Лекция 4

Ilya Yaroshevskiy

24 сентября

Содержание

1 Решетки

1.1 Мотивация

Замечание. Рассмотрим двоичный (n, k, d) код C с порождающей матрицей G. Пусть $D(x, y)$ – функция расстояния Хемминга. Декодирвание векотора y по максимуму правдоподобия

$$\hat{c} = \text{argmin}_{c \in C} D(c, y) = \text{argmin}_{c \in C} \sum_{i=0}^{n-1} D(c_i, y_i)$$

Существует много кодовых слов содержащих одинаковые префиксы $(c_0, \ldots, c_a)$. Было бы неразумно пересчитывать $\sum_{i=0}^{a} D(c_i, y_i)$ для них каждый раз заново. Декодирование – задача поиска ближайшего кодового слова. Попытаемся сформулировать ее как задачу поиска кратчайшего пути на графе

1.2 Решетка

Определение. Решеткой (trellis) называется граф, обладающий следующими свойствами

- Вершины разбиты на непересекающиеся подмножества (уровни или ярусы)
- Нулевой и последний ярусы содержат по 1 узду (терминальные узлы)
- Грав направленный. Допускается движение только от уровня с меньшим номером к уровню с болшим нжомером. Стрелки при этом, как правило, не рисуют
- Ребрам графа приписаны метки, соответсвующие символам кодовых слов, а также метрики, называемые также весами или длинами. Длина пути равна сумме длин входящих в него ребер. Пример метрики ребра на ярусе i, помеченного c: $D(c, y_i)$

Замечание. Сопоставим пути в решетке между терминальными узлами кодовым словам. Тогда задача поиска кодового слова, минимизирующего некоторую аддитивную функцию функцию длины эквивалентна задаче поиска кратчайшего пути между терминальными узлами решетки

1.3 Алгоритм Витерби

Program 1 Viterbi(y)

```
1: M_{0,0} = 0
2: for i = 1, ..., n do
3:    for v ∈ V_i do
4:       M_{v',v} = M_{i-1,v'} + D(c[i, v', v], y_{i-1}) {Для каждого входящего ребра (v', v) вычислить метрику его пути}
5:       M_{i,v} = min M_{v',v} {Метрика узла}
6:       B_{i,v} = argmin M_{v',v} {Узел-предшественник}
7:    end for
8: end for
9: v = 0
10: for i = n, ..., 1 do
11:    ĉ_{i-1} = c[B_{i,v}, v]
12:    v = B_{i,v}
13: end for
14: return ĉ, M_{n,0}
```

Замечание.

- $M_{i,v}$ – метрика узла v на ярусе i
- V_i – множества узлов на ярусе i
- $c[i, v', v]$ – метка ребра между узлами $v' \in V_{i-1}, v \in V_i$
- Число сложений не превосходит E (число ребер в решетке)
- Число сравнений не превосходит $E - V$, где V – число узлов

1.4 Минимальная решетка кода

Определение. Профиль сложности решетки: $(\xi_0, \ldots, \xi_n), \xi_i = |V_i|$ – число узлов на i-м ярусе

Определение. Решетка называется **минимальной**, если профиль сложности $(\xi'_0, \ldots, \xi'_n)$ любой другой решетки удовлетворяет $\xi'_i \geq \xi_i, 0 \leq i \leq n$.

Замечание. Как построить минимальную решетку

Выпишем все кодовые слова $e_m = (c_{m,0}, \ldots, c_{m,n-1})$ рассмтриваемого кода. Для некоторого i определим прошлое $c^p_m = (c_{m,0}, \ldots, c_{m,i-1})$ и будущее $c^f_m = (c_{m,i}, \ldots, c_{m,n-1})$. В любой решетке пути, входящие в некоторый узел, имеют общее будущее, а пути, исходящие из одного узла имеют общее прошлое.

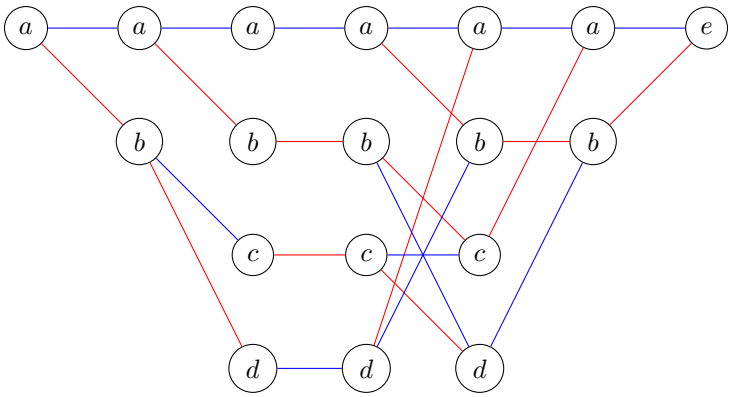
$F_i = \{c^f | \exists c^p : (c^p, c^f) \in C\}$ может быть единственным образом разбито на подмножество $F_i(c^p) = \{c^f | (c^p, c^f) \in C\}$. Сопоставим каждому такому подмножеству узлы на ярусе i. Узел $v \in V_i$ свяжем с узлом $v' \in V_{i+1}$, если для некоторого кодового слова прошлое, соответствующее v' является продолжением на 1 символ одной из последовательностей, ведущих в узле v. Пометим этим символом ребро (v, v').

Пример. Порождающая матрица

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Кодовые слова: $c_0 = (000000), c_1 = (110100), c_2 = (101010), c_3 = (011110), c_4 = (101101), c_5 = (011001), c_6 = (000111), c_7 = (110011)$

i	c^p	$F_i(c^p)$	v_i	v_{i-1}	$c_{i,v_i,v_{i-1}}$
0	$\varnothing$	$\{c_m \mid 0 \le m \le 7\}$	a	–	–
1	0	$00000, 11110, 11001, 00111$	a	a	0
	1	$10100, 01010, 01101, 10011$	b	a	1
2	00	$0000, 0111$	a	a	0
	01	$1110, 1001$	b	a	1
	10	$1010, 1101$	c	b	0
	11	$0100, 0011$	d	b	1
3	000	$000, 111$	a	a	0
	011	$110, 001$	b	b	1
	101	$010, 101$	c	c	1
	110	$100, 011$	d	d	0
4	$0000, 1101$	00	a	a, d	$0, 1$
	$0001, 1100$	11	b	a, d	$1, 0$
	$0111, 1010$	10	c	b, c	$1, 0$
	$0110, 1011$	01	d	a, c	$0, 1$
5	$00000, 11010, 10101, 01111$	0	a	a, c	$0, 1$
	$10110, 01100, 00011, 11001$	1	b	b, d	$1, 0$
6	$\{c_m \mid 0 \le m \le 7\}$	$\varnothing$	a	a, b	$0, 1$



синие ребра – 0, красные – 1

Утверждение. *Полученная решетка T минимальна*

Доказательство. Рассмотрим произвольную решетку T' этого кода. В T' два слова $c_1 = (c_1^p, c_1^f), c_2 = (c_2^p, c_2^f)$ могут иметь общую вершину на ярусе i только если $F_i(c_1^p) = F_i(c_2^p))., , T'$, проходят также через общий узел в T. Обратное не верно. Следовательно, число узлов на ярусе i в T' не меньше числа узлов на этом же ярусе в T. □

Теорема 1.1. Всякий код имеет минимальную решетку, все минимальные решетки совпадают с точностью до нумерации узлов каждого яруса

1.5 Минимальная спэновая форма матрицы

Замечание. Для (n, k) линейного кода C для каждого $i, 0 \le i \le n$ прошлое и будущее также являются линейными кодами. Для простроения решетки удобно найти базисы (порождающие матрицы) этих кодов. Удобно сделать это сразу для всех i. Началом $b(x)$ вектора $(x_0, \ldots, x_{n-1})$ будем называть номер первого его ненулевого элемента. Концом $e(x)$ вектора $(x_0, \ldots, x_{n-1})$ будем называть номер последнего его ненулевого элемента. Элементы на позициях $b(x), \ldots, e(x) - 1$ будем называть активными.

Определение. Порождающая матрица называется **приведенной к минимальной спэновой форме** (МСФ), если все начала строк различны и все концы строк различны.

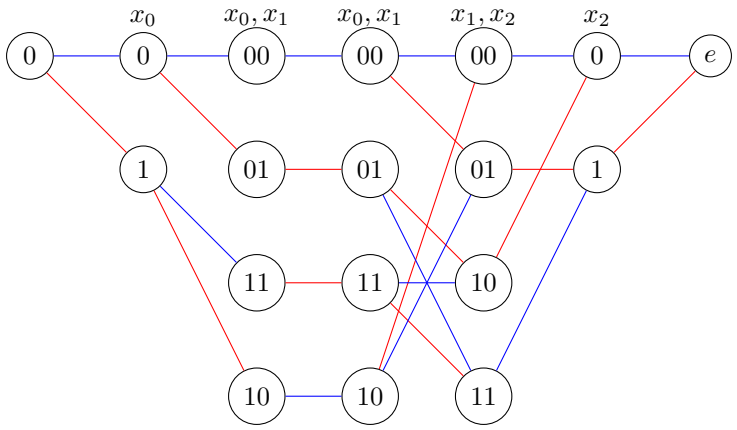
Замечание. Для определенности потребуем также, чтобы строки матрицы были упорядочены по возрастанию начал строк. Матрица может быть приведена к МСФ с помощью элементарных операций над строками

Пример. Приведем порождающую матрицу к МСФ

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \xrightarrow{(1)+(2),(1)+(3),(2)\leftrightarrow(3)} \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\xrightarrow{(2)+(1),(3)+(2)} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

На ярусе i узлы нумеруются значениями информационных символов, соответсвующих строкам МСФ, активным в позиции i. Ребра помечаются линейными комбинациями активных элементов столбца порождающей матрицы. Полученная решетка минимальна



Теорема 1.2. Решетка, получаемая по порождающей матрице в МСФ, минимальна

Доказательство. Докажем, что $\forall l$ пути, определяющиу кодовые слова с одинаковыми c^f длины $n-l$, не проходят через различные узлы на ярусе с номером l. Узел, через который проходит путь на ярусе l, определяется значениями информационных символов, соответсвующих активным на этом ярусе строкам. Эти строки ЛНЗ и заканчиваются на ярусах с номерами $> l \implies$ нетривиальные линейные комбинации этих строк отличаются хотя бы на одной позиции правее l. Предположим, что есть 2 слова с одинаковым будущим, проходящие через ранзые узлы на ярусе l. Их сумма образует слово, активное на ярусе l, равное 0 на позициях правее l. Таких слов быть не может, т.е слова с одинаковым будущим проходят через одни и те же узлы, т.е. решетка минимальна. □

1.6 Построение решетки по проверочной матрице

Замечание.

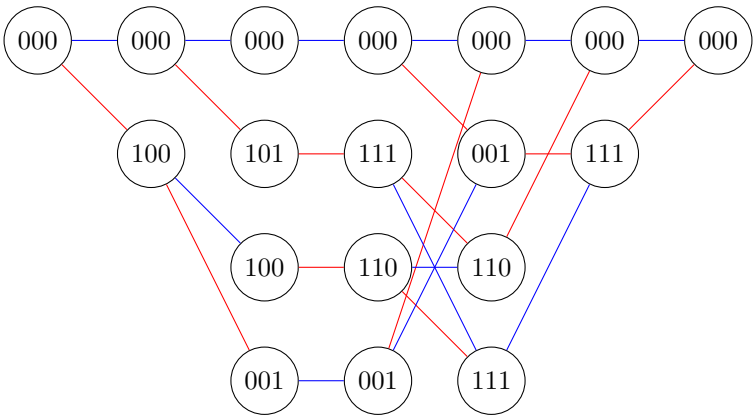
- Пусть дана проверочная матрица $H = (h_0, \dots, h_{n-1})$.
- Пусть $S_0 = 0$
- Накопленный синдром $S(x_0, \dots, x_{j-1}) = \sum_{i=0}^{j-1} h_i x_i$

Будем нумеровать узлы v в решетке накопленными синдромами $S(v)^T$. Существует ребро, помеченное c, из v' на ярусе i в v на ярусе $i+1$, если $S(v) = S(v')+ch_i$. Оставим единственный конечный узел, соответсвующий нулевому синдрому (т.е. допустимым кодовым словам). Удалим нетерминальные узлы, из которых не выходят ребра. Полученная решетка минимальна.

Замечание. Для реализации удобно (но не необходимо) привести проверочную матрицу к МСФ. Это упростит нумерацию узлов

Пример.

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$



Теорема 1.3. Решетка, построенная по проверочной матрице, минимальна

Доказательство. Докажем, что пути с одинаковыми c^f не проходят через разные узлы. Для кодового слова $c = (c^p, c^f)$ частичные синдромы, вычисленные по c^p и c^f, совпадают. Следовательно, все совпадающие c^f исходят из одного и того же узла, определяемого частичным синдромом c^p $\square$

2 Декодирование с мягким выходом

Замечание. Длинные коды можно построить, комбинируя короткие. Декодеры "составных"кодов могут быть построены из декодеров компонентных кодов. Взаимодействие декодеров может осуществляться путем обмена апостериорными вероятностями

$$p\{c_i = a|y_0^{n-1}\} = \sum_{c \in C_i(a)} p\{c|y_0^{n-1}\}$$

- $y_0^{n-1} = (y_0, \ldots, y_{n-1})$ – результат передачи кодовго слова кода C по каналу без памяти
- $C_i(a) = \{(c_0, \ldots, c_{n-1}) \in C | c_i = a\}$

Апостериорные логарифмические отношения правдоподобия для двоичных кодов

$$L_i = \ln \frac{p\{c_i = 0|y_0^{n-1}\}}{p\{c_i = 1|y_0^{n-1}\}}$$

Исходные данные ЛОПП символов кодового слова $L_i = \ln \frac{p(y_i|c_i=0)}{p(y_i|c_i=1)}$

Замечание. Такое декодирование называется декодированием с мягким входом и мягким выходом (soft input soft output, SISO)

2.1 Алгоритм Бала-Коке-Елинека-Равина

$$L_i = \ln \frac{P\{c_i = 0|y_0^{n-1}\}}{P\{c_i = 1|y_0^{n-1}\}} = \ln \frac{\sum_{(s',s)\in S_0} \frac{p(s_i=s', s_{i+1}=s, y_0^{n-1})}{p(y_0^{n-1})}}{\sum_{(s',s)\in S_1} \frac{p(s_i=s', s_{i+1}=s, y_0^{n-1})}{p(y_0^{n-1})}}$$

где S_0 и S_1 – множества пар состояний $s' \in V_i, s \in V_{i+1}$, переход между которыми помечен соответственно 0 и 1, $p(y_0^{n-1})$ – совместная плотность распределения принятых сигналов, $p(s_i = s', s_{i+1} = s, y_0^{n-1}$ – совместная плотность распределения принятых сигналов и состояний кодера на ярусах i и $i+1$

Поведение кодера при обработке i-го символа определяется только его состоянием s' на предыдущем шаге; канал не имеет памяти

$$p(s_i = s', s_{i+1} = s, y_0^{n-1}) = p(s_i = s', y_0^{i-1}) p(s_{i+1} = s, y_i | s_i = s', y_0^{i-1}) p(y_{i+1}^{n-1} | s_{i+1} = s, s_i = s', y_0^i) =$$
$$= \underbrace{p(s_i = s', y_0^{i-1})}_{\alpha_i(s')} \underbrace{p(s_{i+1} = s, y_i | s_i = s')}_{\gamma_{i+1}(s',s)} \underbrace{p(y_{i+1}^{n-1} | s_{i+1} = s)}_{\beta_{i+1}(s)}$$

Из формулы Байеса:

$$\alpha_i(s) = \sum_{\tilde{s} \in V_{i-1}} \alpha_{i-1}(\tilde{s})\gamma_i(\tilde{s}, s), s \in V_i$$

$$\beta_i(\tilde{s}) = \sum_{s \in V_{i+1}} \gamma_{i+1}(s, s)\beta_{i+1}(s), s \in V_i$$

Непосредственное вычисление этих величин приводит к значительными ошибкам округления, поэтому приходится ввести вспомогательные величины $\alpha'_i(s) = \frac{\alpha_i(s)}{p(y_0^{i-1})}$ и $\beta'_i(s) = \frac{\beta_i(s)}{p(y_i^{n-1}|y_{i-1})}$. Поделив $p(s_i = s', s_{i+1} = s, y_0^{n-1})$ на $\frac{p(y_0^{n-1})}{p(y_i)} = p(y_0^i|y_i)p(y_{i+1}^{n-1}|y_0^i) = p(y_0^{i-1})p(y_{i+1}^{n-1}|y_0^i)$, получим

$$p(s_i = s', s_{i+1} = s|y_0^{n-1})p(y_i) = \frac{\alpha_i(s')\gamma_{i+1}(s', s)\beta_{i+1}(s)}{p(y_0^{i-1})p(y_{i+1}^{n-1}|y_0^i)} = \alpha'_i(s')\gamma_{i+1}(s', s)\beta'_{i+1}(s)$$

При этом

$$L_i = \ln \frac{\sum_{(s',s)\in S_1} \alpha'_i(s')\gamma_{i+1}(s', s)\beta'_{i+1}(s)}{\sum_{(s',s)\in S_0} \alpha'_i(s')\gamma_{i+1}(s', s)\beta'_{i+1}(s)}$$

Замечание. Хотим избавиться от ошибок округления

Учитывая, что $p(y_0^{i-1}) = \sum_{x \in V_i} \alpha'_i(s)$, получим

$$\alpha'_i(s) = \frac{\alpha'_i(s)}{\sum_{s' \in V_i} \alpha'_i(s')} = \frac{\sum_{\tilde{s} \in V_{i-1}} \alpha'_{i-1}(\tilde{s})\gamma_i(\tilde{s}, s)}{\sum_{s' \in V_i} \sum_{\tilde{s} \in V_{i-1}} \alpha'_{i-1}(\tilde{s})\gamma_i(\tilde{s}, s')}$$

Начальные условия: $\alpha'_0(s) = \alpha_0(s) = \begin{cases} 1 & , s = 0 \\ 0 & , s \neq 0 \end{cases}$

$$p(y_i^{n-1}|y_0^{i-1}) = p(y_i^{n-1}|y_0^{i-1})\frac{p(y_0^{i+1})}{p(y_0^{i-1})} = p(y_{i+1}^{n-1}|y_0^{i+1})\frac{p(y_0^{i+1})}{p(y_0^{i-1})} = \frac{p(y_{i+1}^{n-1}|y_0^{i+1})}{p(y_0^{i-1})}p(y_0^{i+1}) =$$
$$= \frac{p(y_{i+1}^{n-1}|y_0^{i+1})}{p(y_0^{i-1})} \sum_{x \in V_{i+1}} \alpha_{i+1}(s) = p(y_{i+1}^{n-1}|y_0^{i+1}) \sum_{s \in V_i} \sum_{s' \in V_{i+1}} \alpha'_i(s')\gamma_{i+1}(s', s)$$

Отсюда вытекает что

$$\beta'_i(\tilde{s}) = \frac{\beta_i(s)}{p(y_i^{n-1}|y_{i-1})} = \frac{\sum_{s \in V_{i+1}} \gamma_{i+1}(\tilde{s}, s)\beta'_{i+1}(s)}{\sum_{s \in V_i} \sum_{s' \in V_{i+1}} \alpha'_i(s')\gamma_{i+1}(s', s)}$$

Начальными значениями для этой рекуррентной формулы являются

$$\beta' n(s) = \beta_n(s) = \begin{cases} 1 & , s = 0 \\ 0 & , s \neq 0 \end{cases}$$

$$\gamma_{i+1}(s', s) = p(s_{i+1} = s, y_i|s_i = s') = P\{s_{i+1} = s|s_i = s'\}p(y_i|s_i = s', s_{i+1} = s) =$$
$$= P\{c_i = \delta(s', s)\}p(y_i|c_i = \delta(s', s))$$

Вероятность $P\{c_i = \delta(s', s)\}$ представляет собой априорную вероятность того, что этот бит равен метке $\delta(s', s)$ перехода между состояниями s', s

Если рассматриваемый декодер используется как часть итеративного декодера составного кода, эта вероятность может быть найдена из апостериорных логарифмических отношений правдоподобия $L_i^{(e)}$, вычисленных другим декодером, как $P\{c_i = 1\} = \frac{\exp(L_i^{(e)})}{1+\exp(L_i^{(e)})}$, откуда следует, что

$$P\{c_i = a\} = \frac{\exp\left(\frac{L_i^{(e)}}{2}\right)}{1 + \exp(L_i^{(e)})} \exp\left((2a - 1)\frac{L_i^{(e)}}{2}\right)$$

В противном случае все символы можно считать равновероятными, что эквивалентно $L_i^{(e)} = 0$

- Нахождение логарифмических отношений правдоподобия отдельных символов кодового слова $L(c_i), i = 0 \dots n-1$
- Вычисление величин $\gamma_k(s', s) = P\{c_i = \delta(s', s)\}p(y_i|c_i = \delta(s', s))$
- Вычисление $\alpha'_i(s)$ (прямая рекурсия) согласно $\alpha'_i(s) = \frac{\sum_{\tilde{s}\in V_{i-1}} \alpha_{i-1}(\tilde{s})\gamma_i(\tilde{s},s)}{\sum_{s'\in V_i}\sum_{\tilde{s}\in V_{i-1}} \alpha_{i-1}(\tilde{s})\gamma_i(\tilde{s},s')}$, $0 < i \leq n$
- Вычисление $\beta'_i(s)$ (обратная рекурсия) согласно $\beta'_i(\tilde{s}) = \frac{\sum_{s\in V_{i+1}} \gamma_{i+1}(\tilde{s},s)\beta'_{i+1}(s)}{\sum_{s\in V_i}\sum_{s'\in V_{i+1}} \alpha'_i(s')\gamma_{i+1}(s',s)}$, $0 \leq i < n$
- Вычисление апостериорных логарифмических отношений правдоподобия $L_i, 0 \leq i < n$, информационных битов согласно $L_i = \ln \frac{\sum_{(s',s)\in S_1} \alpha'_i(s')\gamma_{i+1}(s',s),\beta'_{i+1}(s)}{\sum_{(s',s)\in S_0} \alpha'_i(s')\gamma_{i+1}(s',s)\beta'_{i+1}(s)}$
- Принятие решения относительно каждого символа

Замечание. Полученная последовательность решений может не являться кодовым словом