СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

А.Б. Левина

Теория информации и теория кодирования

Линейное кодирование

СПбГЭТУ «ЛЭТИ», 2022 г.

1 ВВЕДЕНИЕ

На данной лекции будут изучены изучены линейные коды, их применимость, математические основы, построение порождающей и проверочной матрицы, понятие синдрома, синдром как способ детектирования ошибки, коды Хэмминга, Рида-Соломона, коды с проверкой на четность циклические коды, коды Варшамова-Тенегольца.

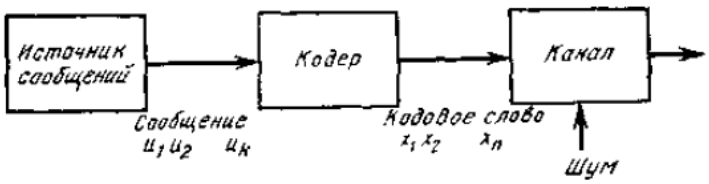
2. ЛИНЕЙНОЕ КОДИРОВАНИЕ

Для начала давайте вспомним, что такое линейное кодирование. **Линейный код** — это важный тип блокового кода, использующийся в схемах определения и коррекции ошибок.

Линейные коды:

«+» по сравнению с другими кодами, позволяют реализовывать более эффективные алгоритмы кодирования и декодирования информации;

«-» линейность легко взламывается.



Основные понятия:

- Сообщение (k символов)
- Проверочные символы (n-k символов)
- Кодовое слово (n символов)
- Код

$$\mathbf{H}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \mathbf{H}\mathbf{x}^T = \mathbf{0}, \qquad (1.1)$$

где $((n-k)\times n)$ — матрица $\mathbf{H}$, называемая *проверочной матрицей кода*, имеет вид

$$\mathbf{H} = [\mathbf{A} \mid \mathbf{I}_{n-k}]. \qquad (1.2)$$

Здесь $\mathbf{A}$ — некоторая фиксированная $((n-k)\times k)$-матрица из 0 и 1, а

$$\mathbf{I}_{n-k} = \begin{pmatrix} 1 & & 0 \\ & 1 & \\ 0 & & \ddots \\ & & & 1 \end{pmatrix}$$

— единичная матрица размера $(n-k)\times(n-k)$. Все операции в уравнении (1.1) выполняются *по модулю 2*, т. е. $0+1=1$; $1+1=0$; $-1=+1$. Мы будем называть это *двоичной арифметикой*.

Пусть H – любая двоичная матрица, *линейный код с проверочной матрицей H* состоит из всех векторов x таких, что $Hx^T=0$.

Если есть сообщение, то как найти соответствующие кодовое слово?

Для этого используется порождающая матрица:

$$\mathbf{G} = [\mathbf{I}_k \mid -\mathbf{A}^T].$$

Параметры кода:

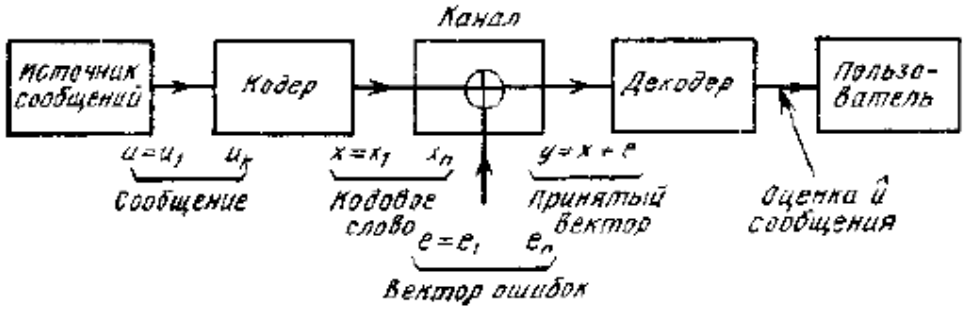
1. Длина: говорят что кодовое слова $x_1,\ldots x_n$ имеет длину n
2. Размерность: Если H имеет n-k линейно независимых строк, то имеется 2^k кодовых слов, это называется размерностью кода
3. Код называется [n, k] – кодом
4. Код использует n символов для передачи k символов то эффективностью/скоростью кода является R=k/n
5. Если x и y – кодовые слова данного кода, то x+y тоже кодовое слова этого кода, если c – любой элемент поля, то cx- тоже кодовое слово
6. Минимальное расстояние кода $d = \min\{dist(u,v)\} = \min\{wt(u-v)\}, v \in C, u \in C, v \neq u$

Определение: Линейный код длины n, размерности k с минимальными расстоянием d называется [n, k, d] -кодом.

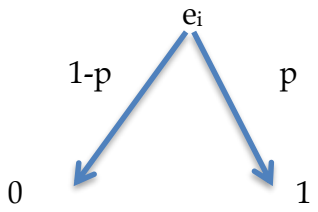
Теорема 1: Минимальное расстояние линейного кода равно минимальному весу ненулевых слов.

Теорема 2: Код с минимальным расстоянием d может исправлять наибольшее целое число от (d-1/2) ошибок.

Рассмотрим декодирование линейного кода.



Вектор ошибки: $e=y-x=e_1….e_n$



Стратегия декодера - выбор наиболее вероятного для принятого у вектора ошибки e. Что бы это обеспечить производится *Декодирование по максимуму правдоподобия*.

Prob $\{e=00000\}=(1-p)^5$

Prob $\{e=01000\}=p(1-p)^4$

Prob $\{e=10010\}=p^2(1-p)^3$

V- вектор ошибки веса a, то Prob $\{e=V\}=p^a(1-p)^{n-a}$ $p<1/2$, то $1-p>p$

Стратегия декодера – y декодируется в ближайшее кодовое слово x, т.е. выбирается вектор ошибок e с наименьшим весом. Тогда происходит *декодирование в ближайшее кодовое слово/полное декодирование*.

При декодировании возможна *ошибка декодирования* –декодер выдает неправильное слово.

Тогда происходит один из трех сценариев:

- *Полное декодирование*

  Декодер ищет ближайшее по весу кодовое слово
- *Неполное декодирование*

  Произошло не более чем l ошибок – исправляем, в противном случае нет
- *Обнаружение ошибок*

  Только проверка на наличие ошибки

3. КОДЫ ХЭММИНГА

Код Хэмминга — самоконтролирующийся и самокорректирующийся код. Построен применительно к двоичной системе счисления.

Позволяет исправлять одиночную ошибку (ошибка в одном бите слова) и находить двойную ошибку.

Назван в честь американского математика Ричарда Хэмминга, предложившего код.

Построение кодов Хэмминга основано на принципе проверки на чётность числа единичных символов: к последовательности добавляется такой элемент, чтобы число единичных символов в получившейся последовательности было чётным.

4. КОДЫ РИДА- СОЛОМОНА

Коды Рида — Соломона (*Reed-Solomon codes*) — недвоичные циклические коды, позволяющие исправлять ошибки в блоках данных. Элементами кодового вектора являются группы битов.

Используются:

- в системах восстановления данных с компакт-дисков
- при создании архивов с информацией для восстановления в случае повреждений.

Определение: Линейный код длины n называется *циклическим*, если для любого кодового слова (x_1, x_2, …., x_n) слово (x_2, x_3, …., x_n, x_1) тоже является кодовым.

Слову соответствует полином:

$$\mathbf{a(x)} = a_0+a_1x+\ldots+a_{n-1}x^{n-1}$$

Так как мы работаем в конечном поле, следовательно можно:

- умножать полиномы
- складывать полиномы

Коды Рида-Соломона, являются линейными $[n,k,d]_q$ кодами, где d=n-k+1. (частный случай кодов БЧХ - Боуза — Чоудхури — Хоквингема)

Порождающий многочлен:

- $g(x)=(x-a^b)\ (x-a^{b+1})\ldots.\ (x-a^{b+\delta-2})$, где (δ- конструктивное расстояние)

Возьмем n точек $t_1,\ldots,t_n$ из поля $\mathbf{F}_q$ и рассмотрим всевозможные многочлены из поля $\mathbf{F}_q$: $c_0+c_1x+\ldots+c_{k-1}x^{k-1}$, где c_i из поля $\mathbf{F}_q$

Рассмотрим значение многочлена в точках $t_1,\ldots,t_n$, тогда:

- Кодовое слово значение одного многочлена во всех точках
- Код C:={$(P(t_1),\ldots P(t_n))|P€\ \mathbf{F}_q[x]$ ^deg P<k}

Теорема: У кода, исправляющего *r* ошибок, кодовое расстояние *d* должно быть не менее 2r + 1.

У кодов Рида-Соломона d=n-k+1, следовательно эти коды могут исправить [(n-k)/2] ошибок

Рассмотрим вопросы декодирования.

1. Дано $P'(t_1),\ldots P'(t_n)\ €\ \mathbf{F}_q[x]$
2. Найти: P: deg P<k и $\#\{i|P(t_i)\neq P'(t_i)\}\leq(n-k)/2$

Для проведения декодирования, нам необходимо научиться считать многочлен ошибок: $E(x)=\prod_{i:\ P(ti)\neq P'(ti)}(x-t_i)$

Введем U(x)=E(x)*P(x)

- deg E – количество разрядов, где произошла ошибка

- deg U ≤ deg E+deg P

- для любого i€ {1,…,n} выполнено $U(t_i)=E(t_i)*P'(t_i)$ так как если ошибки не было, то $U(t_i)=E(t_i)*P'(t_i)=E(t_i)*P(t_i)$, а если ошибка была, то в точке t_i $E(t_i)=0$.

Тогда для решения задачи декодирования нам необходимо научиться решать следующую задачу.

Задача: Ищем многочлены, удовлетворяющие описанным условиям.

- $\deg \tilde{E} = s$ и $\mathrm{coef}_{x^s} \tilde{E} = 1$, где $s \le (n-k)/2$
- $\deg \tilde{U} \le s + k - 1$
- $\forall i \in \{1, \dots, n\}$ выполнено $\tilde{U}(t_i) = \hat{p}_i \cdot \tilde{E}(t_i)$

Зафиксируем s и положим $\tilde{E} = x^s + \sum_{j \le s-1} e_j x^j$ и $\tilde{U} = \sum_{j \le s+k-1} u_j x^j$, где $e_0, \dots, e_{s-1}, u_0, \dots, u_{s+k-1}$ — неопределённые коэффициенты.

$$\begin{cases} \hat{p}_1 t_1^s + \sum\limits_{0 \le j \le s-1} \hat{p}_1 e_j t_1^j = \sum\limits_{0 \le j \le k+s-1} u_j t_1^j \\ \vdots \\ \hat{p}_n t_n^s + \sum\limits_{0 \le j \le s-1} \hat{p}_n e_j t_n^j = \sum\limits_{0 \le j \le k+s-1} u_j t_n^j \end{cases}$$

Так как $U(x)=E(x)*P(x) \Rightarrow P(x)= U(x)/E(x)$, подставляем $(t_1,\dots t_n)$ находим кодовое слово.

Коды Рида-Соломона являются одними из самых широко применяемых линейных кодов.

5. КОДЫ ВАРШАМОВА ·ТЕНЕГОЛЬЦА

Коды Варшамова-Тенегольца способны справлять с несколькими классами ошибок:

- Ошибки выпадения
- Ошибки вставки

Рассмотрим возможные классы ошибок:

1. Ошибка замещения: му**х**а ---му**к**а
2. Ошибка стирания: муха –му?а
3. Ошибка выпадения: муха---уха
4. Ошибка вставки: мука---мурка
5. Комбинация различного вида ошибок

Введем обозначение кодового слова, согласно коду Варшамова-Тенегольца:

$$C:= \{a_1,\dots a_n | \Sigma^{n}_{i=1}\ ia_i \equiv 0(\mathrm{mod}(n+1))\}$$

Пример: n=6, 000011, (5+6) ≠ 0 mod 7
n=6, 100001, (1+6) = 0 mod 7

Рассмотрим случай с ошибкой выпадения:

На входе: ⟶ **a**=$a_1,\dots a_n$

На выходе: ⟶ **a'**=$a'_1,\dots a'_n=a_1,\dots,a_{k---1},a_{k+1},a_n$

Необходимо восстановить a_k и **a** по **a'**

Пример: **a**=010001, a_k=0, k=4, **a'**=01001 / 010001

Восстановить **a** ≠ (a_k, k)

Пусть n_0= # {i>k| a_i=0} n_1= # {i>k| a_i=1}

Если a_k=0, то **a** можно восстановить по **a'** если известно n_1

Пример: n_1=3 ….010101

Если a_k=1, то **a** можно восстановить по **a'** если известно n_0

Пример: n_0=4 1000101

Рассмотрим суммы

$$S := \sum_{i=1}^{n} ia_i \quad \text{и} \quad S' := \sum_{i=1}^{n-1} ia'_i$$

Заметим, что

$$S - S' = \sum_{i=1}^{n} ia_i - \left(\sum_{i=1}^{k-1} ia_i + \sum_{i=k}^{n-1} ia_{i+1} \right) = \sum_{i=k}^{n} ia_i - \sum_{i=k+1}^{n} (i-1)a_i$$

$$= ka_k + \sum_{i=k+1}^{n} a_i$$

Получаем

$$S' = S - \left(ka_k + \sum_{i=k+1}^{n} a_i \right) = S - ka_k - n_1$$

Так как $S \equiv 0 \pmod{(n+1)}$, то

$$S' \equiv -n_1 - ka_k \pmod{(n+1)}$$

Если $a_k = 0$, то $-S' \equiv n_1$.

Если $a_k = 1$, то

$$-S' \equiv n_1 + k = (n - k - n_0) + k = n - n_0$$

1. a_k=0, то (···S') (mod (n+1)) = n_1;

2. a_k=1, то (···S') (mod (n+1)) =n---n_0;

??? a_k

Заметим, что $\|\boldsymbol{a}'\| \geq n_1$ и $\|\boldsymbol{a}'\| \leq (n-1) - n_0$.

Отсюда $n_1 \leq \|\boldsymbol{a}'\| < n - n_0$.

1. Вычисляем (···S') (mod (n+1))

2. Сравниваем (···S') (mod (n+1)) с ||**a'**||

3. if ||**a'**|| ≥ (---S') (mod (n+1)), то выпал 0, а если ||**a'**|| ≤ n---(---S') (mod (n+1)) то выпала 1.

Получаем

$$S' = S - \left(ka_k + \sum_{i=k+1}^{n} a_i \right) = S - ka_k - n_1$$

Так как $S \equiv 0 \pmod{(n+1)}$, то

$$S' \equiv -n_1 - ka_k \pmod{(n+1)}$$

Если $a_k = 0$, то $-S' \equiv n_1$.

Если $a_k = 1$, то

$$-S' \equiv n_1 + k = (n - k - n_0) + k = n - n_0$$

Рассмотрим случай ошибка вставки

1. Ошибка выпадения --- канал выбрасывает элемент на k-ой позиции;
2. Ошибка вставки --- канал вставляет элемент и это значит выбор **k**-ой позиции для вставки и выбор элемента (0 или 1).

На входе: **a**=$a_1,...a_n$

На выходе: **a'**=$a'_1,...a'_n=a_1,...,a_k,$**x**$,a_{k+1},a_n$

Если k=0, то **a'**=x**a**, если k=n+1, то **a'**=**ax**

На входе: **a**=$a_1,...a_n$

На выходе: **a'**=$a'_1,...a'_n=a_1,...,a_k,$**x**$,a_{k+1},a_n$

Тогда

$$S' = S + (k+1)x + \sum_{i>k} a_i$$

S' (mod (n+1)) =0+(k+1) x (mod(n+1)) + Σa_i (mod(n+1)) =(k+1)x+Σa_i=(k+1)x+n_1

T:=S' (mod(n+1))

1. T=0
2. T=||**a'**||
3. 0<T≠||**a'**||

Если T=0, то (k+1)x+n_1=0, либо (k+1)x+n_1=(n+1)z, где z=1,...∞

- $(k+1)x+n_1=0$, $k+1>0$,то $x=0$, $n_1=0$, следовательно $a_{k+1}=\ldots=a_n=0$
- $(k+1)x+n_1=n+1$, $x\neq 0$ и n_1 максимально возможный, следовательно $x=1$, $a_{k+1}=\ldots=a_n=1$

Удаляем из **a'** последний символ, получаем **a.**

Если $T=||\mathbf{a'}||>0$, $S'=(k+1)x+n_1$

- $x=0$, тогда $||\mathbf{a'}||=n_1$, $a_1=\ldots=a_k=0$
- $x=1$, тогда $||\mathbf{a'}||=(k+1+n_1)$ следовательно $a_1=\ldots=a_k=1$

То есть **a** можно получить из **a'** удалив первый символ.

$$\text{Если } 0<T\neq||\mathbf{a'}||,\ T:=(k+1)x+n_1$$

- $x=0$, тогда $T=n_1$, $T\neq||\mathbf{a'}||$, следовательно $n_1<||\mathbf{a'}||$
- $x=1$, тогда $T=k+1+n_1>||\mathbf{a'}||$, $T=k+1+(n\cdots k\cdots n_0)=n+1\cdots n_0$

Тогда алгоритм можно сформулировать следующим образом.

- Вычисляем $T:=S'\ (\mathrm{mod}(n+1))$
- Вычисляем $||a'||$
- Проводим сравнение T и $||a'||$

Если $T<||a'||$, следовательно n_1 известно, выбрасываем 0 после n_1 единиц справа, если $T>||a'||$ следовательно n_0 известно, выбрасываем 1 после n_0 нулей справа.

5. ЗАКЛЮЧЕНИЕ

На данной лекции мы познакомились с линейными кодами, рассмотрели коды Хэмминга, Рида-Соломона, коды Варшамова-Тенегольца.