгоритмов, являющихся решениями задачи 2.2; описаны основные этапы вычислительного эксперимента по определению этих оценок для алгоритмов из разделов 2–3 в применении к задачам распознавания фрагментов генов и определения вторичной структуры белков. С помощью эксперимента определены оптимальные значения параметров моделей, позволяющие достичь качества, сравнимого с современными алгоритмами:

- длина зависимой части цепочки $m = 1$;
- порядок модели $l \in \{6, 7\}$ для задачи 1.1, $l \in \{3, 4, 5\}$ для задачи 1.2;
- приближение отсутствующих вероятностей с помощью сумм более общих моделей по формулам вида (2.27).

Подтверждены оценки вычислительной сложности для алгоритмов раздела 2, а также эффективность использования композиций (3.2) для геномов, плохо поддающихся анализу при помощи отдельных моделей. В то же время, использование композиций для определения пространственной структуры белков признано нецелесообразным.

РАЗДЕЛ 5
ВЫВОДЫ

В диссертации разработаны и теоретически обоснованы новые методы анализа последовательностей символов на основе обобщений модели Маркова со скрытыми переменными. Результаты научных изысканий применены на практике для решения двух актуальных задач биоинформатики — распознавания фрагментов генов (экзонов и интронов) и определения вторичной структуры белка. В рамках диссертационных исследований автором получены следующие результаты:

1) Разработан математический аппарат, позволяющий единообразно описать обе описанные задачи биоинформатики, а также критерии качества, применяемые в биоинформатике для оценки решающих их алгоритмов.
2) Предложена новая вероятностная модель как обобщение модели Маркова со скрытыми переменными. Разработан алгоритм динамического программирования с использованием этой модели для решения поставленной задачи. Сформулированы особенности имплементации: структуры хранения данных, ориентированные на повышение скорости работы алгоритма, а также предложена аппроксимация данных, которые невозможно получить с помощью обучающей выборки.
3) В ходе вычислительного эксперимента для задачи определения вторичной структуры белка и задачи распознавания фрагментов генов в случае достаточно простых геномов (растений, насекомых, рыб) показано, что качество классификации разработанного алгоритма находится на уровне современных алгоритмов биоинформатики.
4) Предложены композиции обобщенных моделей с эксклюзивной компетентностью составляющих. Доказано, что оценки вероятностных параметров предложенных базовых моделей композиции вычисляются так же, как оценки моделей, не ограниченных определенной областью компетентности. Описаны разбиения на области компетентности на основе бинарных деревьев предикатов, использующих концентрации коротких цепочек наблюдаемых символов; приведены алгоритмы для построения таких деревьев, а также методы отсеивания предикатов для достижения оптимальной скорости работы.
5) На основе вычислительного эксперимента сделан вывод, что алгоритмические композиции повышают качество классификации на 10–15% для геномов птиц и млекопитающих.
6) Разработан программный комплекс для решения поставленных задач распознавания скрытых последовательностей.

ЛИТЕРАТУРА

1. *Ридли М.* Геном: автобиография вида в 23 главах. — М.: Эксмо, 2008. — 432 pp.
2. *Медников Б. М.* Аксиомы биологии. — М.: Знание, 1982. — 136 pp.
3. *Anfinsen C.* The formation and stabilization of protein structure // Biochemistry Journal. — 1972. — No. 128 (4). — Pp. 737–749.
4. *Richardson J. S.* The anatomy and taxonomy of proteins // Advances in Protein Chemistry. — 1981. — No. 34. — Pp. 167–339.
5. *Pace N. C., Scholtz M. J.* A helix propensity scale based on experimental studies of peptides and proteins // Biophysical Journal. — 1998. — No. 75. — Pp. 422–427.
6. Current methods of gene prediction, their strengths and weaknesses / C. Mathé, M. F. Sagot, T. Schiex, P. Rouzé // Nucleic Acids Research. — 2002. — No. 30. — Pp. 4103–4117.
7. *Wang Z., Chen Y., Li Y.* A brief review of computational gene prediction methods // Genome Proteomics Bioinformatics. — 2004. — No. 2. — Pp. 216–221.
8. *Rabiner L., Juang B. H.* An introduction to hidden Markov models // IEEE ASSP Magazine. — 1986. — No. 3. — Pp. 4–16.
9. *Forney G. D.* The Viterbi algorithm // Proceedings of the IEEE. — 1973. — No. 61 (3). — Pp. 268–278.
10. Efficient decoding algorithms for generalized hidden Markov model gene finders / W. H. Majoros, M. Pertea, A. L. Delcher, S. L. Salzberg // BMC Bioinformatics. — 2005. — No. 6.
11. *Winters-Hilt S.* Hidden Markov model variants and their application // BMC Bioinformatics. — 2006. — No. 7 (2). — Pp. 1–14.
12. *Dempster A. P., Laird N. M., Rubin D. B.* Maximum likelihood from incomplete data via the EM algorithm // Journal of the Royal Statisitical Society. — 1977. — No. 39 (1). — Pp. 1–38.
13. Biological sequence analysis probabilistic models of proteins and nucleic acids / R. Durbin, S. Eddy, A. Krogh, G. Mitchison. — Cambridge University Press, 1998.
14. *Burge C., Karlin S.* Prediction of complete gene structures in human genomic DNA // Journal of Molecular Biology. — 1997. — No. 268 (1). — Pp. 78–94.
15. *Majoros W. H., Pertea M., Salzberg S. L.* TigrScan and GlimmerHMM: two open source *ab initio* eukaryotic gene-finders // Bioinformatics. — 2004. — No. 4. — Pp. 2878–2879.
16. *Stanke M., Waack S.* Gene prediction with a hidden Markov model and a new intron submodel // Bioinformatics. — 2003. — No. 19. — Pp. 215–225.
17. *Lim L. P., Burge C. B.* A computational analysis of sequence features involved in recognition of short introns // Proceedings of National Academy of Sciences of the USA. — 2001. — No. 98. — Pp. 11193–11198.
18. *Korf I.* Gene finding in novel genomes // BMC Bioinformatics. — 2004. — No. 5. — P. 59.
19. Integrating genomic homology into gene structure prediction / I. Korf, P. Flicek, D. Duan, M. R. Brent // Bioinformatics. — 2001. — No. 17. — Pp. 140–148.

20. *Yeh R.-F., Lim L., Burge C.* Computational inference of homologous gene structures in the human genome // Genome Research. — 2001. — No. 11. — Pp. 803–816.

21. *Knapp K., Chen Y.-P. P.* An evaluation of contemporary hidden Markov model genefinders with a predicted exon taxonomy // Nucleic Acids Research. — 2007. — No. 35. — Pp. 317–324.

22. *Rost R.* Rising accuracy of protein secondary structure prediction // Protein structure determination, analysis, and modeling for drug discovery / ed. by D. Chasman. — New York, 2003. — Pp. 207–249.

23. *Chou P. Y., Fasman G. D.* Prediction of protein conformation // Biochemistry. — 1974. — No. 13 (2). — Pp. 222–245.

24. *Chou P. Y., Fasman G. D.* Empirical predictions of protein conformation // Annual Review of Biochemistry. — 1978. — No. 47. — Pp. 251–276.

25. *Chou P. Y., Fasman G. D.* Prediction of the secondary structure of proteins from their amino acid sequence // Advances in Enzymology and Related Areas of Molecular Biology. — 1978. — No. 47. — Pp. 45–148.

26. *Kyngäs J., Valjakka J.* Unreliability of the Chou–Fasman parameters in predicting protein secondary structure // Protein Engineering. — 1998. — No. 11 (5). — Pp. 345–348.

27. *Cuff J. A., Barton G. J.* Evaluation and improvement of multiple sequence methods for protein secondary structure prediction // Proteins. — 1999. — No. 34. — Pp. 508–519.

28. *Chen H., Gu F., Huang Z.* Improved Chou–Fasman method for protein secondary structure prediction // BMC Bioinformatics. — 2006. — 7 (Suppl. 4). — S14.

29. *Garnier J., Osguthorpe D. J., Robson B.* Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins // Journal of Molecular Biology. — 1978. — No. 120. — Pp. 97–120.

30. *Garnier J., Gilbrat J. F., Robson B.* GOR method for predicting protein secondary structure from amino acid sequence. // Methods in Enzymology. — 1996. — No. 266. — Pp. 540–553.

31. *Jones D. T.* Protein secondary structure prediction based on position-specific scoring matrices // Journal of Molecular Biology. — 1999. — No. 292. — Pp. 195–202.

32. Basic local alignment search tool / S. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman // Journal of Molecular Biology. — 1990. — No. 215 (3). — Pp. 403–410.

33. Prediction of protein secondary structure and active sites using the alignment of homologous sequences / M. J. J. M. Zvelebil, G. J. Barton, W. R. Tailor, M. J. E. Sternberg // Journal of Molecular Biology. — 1987. — No. 195. — Pp. 957–961.

34. *Berner S. A., Gerloff D.* Patterns of divergence in homologous proteins as indicators of secondary and tertiary structure: a prediction of the structure of the catalytic domain of protein kinases // Advances in Enzyme Regulation. — 1990. — No. 31. — Pp. 121–181.

35. *Salamov A. A., Solovyev V. V.* Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments // Journal of Molecular Biology. — 1995. — No. 247. — Pp. 11–15.

36. *Хайкин С.* Нейронные сети: полный курс. — М.: ООО «И. Д. Вильямс», 2006. — 1104 pp.

37. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs / S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman // Nucleic Acids Research. — 1997. — No. 25 (17). — Pp. 3389–3402.

38. *Qian N.*, *Sejnowski T. J.* Predicting the secondary structure of globular proteins using neural network models // Journal of Molecular Biology. — 1988. — No. 202. — Pp. 865–884.

39. *Rost B.*, *Sander C.* Prediction of secondary structure at better than 70% accuracy // Journal of Molecular Biology. — 1993. — No. 232. — Pp. 584–599.

40. *Chandonia J.*, *Karplus M.* New methods for accurate prediction of protein secondary structure // Proteins: Structure, Function, and Genetics. — 1999. — No. 35. — Pp. 293–306.

41. *Riis S. K.*, *Krogh A.* Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments // Journal of Computational Biology. — 1996. — No. 3. — Pp. 163–183.

42. *Cortes C.*, *Vapnik V.* Support vector networks // Machine Learning. — 1995. — No. 20. — Pp. 273–293.

43. *Vapnik V.* Statistical learning theory. — New York: John Wiley and Sons, Inc., 1998. — 740 pp.

44. *Hua S.*, *Sun Z.* A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach // Journal of Molecular Biology. — 2001. — No. 308. — Pp. 397–407.

45. *Kim H.*, *Park H.* Protein secondary structure prediction based on an improved support vector machines approach // Protein Engineering Design and Selection. — 2003. — No. 8. — Pp. 553–560.

46. Secondary structure prediction with support vector machines / J. J. Ward, L. J. McGuffin, B. F. Buxton, D. T. Jones // Bioinformatics. — 2003. — No. 19 (13). — Pp. 1650–1655.

47. *Platt J. C.* Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods // Advances in large margin classifiers / ed. by A. Smola, P. Barlett, B. Schölkopf, D. Shuurmans. — Cambridge, MA: MIT Press, 1999.

48. A novel method for protein secondary structure prediction using dual-layer SVM and profiles / J. Guo, H. Chen, Z. Sun, Y. Lin // Proteins: Structure, Function, and Bioinformatics. — 2004. — No. 54. — Pp. 738–743.

49. *Белецкий Б. А.*, *Васильев С. В.*, *Гупал А. М.* Предсказание вторичной структуры белков на основе байесовских процедур распознавания // Проблемы управления и информатики. — 2007. — No. 1. — Pp. 61–69.

50. Предсказание вторичной структуры белков на основе байесовских процедур распознавания на цепях Маркова / И. В. Сергиенко, Б. А. Белецкий, С. В. Васильев, А. М. Гупал // Кибернетика и системный анализ. — 2007. — No. 2. — Pp. 59–64.

51. Процедуры распознавания вторичной структуры белков / Б. А. Белецкий, С. В. Васильев, А. А. Вагис, А. М. Гупал // Проблемы управления и информатики. — 2007. — No. 4. — Pp. 134–139.

52. *Сергиенко И. В.*, *Гупал А. М.*, *Пашко С. В.* О сложности задач распознавания образов // Кибернетика и системный анализ. — 1996. — No. 4. — Pp. 76–89.

53. *Гупал А. М.*, *Пашко С. В.*, *Сергиенко И. В.* Эффективность байесовской процедуры классификации объектов // Кибернетика и системный анализ. — 1995. — No. 4. — Pp. 70–88.

54. *Гупал А. М.*, *Пашко С. В.* Эффективность байесовского подхода к решению задач классификации и распознавания // Доп. НАН України. — 1995. — No. 10. — Pp. 81–84.

55. *Кириченко М. Ф.*, *Крак Ю. В.*, *Полищук О. А.* Псевдообратные и проекционные матрицы в задачах синтеза функциональных преобразователей // Кибернетика и системный анализ. — 2004. — No. 3. — Pp. 116–130.

56. Аналіз і синтез ситуацій в системах прийняття рішень / Ю. Г. Кривонос, М. Ф. Кириченко, Ю. В. Крак, В. С. Донченко, А. І. Куляс. — К.: Наукова думка, 2009. — 365 pp.

57. *Donchenko V.*, *Krivonos Y.*, *Krak Y.* Recurrent procedure in solving the grouping information problem in applied mathematics // International Journal “Information Models and Analysis”. — 2012. — No. 1. — Pp. 62–77.

58. *Сергиенко И. В.*, *Гупал А. М.* Статистический анализ генома // Цитология и генетика. — 2004. — No. 4. — Pp. 76–81.

59. *Гупал А. М.*, *Вагис А. А.* Комплементарность оснований в хромосомах ДНК // Проблемы управления и информатики. — 2005. — No. 5. — Pp. 153–157.

60. *Белецкий Б. А.*, *Гупал А. М.* Статистический анализ геномов бактерий. Комплементарность оснований // Проблемы управления и информатики. — 2005. — No. 6. — Pp. 135–140.

61. *Белецкий Б. А.*, *Гупал А. М.* Статистический анализ геномов растений // Доповіді національної академії наук України. — 2006. — No. 7. — Pp. 84–87.

62. *Гупал А. М.*, *Гупал Н. А.*, *Островский А. В.* Симметрия и свойства записи генетической информации в ДНК // Проблемы управления и информатики. — 2011. — No. 3. — Pp. 120–127.

63. *Журавлёв Ю. И.* Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. — 1978. — No. 33. — Pp. 5–68.

64. *Рудаков К. В.* Алгебраическая теория универсальных и локальных ограничений для алгоритмов распознавания. — 1992. — Диссертация на соискание учёной степени д.ф.-м.н., М.: ВЦ РАН.

65. *Meyer I.*, *Durbin R.* Comparative *ab initio* prediction of gene structures using pair HMMs // Bioinformatics. — 2002. — No. 18. — Pp. 1309–1318.

66. *Burset M.*, *Guigó R.* Evaluation of gene structure prediction programs // Genomics. — 1996. — No. 34. — Pp. 353–367.

67. Национальный центр биотехнологической информации США. — URL: http://ncbi.nlm.nih.gov/.

68. База данных белков Центра молекулярной и биомолекулярной информатики. — URL: ftp://ftp.cmbi.ru.nl/pub/molbio/data/dssp/.

69. *Kearns M.* A bound on the error of cross validation using the approximation and estimation rates, with consequences for the training-test split // Advances in Neural Information Processing Systems / ed. by D. S. Touretzky, M. C. Mozer, M. E. Hasselmo. — The MIT Press, 1996. — Pp. 183–189.

70. *Ng A. Y.* Preventing overfitting of cross-validation data // Proc. 14th International Conference on Machine Learning. — Morgan Kaufmann, 1997. — Pp. 245–253.

71. *Сергиенко И. В.*, *Гупал А. М.*, *Островский А. В.* Симметрия в белках, синтезируемых по нитям ДНК // Проблемы управления и информатики. — 2012. — No. 3. — Pp. 141–148.

72. *Островський О. В.* Статистичні закономірності структури білків живих організмів // Матеріали III Всеукраїнської науково-практичної заочної конференції «Наука України. Перспективи та потенціал». — Всеукраїнське громадське об'єднання «Нова освіта». Запоріжжя, 2012. — Pp. 84–86.

73. *Сергиенко И. В.*, *Гупал А. М.*, *Островский А. В.* Распознавание фрагментов генов в ДНК с применением моделей Маркова со скрытыми переменными // Кибернетика и системный анализ. — 2012. — No. 3. — Pp. 58–67.

74. *Гупал А. М.*, *Островский А. В.* Методы распознавания фрагментов генов в ДНК на основе моделей Маркова // Тезисы международной конференции «Сучасна інформатика: проблеми. досягнення та перспективи розвитку». — Киев, 2013. — Pp. 85–86.

75. *Островский А. В.* Методы распознавания фрагментов генов в геномах высших организмов: MA thesis физ.-мат. наук / Островский Алексей Викторович. — М. : Московский физикотехнический институт, 2012. — 47 pp.

76. *Островський О. В.* Визначення фрагментів генів за допомогою моделей Маркова // Матеріали XIII Всеукраїнської науково-практичної заочної конференції «Сучасна освіта і наука в Україні: традиції та інновації». — Всеукраїнське громадське об'єднання «Нова освіта». Запоріжжя, 2012. — Pp. 88–92.

77. *Островский А. В.* Определение вторичной структуры белков с помощью моделей Маркова // Проблемы управления и информатики. — 2013. — No. 2. — Pp. 140–147.

78. *Гупал А. М.*, *Островский А. В.* Использование композиций моделей Маркова для определения функциональных участков генов // Кибернетика и системный анализ. — 2013. — No. 5. — Pp. 61–68.

79. *Островський О. В.* Визначення структури генів за допомогою композицій моделей Маркова // Матеріали III Всеукраїнської науково-практичної заочної конференції «Сучасна наука: теорія і практика». — Всеукраїнське громадське об'єднання «Нова освіта». Запоріжжя, 2013. — Pp. 104–108.

80. *Sumner A. T.*, *de la Torre J.*, *Stuppia L.* The distribution of genes on chromosomes: a cytological approach // Journal of Molecular Evolution. — 1993. — No. 37 (2). — Pp. 117–122.

81. *Aïssani B.*, *Bernardi G.* CpG islands, genes and isochores in the genomes of vertebrates // Gene. — 1991. — No. 106 (2). — Pp. 185–195.

82. *Janin J.* Surface and inside volumes in globular proteins // Nature. — 1979. — No. 277. — Pp. 491–492.

83. Affinities of amino acid side chains for solvent water / R. Wolfenden, L. Andersson, P. Cullis, C. Southgate // Biochemistry. — 1981. — No. 20. — Pp. 849–855.

84. *Kyte J.*, *Doolite R.* A simple method for displaying the hydropathic character of a protein // Journal of Molecular Biology. — 1982. — No. 157. — Pp. 105–132.

85. Hydrophobicity of amino acid residues in globular proteins / G. Rose, A. Geselowitz, G. Lesser, R. Lee, M. Zehfus // Science. — 1985. — No. 229. — Pp. 834–838.

86. GenBank / D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, E. W. Sayers // Nucleic Acids Research. — 2011. — No. 39. — Pp. D32–D37.

87. *Kabsch W.*, *Sander C.* Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features // Biopolymers. — 1983. — No. 22 (12). — Pp. 2577–2637.

ПРИЛОЖЕНИЯ

Приложение А

Особенности реализации предложенных алгоритмов распознавания

Для реализации алгоритмов, описанных в разделе 2, а также их композиций, построенных в соответствии с разделом 3, был выбран язык программирования Java. Основаниями для такого выбора являются:

- строгая статическая типизация, позволяющая избежать характерных для языков программирования с динамической типизацией (Python, Perl, PHP) ошибок, возникающих вследствие несоответствия типов и неопределенности структуры объектов, с которыми происходит работа;
- парадигма объектно-ориентированного программирования, позволяющая за счет наличия классов и интерфейсов четко описать основные сущности задачи, а также при необходимости расширить функциональность полученного кода;
- отсутствие потребности в низкоуровневом программировании, в частности, организации непосредственной работы с динамической памятью; эффективный автоматический сборщик мусора;
- наличие в стандартной библиотеке языка структур, необходимых для реализации описанных алгоритмов, — хэш-таблиц, векторов переменного размера, средств, направленных на использование нескольких рабочих потоков;
- высокая скорость работы и приемлемое потребление ресурсов, в частности оперативной памяти, по сравнению с языками программирования более высокого уровня наподобие Python;
- виртуальная среда исполнения (Java Virtual Machine), позволяющая достичь переносимости между платформами (в том числе различными 32/64-битными операционными системами) без необходимости перекомпилирования или правки исходного кода приложения;
- широкая распространенность, бесплатность, наличие нескольких альтернативных реализаций (Oracle, OpenJDK, IceCoffee), в том числе под свободной лицензией GPL;
- наличие сторонних библиотек для работы с биоинформационными данными наподобие BioJava;
- существование эффективных свободно распространяемых средств разработки приложений (Eclipse, NetBeans), позволяющих автоматизировать исполнение определенных действий (напр., рефакторинг).

В соответствии с выбранным языком, для реализации описанных в разделах 2–3 моделей и композиций в комплексе используется парадигма объектно-ориентированного программирования. Основные сущности, рассмотренные в этих разделах, представлены в виде классов и интерфейсов (рис. А.1); их детальное описание приводится ниже.

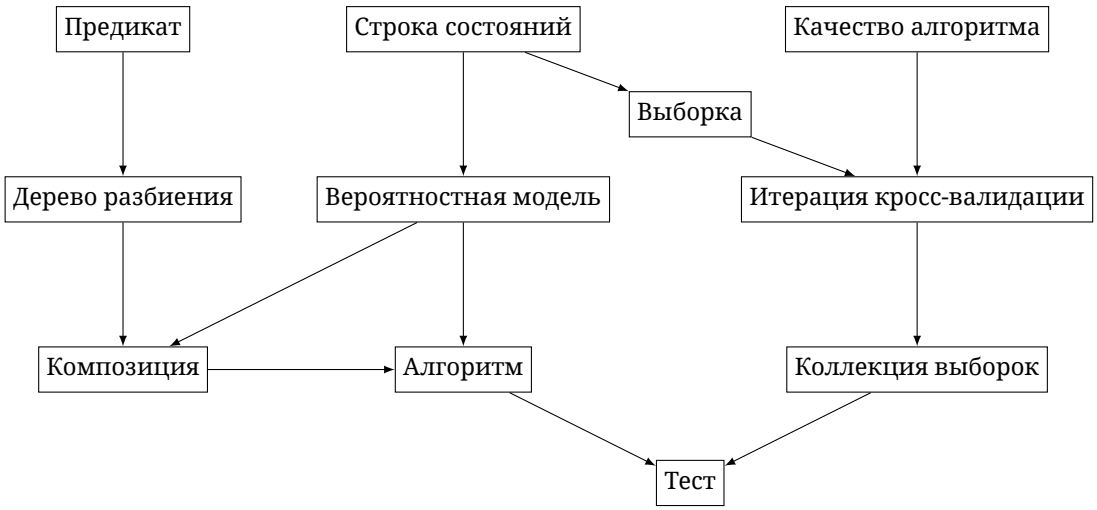


Рис. A.1. Иерархия основных понятий, используемых в вычислительном эксперименте

§ A.1. Обработка входных данных

¶ **A.1.1. Задача распознавания фрагментов генов.** Источником данных по геномам живых организмов является база Национального центра по биоинформатике США (NCBI) [67]. Инструментарий сайта NCBI позволяет просматривать и загружать описания отдельных хромосом биологических видов в стандартизированном формате GenBank, включающем в себя следующую информацию [86]:

- последовательность нуклеотидов, составляющую одну из нитей ДНК хромосомы, в направлении $5'$–$3'$, совпадающем с порядком считывания оснований при синтезе белка (раздел 1.1.1). Помимо символов A, C, G и T, соответствующих четырем нуклеотидам, описанная выше последовательность может включать символы N, отвечающие позициям ДНК, для которых установление оснований представляется на данный момент невозможным.
- маркеры различных типов для определенных локаций нитей ДНК, упорядоченные по местоположению. (Согласно формату GenBank, локация представляет собой отрезок, задающийся начальной и конечной позициями оснований нити ДНК, либо упорядоченный набор из непересекающихся отрезков.) В рамках рассматриваемой задачи интерес представляют два вида маркеров: гены и соответствующие генам кодирующие последовательности (CDS), являющиеся конкатенацией экзонов. Вследствие явления альтернативного сплайсинга, одному гену может соответствовать два или более маркера кодирующей последовательности.

Файлы GenBank предоставляют сведения, избыточные для поставленной задачи распознавания фрагментов генов; так, они содержат сведения обо всех нуклеотидах ДНК, в то время как в рамках задачи интерес представляют сравнительно небольшие ее участки. В связи с этим данные, полученные из NCBI, подвергаются обработке с целью извлечь наборы цепочек полных состояний (раздел 2.1), соответствующих всем генам данной хромосомы, а затем объединить их для всех хромосом рассматриваемого биологического вида. Для преобразования используется пакет приложений BioJava, предоставляющий, в частности, следующие возмож-

Алгоритм A.1. Преобразование файла в формате GenBank

Вход: последовательность нуклеотидов ДНК S; множество маркеров M

Выход: выборка $T \subset R_q^*$

1: $T := \varnothing$;
2: **для всех** маркеров $m \in M$
3: Определить локацию m: $l = ([a_1, b_1], [a_2, b_2], \ldots, [a_k, b_k])$.
4: **если** тип m — CDS, **то**
5: **если** $\exists i : a_1 \leqslant i \leqslant b_k \quad S_i = N$, **то**
6: Ген содержит неизвестные нуклеотиды, пропустить его.
7: **иначе**
8: $Q := \varepsilon$;
9: **для** $i = 1, 2, \ldots, k-1$
10: $Q := Q(S_{a_i \ldots b_i} \times ex^{b_i - a_i + 1})$; — добавить i-й экзон
11: $Q := Q(S_{b_i+1 \ldots a_{i+1}-1} \times in^{a_{i+1} - b_i - 1})$; — добавить i-й интрон
12: $Q := Q(S_{a_k \ldots b_k} \times ex^{b_k - a_k + 1})$; — добавить последний экзон
13: $T := T \cup \{Q\}$.

ности:

- загрузка в оперативную память и автоматический синтаксический разбор файлов в формате GenBank;
- определение последовательности нуклеотидов, соответствующей заданной локации ДНК;
- операции над локациями — конкатенация, разность и т. п.

Алгоритм, описывающий последовательность шагов по трансформации файлов GenBank, приведен в листинге A.1. В результат преобразования — выборку $T \subset R_q^*$ — входят альтернативные варианты генов, которые отвечают одному и тому же участку ДНК. Как следует из спецификации формата GenBank, маркер гена предваряет все маркеры соответствующих ему кодирующих последовательностей; таким образом, для того, чтобы добавлять в выборку только первую из цепочек CDS, принадлежащих каждому гену, достаточно заменить шаг 4 алгоритма на следующие инструкции:

4: **если** тип m — ген, **то**
5: $saved :=$ **false**;
6: **иначе если** тип m — CDS **и** $\neg saved$, **то**
7: $saved :=$ **true**;

¶ **A.1.2. Задача определения вторичной структуры белка.** Источником сведений о вторичной структуре белков, синтезируемых в различных живых организмах, является база данных Европейского центра по молекулярной и биомолекулярной информатике (CMBI) [68]. Каждому белку в базе соответствует описание в виде отдельного файла формата DSSP, состоящего из двух основных частей [87]:

Таблица A.1

Описание пространственной структуры белка согласно формату DSSP

Обозначение	Описание	Класс согласно задаче 1.2
G	3_{10}-спираль	α (спираль)
H	α-спираль	
I	π-спираль	
E	протяженная нить, входящая в β-лист	β (лист)
B	основание, принадлежащее изолированному β-мостику	
T	поворот, обусловленный водородной связью	o (отсутствие структуры)
S	точка сгиба	
C	прочие основания	

- информация, касающаяся молекулы в целом: название, синтезирующий организм, статистика по водородным связям между аминокислотами, и т. п.;
- последовательность аминокислотных оснований, а также сведения, соответствующие каждому из них, такие как тип вторичной структуры или пространственные углы поворота относительно предыдущей аминокислоты.

Вследствие малого размера и жестко заданной структуры DSSP-файлов, для их разбора используются стандартные средства работы с текстовыми файлами Java. Поскольку формат DSSP определяет большее число классов вторичной структуры белка, для ее соответствия поставленной в разделе 1.1.2 задаче 1.2 производится преобразование согласно табл. A.1.

¶ **A.1.3. Организация данных в приложении.** Для формализации понятия выборки из строк полных состояний используется интерфейс **SequenceSet**. Цепочки наблюдаемых и скрытых состояний в пределах класса представлены байтовыми массивами, элементы которых являются индексами (с отсчетом от нуля) соответствующих наблюдаемых или скрытых символов в упорядоченных некоторым образом алфавитах R_s и R_h. При этом предполагается, что размер алфавитов не превышает 128 (максимальное значение типа **byte** в Java, увеличенное на единицу), что справедливо для обеих рассматриваемых задач. Доступ к наблюдаемым и скрытым частям строк из выборки осуществляется с помощью методов `observed` и `hidden`, соответственно.

Интерфейс **SequenceSet** имеет две реализации, различающиеся по своему назначению:

- **GenericSequenceSet** — наиболее общая реализация интерфейса для произвольных множеств наблюдаемых и скрытых символов;
- **EstimatesSet** — версия выборки, используемая алгоритмами для временного хранения в оперативной памяти скрытых цепочек символов с целью оценки качества распознавания.

Класс **GenericSequenceSet** реализует два различных способа хранения данных в постоянной памяти: первый из них по отдельности записывает наблюдаемые и скрытые цепочки символов, что позволяет применять его для произвольных алфавитов R_s и R_h; второй использует строки полных состояний, что подходит для задач с малым размером алфавитов, напр.,

для распознавания фрагментов генов. Для дополнительного уменьшения объема потребляемой памяти используется потоковое сжатие данных согласно методу Лемпеля — Зива (LZ77) с помощью библиотеки gzip.

§ A.2. Имплементация вероятностных моделей

¶ **A.2.1. Хранение фрагментов строк полных состояний.** Структуры данных для работы с моделями $\mathcal{M}(l, m)$, рассмотренные в разделе 2.5, используют для описания коротких последовательностей полных состояний целыми числами функции $\rho, \sigma : R_q^* \to \mathbb{N}_0$. Для формализации этого описания в приложении введен класс **Fragment**, обладающий тремя публичными полями, соответствующими длине характеризуемой последовательности и значениям этих двух функций (листинг A.2). Для использования экземпляров класса в качестве ключей хэш-таблиц (**HashMap**) необходимо переопределение методов `hashCode` и `equals` так, чтобы они использовали перечисленные выше поля.

Поскольку объекты класса **Fragment** в целях экономии потребляемой памяти не хранят множеств наблюдаемых и скрытых символов, для действий над фрагментами, такими как конкатенация или выделение подстрок, в приложении применяется шаблон проектирования «фабрика» в виде отдельного класса **FragmentFactory**.

¶ **A.2.2. Хранение параметров моделей.** Функциональность, связанная с хранением начальных и переходных вероятностей для моделей $\mathcal{M}(l, m)$, в соответствии с парадигмой объектно-ориентированного проектирования выделена в отдельный класс **MarkovChain**. Статистики, касающиеся вероятностей, хранятся в экземплярах класса в виде хэш-таблиц (класс **HashMap**):

- для начальных вероятностей $\pi(x)$ используется таблица **Map<Fragment, Integer>**, в которой ключами являются фрагменты x, значениями — числа $N_{st}(T, x)$;
- для хранения переходных вероятностей $p(y \,|\, x)$ применяется таблица **Map<Fragment, int[]>**, ключами которой являются фрагменты x, значениями — массивы размера $|R_q|^m + 1$. Первые $|R_q|^m$ чисел каждого массива равняются значениям $N^{l,m}(T, xy)$ для всех строк $y \in R_q^m$, упорядоченных по индексу

$$|R_s|^m \cdot \rho(y) + \sigma(y) \in [0, |R_q|^m - 1];$$

  последнее число массива — $N^{l,m}(T, x)$, т. е. сумма остальных элементов.

Предложенная архитектура позволяет получать оценки вероятностей π и p согласно формулам (2.17)–(2.18) за время, определяющееся временем доступа к элементам хэш-таблиц, т. е. $O(1)$ в среднем; именно этой цели подчинено хранение избыточной информации о числах $N^{l,m}(T, x)$ — в их отсутствие время вычисления переходных вероятностей возросло бы в l раз.

Публичные методы класса **MarkovChain**, в соответствии с его назначением, делятся на две основные группы:

- получение вероятностей — методы `getInitialP` и `getTransP`;

Листинг A.2. Класс, соответствующий фрагментам строк полных состояний

```
1:  public final class Fragment {
2:      /** Значение функции ρ(x). */
3:      int observed;
4:      /** Значение функции σ(x). */
5:      int hidden;
6:      /** Длина фрагмента |x|. */
7:      int length;
8:
9:      /**
10:      * Создание фрагмента по заданным полям.
11:      */
12:     Fragment(int observed, int hidden, int length) {
13:         this.observed = observed;
14:         this.hidden = hidden;
15:         this.length = length;
16:     }
17:
18:     @Override
19:     public int hashCode() {
20:         final int prime = 31;
21:         int result = 1;
22:         result = prime * result + hidden;
23:         result = prime * result + length;
24:         result = prime * result + observed;
25:         return result;
26:     }
27:
28:     @Override
29:     public boolean equals(Object obj) {
30:         if (this == obj) return true;
31:         if (obj == null) return false;
32:         if (getClass() != obj.getClass()) return false;
33:
34:         Fragment other = (Fragment) obj;
35:         if (hidden != other.hidden) return false;
36:         if (length != other.length) return false;
37:         if (observed != other.observed) return false;
38:         return true;
39:     }
40: }
```

Листинг A.3. Обучение на строке полных состояний в классе **FallthruChain**

```
@Override public void digest(byte[] observed, byte[] hidden) {
    super.digest(observed, hidden);
    /* minSize — минимальный порядок используемых моделей; */
    /* tailOrder — порядок главной модели; */
    /* subchains[i] — подмодель порядка i. */
    for (int i = minSize; i < tailOrder; i++) {
        byte[] truncObs = Arrays.copyOfRange(
            observed, tailOrder - i, observed.length);
        byte[] truncHid = Arrays.copyOfRange(
            hidden, tailOrder - i, hidden.length);
        subchains[i].digest(truncObs, truncHid);
    }
}
```

- обучение: на основе пары из наблюдаемой и скрытой строк либо с использованием целой выборки — методы `digest` и `digestSet` соответственно; сброс результатов обучения (метод `reset`).

¶ **A.2.3. Аппроксимация отсутствующих данных.** Методы аппроксимации отсутствующих начальных и переходных вероятностей, описанные в параграфе 2.6.3, реализованы за счет производного от **MarkovChain** класса **FallthruChain**, переопределяющего его методы, касающиеся обучения и извлечения вероятностей. При этом параметры моделей меньшего порядка хранятся в переменных класса **MarkovChain** (листинг A.3). За счет такой архитектуры класса, приближенные значения вероятностей вычисляются с использованием методов `getInitialP` и `getTransP` для цепей из массива `subchains`.

¶ **A.2.4. Вычислительные алгоритмы.** Функциональность, непосредственно относящаяся к решению задачи 2.2, выражена абстрактным классом **SeqAlgorithm** с двумя группами методов:

- обучение — на основе отдельной строки полных состояний (метод `train`) либо их множества (`trainSet`); сброс результатов обучения (`reset`);
- распознавание скрытых состояний — для строки наблюдаемых символов (метод `run`) или для объекта класса **SequenceSet** (метод `runSet`).

Для контроля процесса распознавания в методе `runSet` используется интерфейс **WorkListener**, обладающий двумя методами:

- `seqCompleted(`i, H`)` — вызывается по завершении вычисления цепочки скрытых состояний $H \in R_h^*$ для элемента выборки с индексом i;
- `finished()` — вызывается после окончания вычислений.

Алгоритмам, рассмотренным в разделе 2, соответствуют различные классы, наследованные от **SeqAlgorithm**:

- Алгоритм 2.3 реализован классом **ViterbiAlgorithm**.

- Класс **GeneViterbiAlgorithm** является модификацией предыдущего класса для задачи распознавания фрагментов генов, заключающейся в замене шагов 12–15 алгоритма 2.3 на

  12: **если** $F_a(|S|, 0) = -\infty$, **то**
  13: **вернуть** ε — отказ от распознавания
  14: **иначе**
  15: $\rho_x := 0$;

- Класс **FallthruAlgorithm**, реализующий алгоритм 2.3 для вероятностной модели с аппроксимацией отсутствующих данных, отличается от **ViterbiAlgorithm** использованием для определения начальных и переходных вероятностей класса **FallthruChain** вместо **MarkovChain**.
- Метод определения недостающих вероятностей за счет использования голосования по старшинству (раздел 2.6.2) реализован классом **PriorityCompAlgorithm**.

Параллельные вычисления реализованы с помощью класса **ThreadedAlgorithm**, представляющего собой надстройку над произвольным однопоточным алгоритмом решения задачи 2.2. Весь параллельно исполняемый код заключен в рамках метода `run`; для его реализации используются примитивы из модуля `java.util.concurrent` (листинг A.4).

§ A.3. Имплементация композиций алгоритмов

Для имплементации понятия предиката, введенного в разделе 3.4, используется интерфейс **PartitionRule**, описывающий методы, вычисляющие его значение на отдельной наблюдаемой строке либо целой выборке. Предикаты вида $I_{X,\omega}$ реализованы в виде имплементирующего этот интерфейс класса **ContentPartitionRule**; при этом множества $X \subset R_s^u$ представлены как расширения класса **HashSet<Integer>** с дополнительными методами, используемыми в имплементациях алгоритмов 3.2–3.4. Бинарное дерево разбиения D, получаемое при помощи алгоритма 3.5, реализовано в виде отдельного класса **PartitionRuleTree**, включающего в себя код определения области разбиения для произвольной строки $S \in R_s^*$, соответствующий алгоритму 3.6.

В соответствии с приведенной в предыдущем разделе архитектурой, алгоритмические композиции с эксклюзивной компетентностью составляющих (3.2) описываются абстрактным классом **SwitchAlgorithm**, производным от **SeqAlgorithm** (листинг A.5). Его конкретизация реализована в виде класса-потомка **TreeSwitchAlgorithm**, переопределяющего абстрактный метод вычисления индекса модели по строке наблюдаемых состояний так, чтобы он использовал дерево разбиения, т. е. экземпляр класса **PartitionRuleTree**.

Согласно парадигме объектно-ориентированного проектирования, код для реализации вычислений меры качества разделения согласно алгоритму 3.1 выделен в отдельный класс **RuleEntropy**. Экземпляры этого класса используются в реализации шага 5 алгоритма 3.5. Для параллельного нахождения $\Delta(T_i, I)$ по различным предикатам $I \in \mathrm{Pred}(T_i)$ на этом шаге, как и в классе **ThreadedAlgorithm**, применяются примитивы из модуля `java.util.concurrent`.

Листинг A.4. Реализация параллельных вычислений в классе **ThreadedAlgorithm**

```
1:  public class ThreadedAlgorithm implements SeqAlgorithm {
2:      /** Количество потоков исполнения. */
3:      private final int nThreads;
4:      /** Базовый алгоритм решения. */
5:      private final SeqAlgorithm baseAlgorithm;
6:
7:      /* (код опущен) */
8:
9:      private class SequenceTask implements Callable<byte[]> {
10:         private final SequenceSet set;
11:         private final int index;
12:         private final WorkListener listener;
13:
14:         public SequenceTask(SequenceSet set, int i, WorkListener l) {
15:             this.set = set;
16:             this.index = i;
17:             this.listener = l;
18:         }
19:
20:         public byte[] call() throws Exception {
21:             byte[] result = baseAlgorithm.run(set.observed(index));
22:             listener.seqCompleted(index, result);
23:             return result;
24:         }
25:     }
26:
27:     public SequenceSet runSet(SequenceSet set, WorkListener l) {
28:         ExecutorService executor = Executors.newFixedThreadPool(nThreads);
29:         List<SequenceTask> tasks = new ArrayList<SequenceTask>();
30:         for (int i = 0; i < set.length(); i++)
31:             tasks.add(new SequenceTask(set, i, l));
32:         EstimatesSet estimates = new EstimatesSet(set);
33:         try {
34:             List<Future<byte[]>> results = executor.invokeAll(tasks);
35:             for (int i = 0; i < set.length(); i++)
36:                 estimates.put(i, results.get(i).get());
37:             l.finished();
38:             return estimates;
39:         } catch (Exception e) {
40:             throw new RuntimeException(e);
41:         } finally {
42:             executor.shutdown();
43:         }
44:     }
45: }
```

Листинг A.5. Реализация алгоритмических композиций классом **SwitchAlgorithm**

```
1:  public abstract class SwitchAlgorithm extends SeqAlgorithm {
2:
3:      private SeqAlgorithm algs[];
4:
5:      protected SwitchAlgorithm(SeqAlgorithm[] algs) {
6:          this.algs = algs.clone();
7:      }
8:
9:      /**
10:      * Возвращает индекс составляющего алгоритма, который должен распознавать
11:      * заданную последовательность.
12:      */
13:     protected abstract int index(byte[] sequence);
14:
15:     public void train(byte[] observed, byte[] hidden) {
16:         algs[index(observed)].train(observed, hidden);
17:     }
18:
19:     public void reset() {
20:         for (SeqAlgorithm alg: algs) {
21:             alg.reset();
22:         }
23:     }
24:
25:     public byte[] run(byte[] sequence) {
26:         return algs[index(sequence)].run(sequence);
27:     }
28: }
```

§ A.4. Тестирование алгоритмов распознавания

Для представления данных, касающихся метрик качества, описанных в разделе 4.1, предназначен класс **PredictionQuality**, поддерживающий три способа вычисления величин

$$\left\{(TP_h, TN_h, FP_h, FN_h, TR_h, PR_h, AR_h) \,|\, h \in R_h\right\}$$

(определения 4.1–4.9), которые необходимы для определения символьных и фрагментных метрик:

- одномоментное вычисление по заданной выборке $T \subset R_q^*$ и результату работы алгоритма $\{\mathcal{A}(\mathrm{Pr}_s(Q)) \,|\, Q \in T\}$;
- потоковое вычисление, при котором эталонная выборка задана изначально, а результаты работы алгоритма на отдельных ее последовательностях добавляются позже;
- усреднение результатов по нескольким экземплярам класса с использованием инструментов рефлексии языка Java (листинг A.6).

Объекты класса **PredictionQuality** используются классом **AlgorithmRun**, связывающим качество работы алгоритма с конкретной обучающей или контрольной выборкой, соответствующей одному из запусков алгоритма при n-кратной кросс-валидации. Совокупность всех $2n$ обучающих и контрольных выборок представлена классом **CrossValidation**.

Листинг A.6. Применение рефлексии для усреднения качества распознавания

```
1:  public class PredictionQuality {
2:      /** Экземпляры класса, по которым производится усреднение. */
3:      private PredictionQuality[] samples = null;
4:
5:      public PredictionQuality(PredictionQuality... samples) {
6:          this.samples = samples;
7:      }
8:
9:      private double mean(int state) {
10:         /* Определение вызывающего метода (предпоследнего в стеке вызовов). */
11:         Throwable t = new Throwable();
12:         t.fillInStackTrace();
13:         String methodName = t.getStackTrace()[1].getMethodName();
14:
15:         try {
16:             Method method = getClass().getMethod(methodName, int.class);
17:             double sum = 0.0;
18:             int nSamples = 0;
19:             for (PredictionQuality q : samples)
20:                 sum += (Double)method.invoke(q, state);
21:             sum /= samples.length;
22:             return sum;
23:         } catch (IllegalAccessException e) {
24:         } catch (InvocationTargetException e) {
25:         } catch (NoSuchMethodException e) {
26:         }
27:
28:         return 0.0;
29:     }
30:
31:     /** Символьная специфичность по заданному классу символов. */
32:     public double symbolSpec(int state) {
33:         if (samples != null) return mean(state);
34:         /* Вычислить специфичность, исходя из ее определения. */
35:     }
36:
37:     /** Символьная чувствительность по заданному классу символов. */
38:     public double symbolSens(int state) {
39:         if (samples != null) return mean(state);
40:         /* Вычислить чувствительность, исходя из ее определения. */
41:     }
42:
43:     /* (код опущен) */
44: }
```

Для сохранения промежуточных результатов класс **AlgorithmRun** реализует описанный ранее интерфейс **WorkListener**. За счет этого модель вычислений (листинг A.7) позволяет продолжать их с произвольной точки, а также контролировать качество алгоритма в процессе тестирования.

§ A.5. Выводы к приложению A

Рассмотрены основные особенности реализации предложенных в предыдущих разделах базовых понятий. Обосновано применение для имплементации языка программирования Java, а также метода объектно-ориентированного проектирования. Особое внимание при построении модели задачи уделено вопросам параллельного исполнения кода и расширяемости. Описаны классы и интерфейсы, реализующие:

- чтение и хранение исходных данных для исследуемой задачи;
- вероятностные модели и их параметры, используемые для ее решения;
- систему тестирования, предназначенную для оценки качества и сравнения различных моделей.

Листинг A.7. Вычисление качества работы алгоритма при кросс-валидации

```
1:  SequenceSet input; /* Полная выборка. */
2:  SeqAlgorithm baseAlg; /* Базовый алгоритм распознавания. */
3:
4:  /* Число частей, на которые разбивается выборка для кросс-валидации.  */
5:  final int nFolds = 5;
6:  /* Количество потоков исполнения. */
7:  final int nThreads = 8;
8:  /* Файл для хранения качества тестирования. */
9:  final String debugFile = "debug";
10:
11: SeqAlgorithm alg = new ThreadedAlgorithm(baseAlg, nThreads);
12: CrossValidation cv = new CrossValidation(input, nFolds);
13: cv.setSaveFile(debugFile);
14:
15: for (int f = 0; f < 2 * nFolds; f++) {
16:     /* Высвобождение ресурсов, использовавшихся на предыдущей итерации. */
17:     System.gc();
18:     /* Обучающая выборка, соответствующая f-й выборке коллекции:
19:      * та же, если f mod 2 = 0, или (f − 1)-я иначе. */
20:     final SequenceSet trainingSet = cv.getSet(f - (f % 2));
21:     /* Текущий запуск алгоритма. */
22:     final AlgorithmRun run = cv.get(f);
23:
24:     alg.reset();
25:     alg.train(trainingSet);
26:     run.run(alg);
27:
28:     System.out.println("Quality:");
29:     PredictionQuality q = run.getQuality();
30:     for (int i = 0; i < input.hiddenStates().length(); i++) {
31:         /* Метрики качества для i-го класса скрытых символов. */
32:         System.out.println(input.hiddenStates().charAt(i)
33:                 + ": " + q.toString(i));
34:     }
35:     cv.save();
36: }
37:
38: /* Усредненные результаты по всем обучающим и контрольным выборкам. */
39: System.out.println(cv.meanTraining());
40: System.out.println(cv.meanControl());
```