Teoretične osnove računalništva 2
Teorija informacij

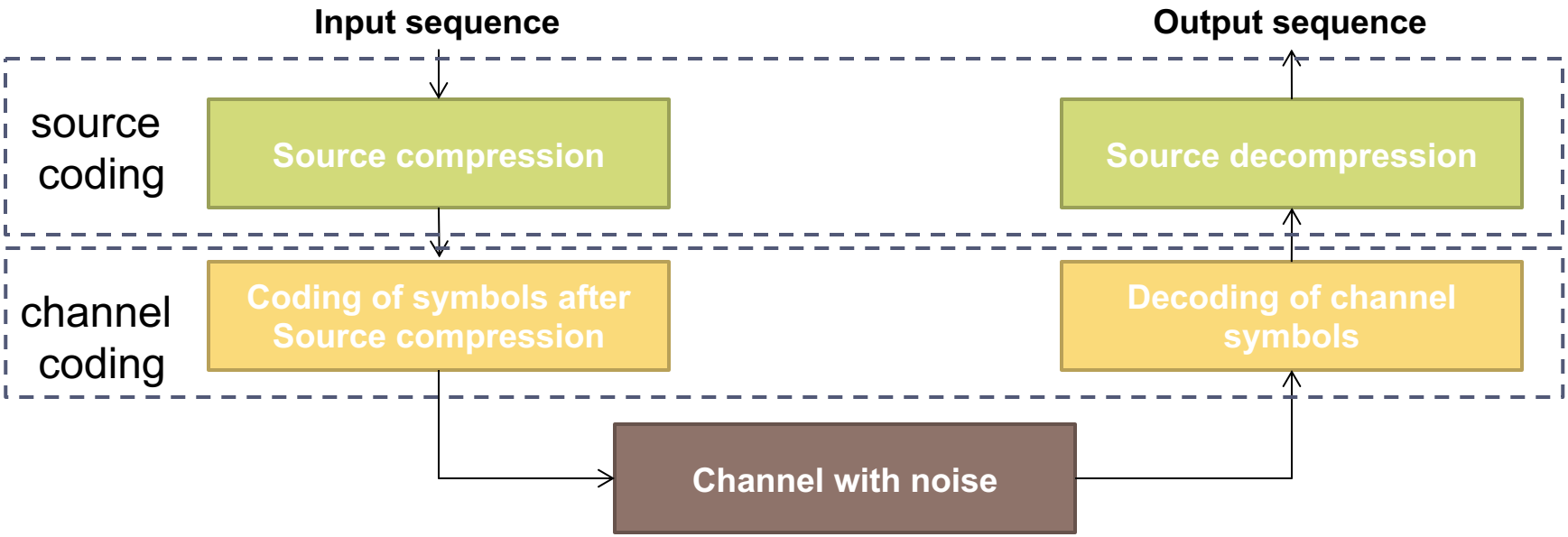
# Communication channels

Enes Pasalic UP FAMNIT študijsko leto 20/21

# Topics overview

- Channel capacity
- Example of channels:
  - Noiseless binary channel,
  - noisy typewriter,
  - binary symmetric channel
  - binary channel with  erasure
- Shannon result on channel coding
- Channel coding:
  - Hamming codes
  - Linear block codes
- Connection between source and channel coding

# Sending information over communication channels




- Source coding (compression):
  - How to encode, so that average length of symbols is small – compression

- Channel coding:
  - How to encode (protect) symbols transmitted over noisy channel so that we can decode the data that we sent without errors.

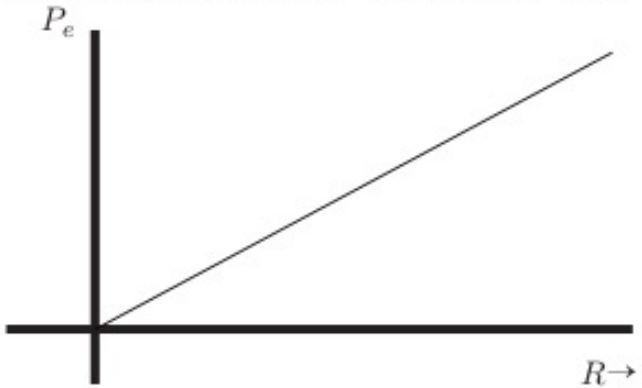
# Radio communication – back to 1930

- Place yourself back in the 1930s.
- Analog communication model of the 1930s.


- Q; Can we achieve perfect communication with an imperfect communication channel?
- Q: Is there an upper bound on the information capable of being sent under different noise conditions?
- If we increase the transmission rate over a noisy channel will the error rate increase?

# Radio communications

- Key: If we increase the transmission rate over a noisy channel will the error rate increase?
- Perhaps the only way to achieve error free communication is to have a rate of zero.
- The error profile we might expect to see is the following:



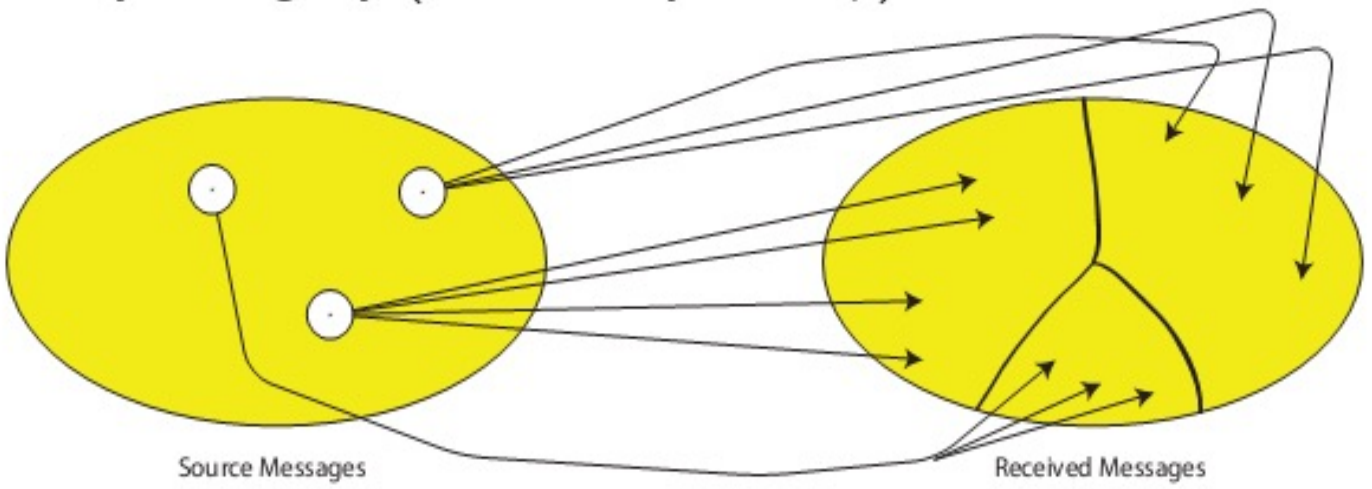

- Here, probability of error $P_e$ goes up linearly with the rate $R$, with an intercept at zero.
- This was the prevailing wisdom at the time. Shannon was critical in changing that.

# Simple example

- Consider representing a signal by a sequence of numbers.
- We now know that any signal (either inherently discrete or continuous, under the right conditions) can be perfectly represented (or at least arbitrarily well) by a sequence of discrete numbers, and they can even be binary digits.
- Now consider speaking such a sequence over a noisy AM channel.
- Very possible one number will be masked by noise.
- In such case, each number we repeat $k$ times, where $k$ is sufficiently large to ensure we can "decode" the original sequence with very small probability of error.
- Rate of this code decreases but we can communicate reliably even if the channel is very noisy.

# A key idea

- If we choose the messages carefully at the sender, then with very high probability, they will be uniquely identifiable at the receiver.
- The idea is that we choose the source messages that (tend to) not have any ambiguity (or have any overlap) at the receiver end. I.e.,



# Example of transmitting the binary data

- Assume we are sending 1000 bits per second with Pb(x=0)=Pb(x=1)=1/2.
- Channel is noisy and with prob. f = 0.1 we have 0 →1 or 1 → 0.
- How much correct information we have transmitted ?

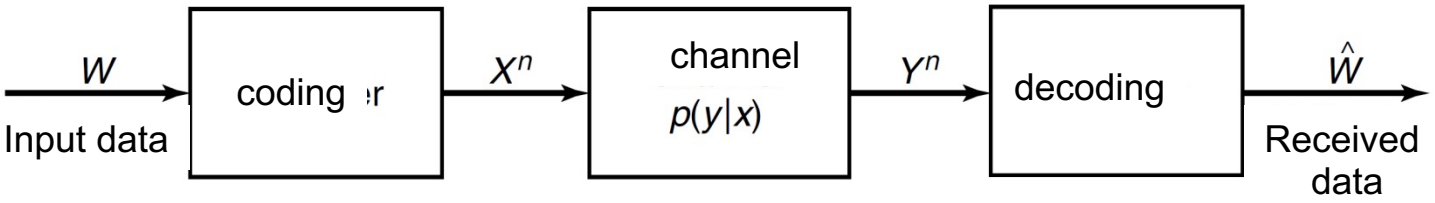
- First quess: 900 bits per second, since 100 bits are erroneous. Is this correct?
- NO, we do not know which 100 bits are erroneous.

- In extreme case when f = 0.5, we cannot transfer any information over the channel !

- **Amount of information, transmitted over channel is measured via mutual information between the input and output sequence, i.e. the difference between the source entropy and the conditional entropy when knowing the output sequence.**

- **Every channel has its CAPACITY, which is the amount of information we can transmit over the channel with arbitrary small probability of error.**

# Discrete channel

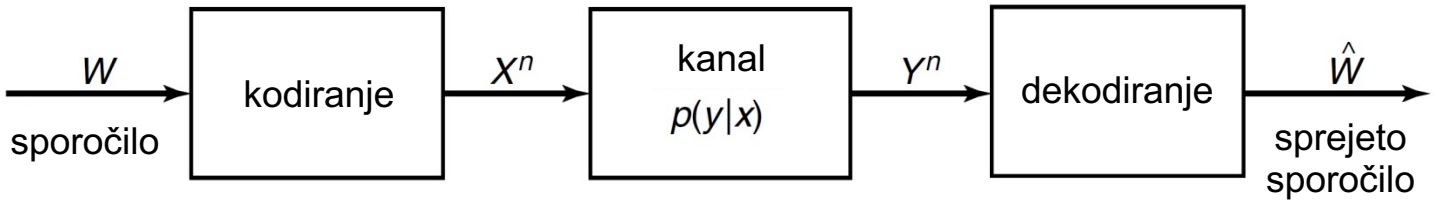



**Discrete memoryless channel (DMC)**

A discrete memoryless channel has input symbols that come form alphabet $\mathcal{X}$ and output symbols taken from alphabet $\mathcal{Y}$, and additionally the transition matrix with entries $p(y|x)$ (receiving symbol $y$ when sending $x$).

It is called memoryless when $p(y|x)$ only depends on the current values $x$ and $y$ and not on previous symbols.

# Channel capacity



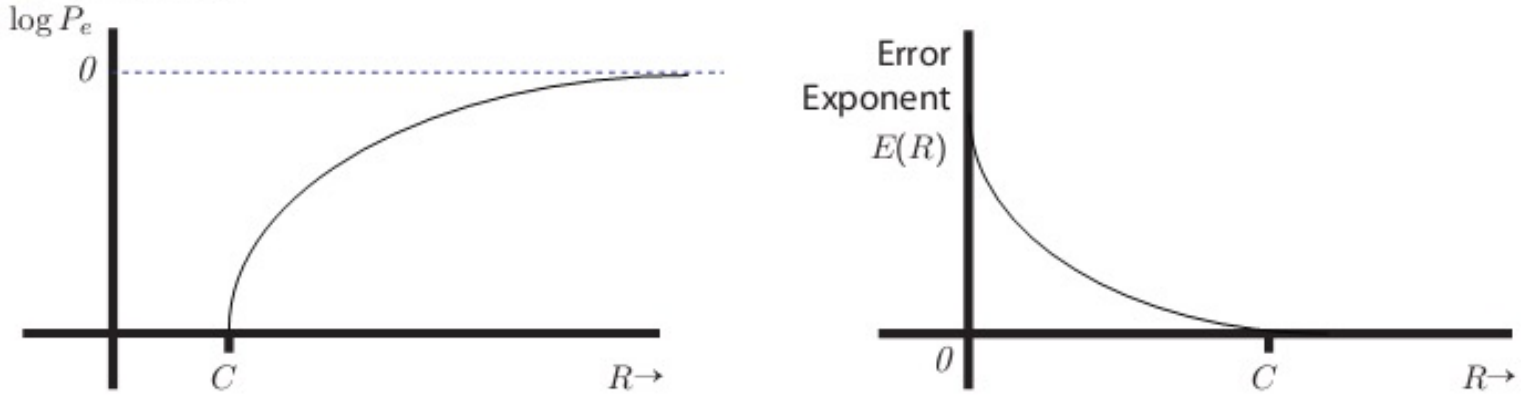

**Capacity of discrete memoryless channel**

The capacity of DMC is given by:

$$C = \max_{p(x)} I(X;Y).$$

# Reliable communication – Shannon limits

- For communication, lower bound on probability of error becomes bounded away from 0 as the rate of the code $R$ goes above a fundamental quantity $C$. Note, $P_e \propto e^{-nE(R)}$.
- That is,




- only possible way to get low error is if $R < C$. Something funny happens at the point $C$, the capacity of the channel.

# Reminder – mutual and conditional entropy

- **Vezana entropija** dveh (ali več) naključnih spremenljivk:

$$H(X,Y) = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p_{X,Y}(x,y)\,\log_2 \frac{1}{p_{X,Y}(x,y)}$$

- **Entropija pogojne porazdelitve**

$$H(X \mid Y=y) = \sum_{x\in\mathcal{X}} p_{X|Y}(x \mid y)\,\log_2 \frac{1}{p_{X|Y}(x \mid y)}$$

- **Pogojna entropija**

$$\begin{aligned} H(X \mid Y) &= \sum_{y\in\mathcal{Y}} p(y)\, H(X \mid Y=y) \\ &= \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p_{X,Y}(x,y)\,\log_2 \frac{1}{p_{X|Y}(x \mid y)} \end{aligned}$$

# Reminder: mutual information and entropy

- Example:

| $P(x,y)$ | | $x$ | | | | $P(y)$ |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | |
| | 1 | 1/8 | 1/16 | 1/32 | 1/32 | 1/4 |
| $y$ | 2 | 1/16 | 1/8 | 1/32 | 1/32 | 1/4 |
| | 3 | 1/16 | 1/16 | 1/16 | 1/16 | 1/4 |
| | 4 | 1/4 | 0 | 0 | 0 | 1/4 |
| $P(x)$ | | 1/2 | 1/4 | 1/8 | 1/8 | |

**We assume p(y | x) known !!**

*H(X,Y) = 27/8 bitov, H(X) = 7/4 bitov, H(Y) = 2 bits*

*Conditional probability and entropy, and mutual information:*

| $P(x \mid y)$ | | $x$ | | | | $H(X \mid y)/\text{bits}$ |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | |
| | 1 | 1/2 | 1/4 | 1/8 | 1/8 | 7/4 |
| $y$ | 2 | 1/4 | 1/2 | 1/8 | 1/8 | 7/4 |
| | 3 | 1/4 | 1/4 | 1/4 | 1/4 | 2 |
| | 4 | 1 | 0 | 0 | 0 | 0 |
| | | | | | | $H(X \mid Y) = 11/8$ |

*H(X|Y) = 11/8 bits,*

*I(X,Y) = H(X) – H(X|Y) = 3/8 bits*

# Example: binary noiseless channel

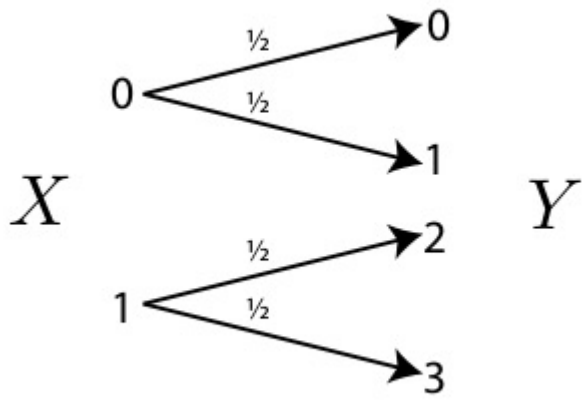
- Noiseless binary channel, diagram shows $p(y|x)$


- So, $p(y = 0|x = 0) = 1 = 1 - p(y = 1|x = 0)$ and $p(y = 1|x = 1) = 1 = 1 - p(y = 0|x = 1)$, so channel is just an input copy.
- One bit sent at a time is received without error, so capacity should be 1 bit (intuitively, we can reliably send one bit per channel usage).
- $I(X;Y) = H(X) - H(X|Y) = H(X)$ in this case, so $C = \max_{p(x)} I(X;Y) = \max_{p(x)} H(X) = 1$.

$$C = \max I(X;Y) = 1\text{bit}, \quad p(x) = (\frac{1}{2}, \frac{1}{2})$$

# Noisy channel with non-overlapping outputs

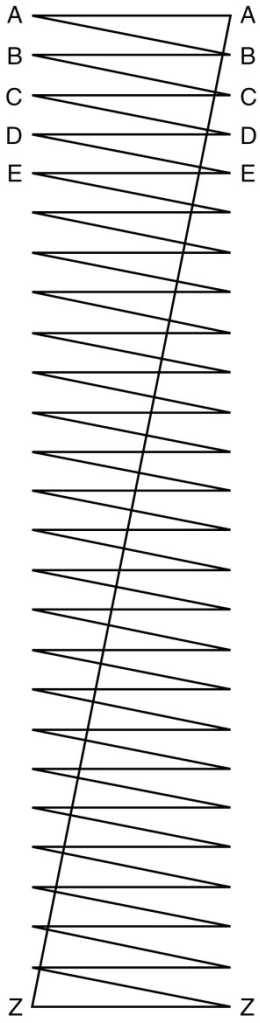
Consider the channel




- Here, $p(Y = 0|X = 0) = p(Y = 1|X = 0) = 1/2$ and $p(Y = 2|X = 1) = p(Y = 3|X = 1) = 1/2$.
- If we receive a 0 or 1, we know 0 was sent. If we receive a 2 or 3, a 1 was sent.
- Thus, $C = 1$ since only two possible error free messages.
- Same argument applies $I(X;Y) = H(X) - \underbrace{H(X|Y)}_{=0} = H(X)$.
- Again uniform distribution $p(0) = p(1) = 1/2$ achieves the capacity.

$$C = \max I(X;Y) = 1\text{bit}, \quad p(x) = (\frac{1}{2}, \frac{1}{2})$$

# ‘Noisy typewriter’



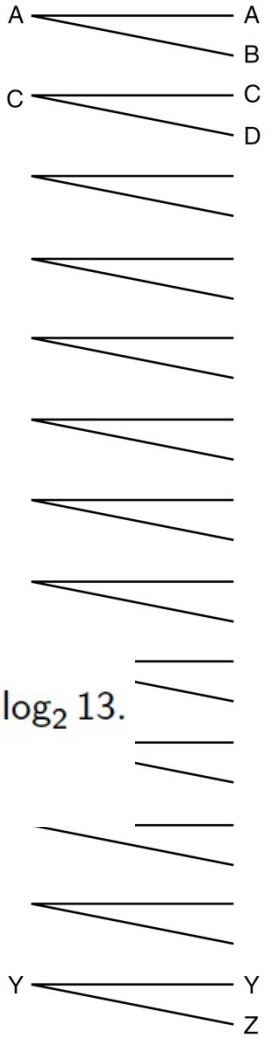

$$P(y = A|x = A) = \frac{1}{2}$$
$$P(y = B|x = A) = \frac{1}{2}$$
$$\ldots$$

We have 26 symbols and can transmit at most 13 without any decoding error (each second).
The capacity of channel is $\log 13$.

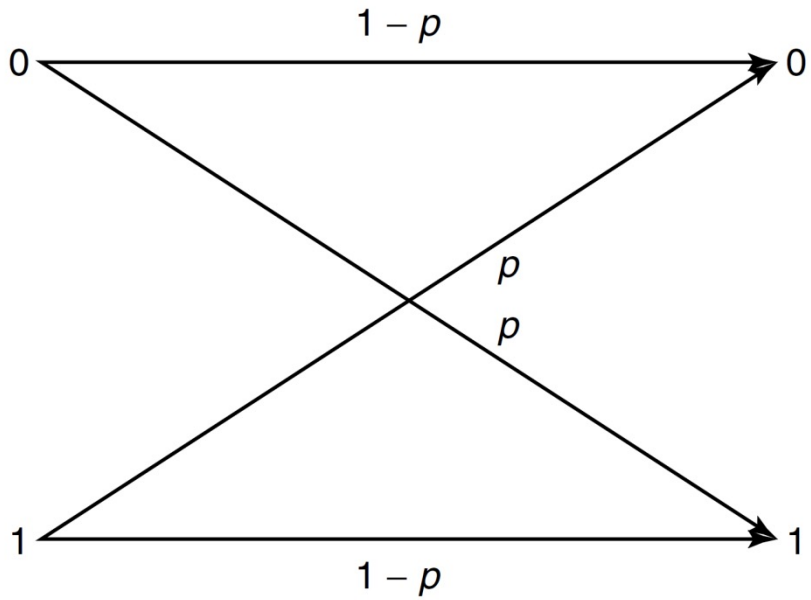
We can also compute: $C = \max I(X;Y) =$

$$\max(H(Y) - H(Y|X)) = \max H(Y) - 1 = \log_2 26 - 1 = \log_2 13.$$

using that $p(x)$ is uniformly distributed.



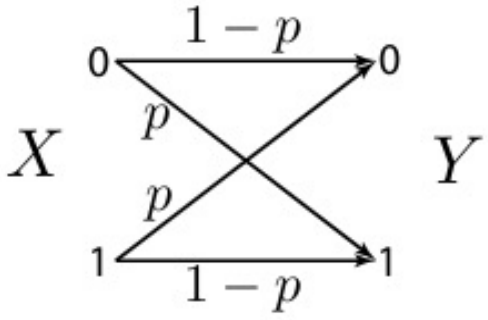
# Binary symmetric channel




$$
\begin{aligned}
I(X;Y) &= H(Y) - H(Y|X) \\
&= H(Y) - \sum p(x) H(Y|X = x) \\
&= H(Y) - \sum p(x) H(p) \\
&= H(Y) - H(p) \\
&\leq 1 - H(p),
\end{aligned}
$$

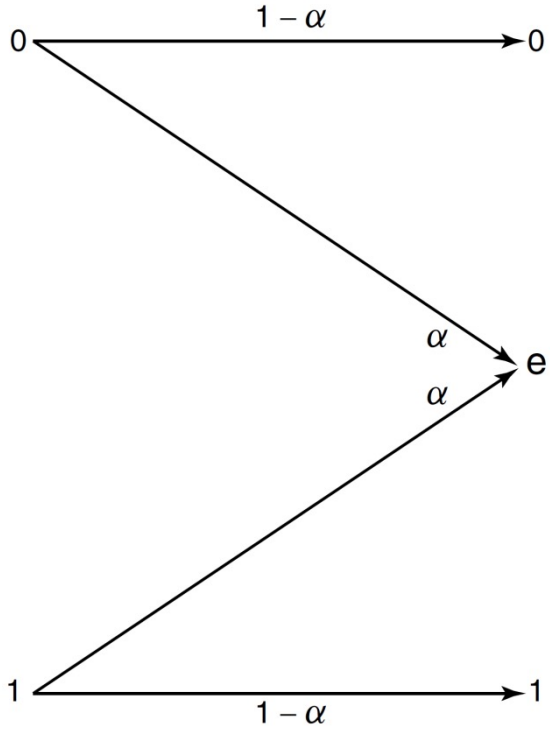
$$
\boxed{C = 1 - H(p)}
$$

# BSC capacity




- Thus, we get that $C = 1 - H(p)$ which happens when $X$ is uniform.
- If $p = 1/2$ then $C = 0$, so if it randomly flips bits, then no information can be sent.
- If $p \neq 1/2$, then we can communicate, albeit potentially slowly. E.g., if $p = 0.499$ then $C = 2.8854 \times 10^{-6}$ bits per channel use. So to send one bit, need to use the channel quite a bit.
- If $p = 0$ or $p = 1$, then $C = 1$ and we can get maximum possible rate (i.e., the capacity is one bit per channel use).

# Binary channel with erasure




Let $\Pr(X = 0) = \pi$ , then

$$
\begin{aligned}
C &= \max_{p(x)} I(X;Y) = \max_{p(x)} (H(X) - H(X|Y)) \\
&= \max_{p(x)} H(X) - \sum p(y) H(X|Y=y) \\
&= \max_{p(x)} H(X) - [\pi(1-\alpha)H(X|Y=0) + (1-\pi)(1-\alpha)H(X|Y=1 \\
&+ (\pi\alpha + (1-\pi)\alpha)H(X|Y=e)] \\
&= \max_{p(x)} H(X) - [\pi(1-\alpha)0 + (1-\pi)(1-\alpha)0 + \alpha H(X|Y=e)] \\
&= \max_{p(x)} H(X) - \alpha H(X) = (1-\alpha) \max_{p(x)} H(X) = 1-\alpha
\end{aligned}
$$

pri $\pi = \frac{1}{2}$

# Symmetric channels

- Example: We have a channel with the transition matrix:

$$p(y|x) = \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.5 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \end{bmatrix}$$

- Every row is a permutation of another row, same for columns. Such channels are called **symmetric**.

- BSC channel is symmetric.
- The capacity of BSC is $C = 1 - H(p)$

# Capacity of symmetric channels

**Capacity of symmetric channel**

The capacity of a symmetric channel is given by:

$$C = \log |\mathcal{Y}| - H(\text{one row of matrix } p(y|x)).$$

The capacity is achieved when $X$ has uniform probability distribution.

In our example:

$$C = \max_{p(x)} I(X;Y) = \log 3 - H(0.5, 0.3, 0.2)$$

Assuming the uniform distribution of the input symbols.

# Channel capacity - properties

Lastnosti kapacitete kanalov:

1. $C \geq 0$, ker je $I(X;Y) \geq 0$.
2. $C \leq \log |\mathcal{X}|$, ker je $C = \max I(X;Y) \leq \max H(X) = \log |\mathcal{X}|$.
3. $C \leq \log |\mathcal{Y}|$ iz istega razloga.
4. $I(X;Y)$ je zvezna funkcija $p(x)$.
5. $I(X;Y)$ je konkavna funkcija $p(x)$.

Ker je $I(X;Y)$ konkavna funkcija na konveksni ovojnici $p(x)$, potem je lokalni maksimum tudi globalni maksimum. Iz druge in tretje lastnosti sledi, da je maksimum omejen, torej je končen.

V splošnosti ne moremo eksplicitno izračunati kapacitete poljubnega kanala. Gornje lastnosti pa nam zagotavljajo, da maksimum obstaja in da je končen. Za izračun lahko uporabimo metode nelinearne optimizacije, npr. gradientni postopek.

# Decoding

- We can "decode" the source using the received string, source distribution, and the channel model $p(y|x)$ via Bayes rule. I.e.

$$\Pr(x|y) = \frac{\overbrace{\Pr(y|x)}^{\text{channel}}\overbrace{\Pr(x)}^{\text{source}}}{\Pr(y)} = \frac{\Pr(y|x)\Pr(x)}{\sum_{x'}\Pr(y|x')\Pr(x')}$$

- If we get a particular $y$, we can compute $p(x|y)$ and make a decision based on that. I.e., $\hat{x} = \operatorname{argmax}_x p(x|y)$.
- This is optimal decoding in that it minimizes the error.
- Error if $x \neq \hat{x}$, and $\Pr(\text{error}) = \Pr(x \neq \hat{x})$.

# Channel capacity and channel coding

- Why is the channel capacity important?

- Shannon:

  **Channel capacity is upper limit on the amount of information that can be transmitted over (noisy) channel with an arbitrary small of errors.**

- We can formalize this statement.

# Shannon‘s result  on channel coding

**Shannon's result on channel coding**

For a DMC channel with capacity $C$ the following property is valid. For arbitrary $\epsilon > 0$ and the number $R < C$ there exists a block code of length $N$ and the transmission rate $\geq R$ (but $< C$) and the decoding algorithm so that the decoding error (for large $N$) is smaller than $\epsilon$.

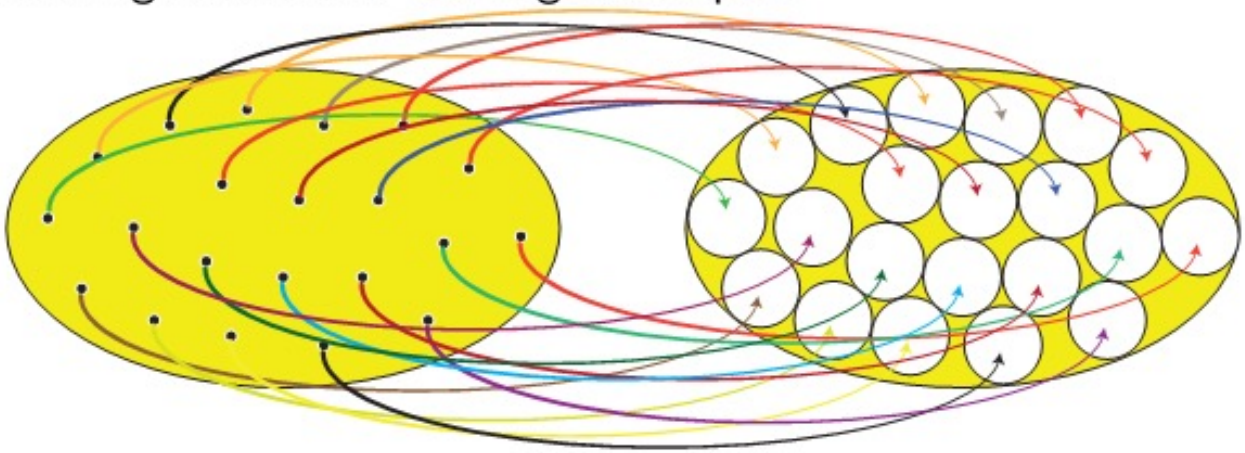
**Shannon's result on channel coding -mathematical version**

For a DMC channel with capacity $C$ the following property is valid. For any information rate $R < C$ there exists a block code of length $(N, K)$ (defining $R = K/N$) such that the error probability of decoding $\lambda^{(N)} \to 0$ when $N \to \infty$.

**Converse:** Any block code $(N, K)$ such that $\lambda^{(N)} \to 0$, when $N \to \infty$, has rate $R < C$.

# Shannon 2nd theorem

- Intuition of this we've already seen in the noisy typewriter and the region partitioning.
- Slightly more precisely, this is a sort of bin packing problem.
- We've got a region of possible codewords, and we pack as many smaller non-overlapping bins into the region as possible.
- The smaller bins correspond to the noise in the channel, and the packing problem depends on the underlying "shape"
- Not really a partition, since there might be wasted space, also depending on the bin and region shapes.



# Shannon - intuition

- Intuitive idea: use typicality argument, like in chapter 3.
- There are $\approx 2^{nH(X)}$ typical sequences, each with probability $2^{-nH(X)}$ and with $p(A_\epsilon^{(n)}) \approx 1$, so the only thing with "any" probability is the typical set and it has all the probability.
- The same thing is true for conditional entropy.
- That is, for a typical input $X$, there are $\approx 2^{nH(Y|X)}$ output sequences.
- Overall, there are $2^{nH(Y)}$ typical output sequences, and we know that $2^{nH(Y)} \geq 2^{nH(Y|X)}$.

# Shannon – intuition II

- Goal: find a non-confusable subset of the inputs that produce disjoint output sequences (as in picture).
- There are $\approx 2^{nH(Y)}$ (typical) outputs (i.e., the marginally typical $Y$ sequences).
- There are $\approx 2^{nH(Y|X)}$ ($X$-conditionally typical $Y$ sequences) outputs. $\equiv$ the average possible number of outputs for a possible input, so this many could be confused with each other. I.e., on average, for a given $X = x$, this is approximately how many outputs there might be.
- So the number of non-confusable inputs is

$$\leq \frac{2^{nH(Y)}}{2^{nH(Y|X)}} = 2^{n(H(Y)-H(Y|X))} = 2^{nI(X;Y)}$$

# Shannon – intuition III

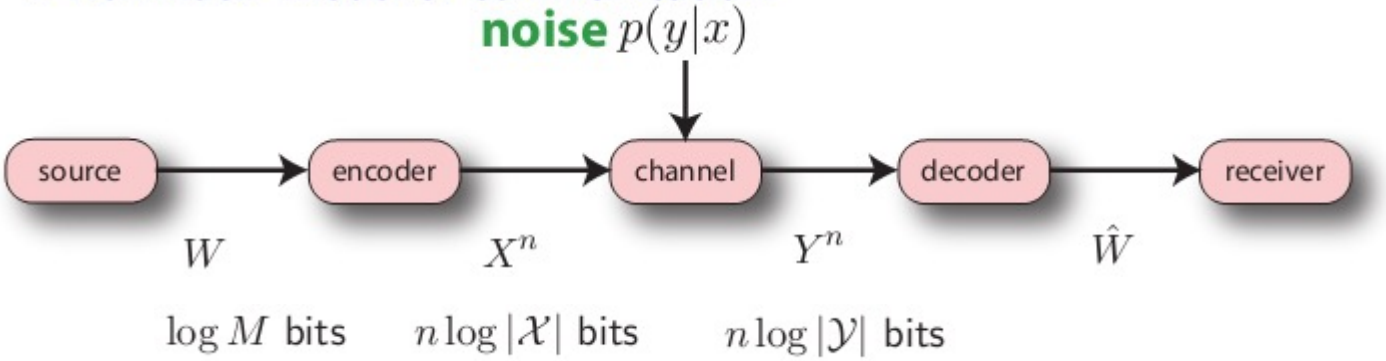
- The number of non-confusable inputs is

$$\leq \frac{2^{nH(Y)}}{2^{nH(Y|X)}} = 2^{n(H(Y)-H(Y|X))} = 2^{nI(X;Y)} \tag{19}$$

- Now of course, to maximize this number, for a fixed channel $p(y|x)$, we find the best $p(x)$ which gives $I(X;Y) = C$, which is the log of the maximum number of inputs possible to use.
- This is the capacity.

# Some definitions

- Reminder: model of communication:




- Message $W \in \{1, \ldots, M\}$ requiring $\log M$ bits per message.
- Signal sent through channel $X^n(W)$, a random codeword.
- Received signal from channel $Y^n \sim p(y^n|x^n)$
- Decoding via guess $\hat{W} = g(Y^n)$.
- Discrete memoryless channel (DMC) $(\mathcal{X}, p(y|x), \mathcal{Y})$

# *(M, n)* code

**Definition 5.4 ($(M, n)$ code)**

An $(M, n)$ code for channel $(\mathcal{X}, p(y|x), \mathcal{Y})$ is:

1. An index set $\{1, 2, \ldots, M\}$
2. An encoding function $X^n : \{1, 2, \ldots, M\} \to \mathcal{X}^n$ yielding codewords $X^n(1)$, $X^n(2)$, $X^n(3)$, $\ldots$, $X^n(M)$. Each source message has a codeword, and each codeword is $n$ code symbols.
3. Decoding function, i.e., $g : \mathcal{Y}^n \to \{1, 2, \ldots, M\}$ which makes a "guess" about original message given channel output.

- In an $(M, n)$ code, $M =$ the number of possible messages to be sent, and $n =$ number of channel uses by the codewords of the code.

# Decoding error

**Definition 5.5 (Probability of Error $\lambda_i$ for message $i \in \{1, \ldots, M\}$)**

$$\lambda_i \triangleq \Pr(g(Y^n) \neq i | X^n = X^n(i)) = \sum_{y^n \in \mathcal{Y}^n} p(y^n | X^n(i)) \mathbf{1}(g(y^n) \neq i) \tag{20}$$

**Definition 5.6 (Max probability of Error $\lambda^{(n)}$ for $(M, n)$ code)**

$$\lambda^{(n)} \triangleq \max_{i \in \{1,2,\ldots,M\}} \lambda_i \tag{21}$$

# Rate of the code

**Definition 5.8 (Rate $R$ of an $(M, n)$ code)**

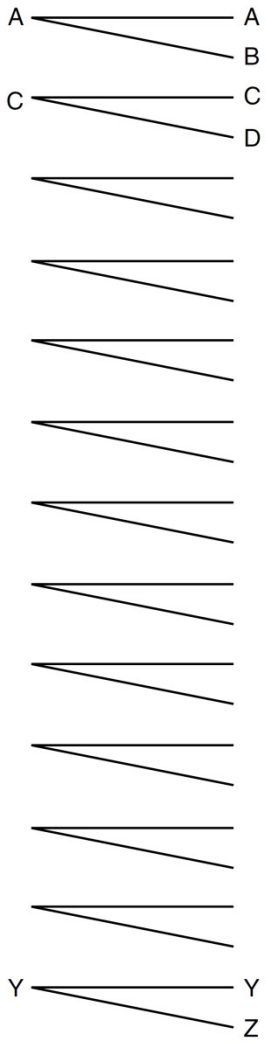
$$R = \frac{\log M}{n} = \frac{\text{total num. of bits in a source message}}{\text{total num. of channel uses needed to send a message}} \tag{24}$$

- The rate $R$ is in units of bits per channel use, or bits per transmission.

**Definition 5.9 (acheivability for a given channel)**

A given rate $R$ is achievable for a given channel if $\exists$ a sequence of $(\lceil 2^{nR} \rceil, n)$ codes such that the maximal probability of error $\lambda^{(n)} \to 0$ as $n \to \infty$.

# Shannon‘s result on channel coding - ‘noisy typewriter’

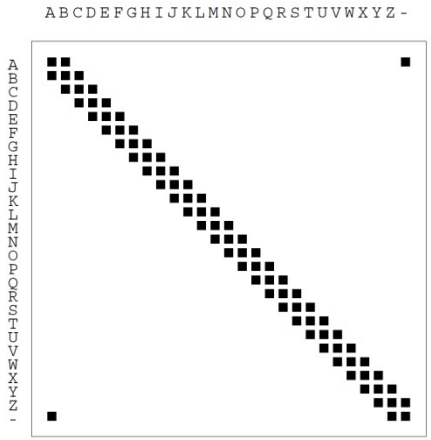



$P(y = A|x = A) = \frac{1}{2}$
$P(y = B|x = A) = \frac{1}{2}$

| | |
|---|---|
| For DMC noisy typewriter with capacity *C* we have: | *C = log13* |
| For arbitrary *ϵ > 0* and number *R < C there exists* block code **of length *N*.** | No matter what are ϵ and *R.* For *N* we select 1. |
| **There exists block code** of length *N* and **rate** ≥ *R* | Block code is {A, C, E, …, Y}. Then K = log13 and rate = K/N = log13. It is larger than R. |
| and **decoding procedure** | When decoding we map the recieved codeword to the closest input. |
| And for suffiicently large *N* **maximal decoding error** < *ϵ*. | In our case, maximal decoding error is 0, smaller than ϵ. |

# Shannon's result proof

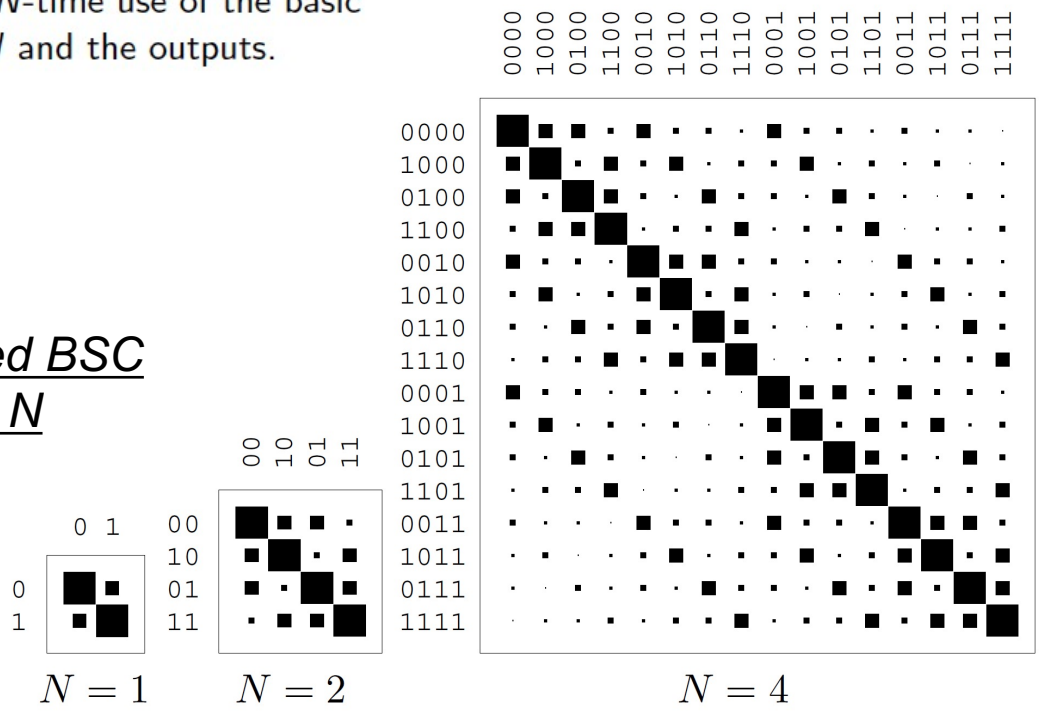
Main idea: Any channel behaves like noisy typewriter for block codes of large length $N$. Meaning that different inputs are mapped into disjoint (non-confusable) outputs.




*Noisy typewriter*

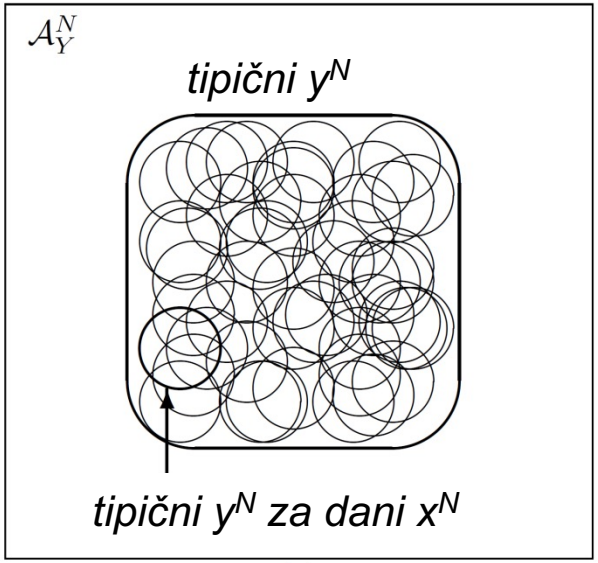
Extended channel of order $N$ is defined as $N$-time use of the basic channel, thus looking at blocks of length $N$ and the outputs.

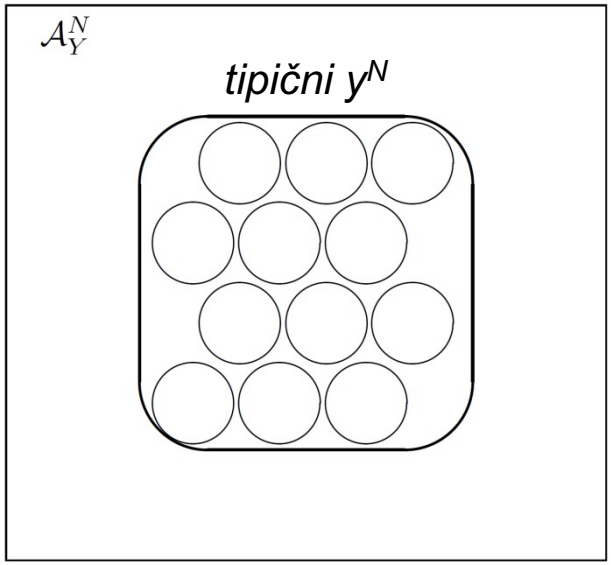
*Extended BSC of order N*



# Dokaz Shannonovega izreka (proof Slovenian version)

Poljuben vhodni niz $x^N$ se tako lahko po razširjenem kanalu prenese v nekaj možnih izhodnih nizov $y^N$. Za velike $N$ se da pokazati, da se lahko vhodni niz z veliko verjetnostjo prenese v samo nekaj izhodnih nizov, torej v zelo majhno podmnožico izhodnih nizov v primerjavi z množico vseh možnih izhodnih nizov. Parom nizov $(x^N, y^N)$, ki ustrezajo zgornji lastnosti, pravimo *tipični nizi*.




Vsak (tipičen) niz $x^N$ se lahko prenese v približno $2^{NH(Y|X)}$ možnih $y^N$ nizov, ki so vsi približno enako verjetni (AEP). Če hočemo pravilno dekodirati prenešene nize, seveda želimo, da se noben par vhodnih nizov ne preslika v isti izhodni niz, sicer lahko pride do napak pri dekodiranju. To pomeni, da moramo množice izhodnih nizov za dane vhodne nize razredčiti, da bodo med seboj disjunktne.

# Dokaz Shannonovega izreka (Slovenian version)

Koliko je takšnih disjunktnih množic?

Skupno število vseh tipičnih izhodnih nizov je približno $2^{NH(Y)}$. En tipičen vhodni niz zavzame (se preslika v) približno $2^{NH(Y|X)}$ možnih izhodnih nizov. Tako je prostora za skupno število disjunktnih množic nizov navzgor omejeno z $\frac{2^{N(H(Y)}}{2^{NH(Y|X)}} = 2^{N(H(Y)-H(Y|X))} = 2^{NI(X;Y)}$.

Torej, po $N$-krat razširjenem kanalu lahko prenesemo največ $2^{NI(X;Y)}$ nizov dolžine $N$, ki jih je možno brez napak dekodirati. Največja zgornja meja je dosežena v primeru, ko maksimiziramo medsebojno informacijo $I(X;Y)$, to pa je ravno definicija kapacitete kanala $C$. V tem primeru tako lahko prenesemo največ $2^{NC}$ nizov, oziroma se stopnja prenosa informacije poljubno približa kapaciteti C bitov na posamezen prenos z maksimalno napako prenosa, ki gre proti 0.

# Block codes

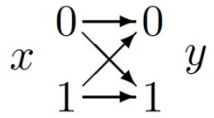
- Shannon‘s result states that there exist block codes, with rate $R < C$, so that we can transmit information realiably (arbitrary small eror of decoding) for large $N$.
- Apart from small decoding error, we require that both coding and decoding are efficient (form implementation point of view)
- We will consider two examples of block codes:
  - Repetition coding,
  - Hamming codes, as a special case of linear block codes.

# Repetition codes

- Simplest coding scheme (least efficient) is a **repetition code:**
- Example:
  - We want to transmit 1 over a channel, and the coding implies that we reapeat 1 $N$ times, e.g. $N$=3, thus 1 $\rightarrow$ 111 and 0 $\rightarrow$ 000
  - Coding uses 3 symbols, to transmit 1 bit, hence the rate is $R$ =1/3 bits/symbol (bits/per channel use).
  - If the channel is BSC, optimal decoding procedure is 'majority rule': count the number of 1s and 0s. The largest number decides what was sent.
- We consider this approach in more detail.

# Repetition codes

- Given is a BSC channel:




$$P(y=0\,|\,x=0) = 1-f; \quad P(y=0\,|\,x=1) = f;$$
$$P(y=1\,|\,x=0) = f; \quad P(y=1\,|\,x=1) = 1-f.$$

- We send the following bits through the channel:

$$\mathbf{s} = 0\ 0\ 1\ 0\ 1\ 1\ 0$$

- Use repetition code with $N$=3, say $R_3$

| **s** | **t** |
|---|---|
| 0 | 000 |
| 1 | 111 |

- **Transmission**: Use the following vector addition **r** = **t**+**n** (modulo 2), where **n** is the error bit vector (noise):

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **s** | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| **t** | 000 | 000 | 111 | 000 | 111 | 111 | 000 |
| **n** | 000 | 001 | 000 | 000 | 101 | 000 | 000 |
| **r** | 000 | 001 | 111 | 000 | 010 | 111 | 000 |
| $\hat{\mathbf{s}}$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

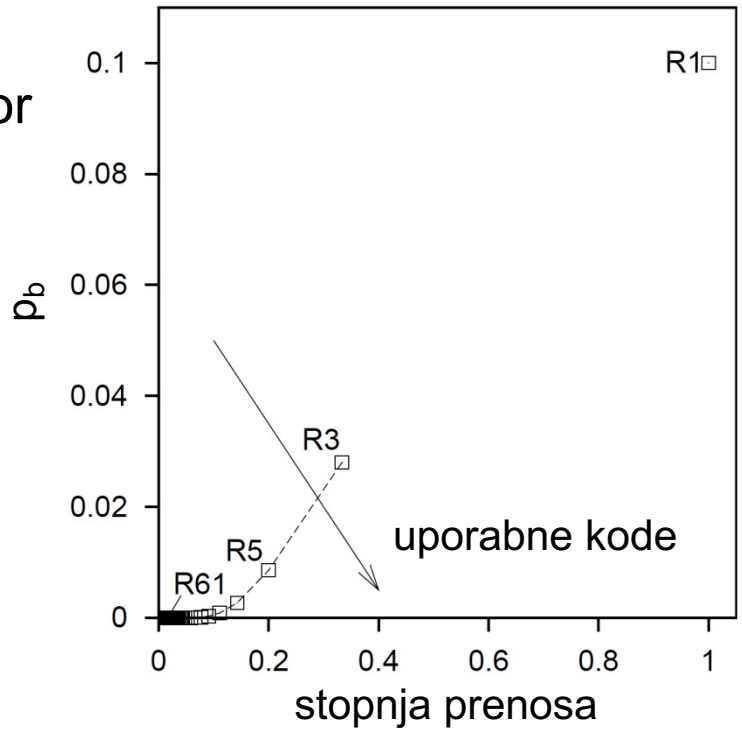
- **Decoding**:

# Repetition codes

- Easy to show that the probability of error for *N* odd is:

$$p_{\mathrm{b}} = \sum_{n=(N+1)/2}^{N} \binom{N}{n} f^n (1-f)^{N-n} \simeq \frac{1}{\sqrt{\pi N/8}} f[4f(1-f)]^{(N-1)/2}$$

- Rate versus probability of error for Pb(0 →1*)* = *p* = 0.1 :




**For small $p_b$ (when N $\rightarrow\infty$) the information rate *R* goes to zero ( *R* = 1/N). BAD!!**

# Repetition code - example

Suppose we use the three-symbol repetition code for a BSC with $f = 0.1$. Assume that the probability of 000 being sent is 0.7 and the probability of 111 being sent is 0.3.

What codeword should the decoder guess if the received symbols are 101?

$$
\begin{aligned}
&P(w = 000 \,|\, b_1 = 1, b_2 = 0, b_3 = 1) \\
&\quad = \frac{P(w = 000)\, P(b_1 = 1, b_2 = 0, b_3 = 1 \,|\, w = 000)}{P(b_1 = 1, b_2 = 0, b_3 = 1)} \\
&\quad = \frac{0.7 \times 0.1 \times 0.9 \times 0.1}{0.7 \times 0.1 \times 0.9 \times 0.1 + 0.3 \times 0.9 \times 0.1 \times 0.9} = 0.206
\end{aligned}
$$

$$
\begin{aligned}
&P(w = 111 \,|\, b_1 = 1, b_2 = 0, b_3 = 1) \\
&\quad = \frac{P(w = 111)\, P(b_1 = 1, b_2 = 0, b_3 = 1 \,|\, w = 000)}{P(b_1 = 1, b_2 = 0, b_3 = 1)} \\
&\quad = \frac{0.3 \times 0.9 \times 0.1 \times 0.9}{0.7 \times 0.1 \times 0.9 \times 0.1 + 0.3 \times 0.9 \times 0.1 \times 0.9} = 0.794
\end{aligned}
$$

The decoder should guess that 111 was sent.

# Associating codewords with messages

Suppose our original message is a sequence of $K$ bits. (Or we might break our message up into $K$-bit blocks.)

If we use a code with $2^K$ codewords, we can send this message (or block) as follows:

- The encoder maps the block of $K$ message symbols to a codeword.
- The encoder transmits this codeword.
- The decoder guesses at the codeword sent.
- The decoder maps the guessed codeword back to a block of $K$

# Shannon – once again

Shannon's noisy coding theorem states that:

> For any channel with capacity $C$, any desired error probability, $\epsilon > 0$, and any transmission rate, $R < C$, there exists a code with some length $N$ having rate at least $R$ such that the probability of error when decoding this code by maximum likelihood is less than $\epsilon$.

In other words: We can transmit at a rate arbitrarily close to the channel capacity with arbitrarily small probability of error.

A near converse is also true: We *cannot* transmit with arbitrarily small error probability at a rate greater than the channel capacity.

# Why cannot we use BSC beyond the capacity ?

We'll see here that if we could transmit through a BSC beyond the capacity $C = 1 - H_2(f)$, with vanishingly small error probability, we could compress data to less than its entropy, which we know isn't possible.

In particular, suppose that we can encode blocks of length $K$ into codewords of length $N$, with a very small probability of decoding error...

Here's how we use this code to compress data:

- Represent our data as two blocks: $x$ is of length $K$ and has bit probabilities of $1/2$, $y$ is of length $N$ and has bit probabilities of $f$ and $1-f$. The total information in $x$ and $y$ is $K + NH_2(f)$.

# Going below entropy

- Represent our data as two blocks: $x$ is of length $K$ and has bit probabilities of $1/2$, $y$ is of length $N$ and has bit probabilities of $f$ and $1-f$. The total information in $x$ and $y$ is $K + NH_2(f)$.
- Encode $x$ in a codeword $w$ of length $N$, and compute $z = w + y$, with addition modulo 2. This $z$ is the compressed form of the data.
- Apply the decoding algorithm to recover $w$ from $z$, treating $y$ as noise. We can then recover $x$ from $w$ and also $y = z - w$.
- The encoder checks whether the previous step would produce any errors, and if so transmits extra bits to identify corrections. This adds only a few bits, on average.

**The result:** We compressed a source with entropy $K + NH_2(f)$ into only slightly more than $N$ bits.

This is possible only if $N \geq K + NH_2(f)$, which implies that

$$R = K/N \leq 1 - H_2(f) = C$$

# Vector spaces over $Z_2$

Just as we can define vectors over the reals, we can define vectors over any other field, including over $Z_2$. We get to add such vectors, and multiply them by a scalar from the field.

We can think of these vectors as $N$-tuples of field elements. For instance, with vectors of length five over $Z_2$:

$$\begin{aligned}(1,0,0,1,1)+(0,1,0,0,1) &= (1,1,0,1,0)\\ 1\cdot(1,0,0,1,1) &= (1,0,0,1,1)\\ 0\cdot(1,0,0,1,1) &= (0,0,0,0,0)\end{aligned}$$

We refer to the vector space of all $N$-tuples from $Z_2$ as $Z_2^N$. We will use boldface letters such as $\mathbf{u}$ and $\mathbf{v}$ to refer to such vectors.

# Linear codes

We can view $Z_2^N$ as the input and output alphabet of the $N$th extension of a binary channel.

A code, $\mathcal{C}$, for this extension of the channel is a subset of $Z_2^N$.

$\mathcal{C}$ is a *linear code* if the following conditions hold:

1) If $\mathbf{u}$ and $\mathbf{v}$ are codewords of $\mathcal{C}$, then $\mathbf{u}+\mathbf{v}$ is also a codeword of $\mathcal{C}$.
2) If $\mathbf{u}$ is a codeword of $\mathcal{C}$ and $z$ is in $\mathcal{A}_X$, then $z\mathbf{u}$ is also a codeword of $\mathcal{C}$.

In other words, $\mathcal{C}$ must be a subspace of $\mathcal{A}_X^N$. Note that the all-zero codeword must be in $\mathcal{C}$, since $\mathbf{0} = 0\mathbf{u}$ for any $\mathbf{u}$.
In other words, $\mathcal{C}$ must be a subspace of $\mathcal{A}_X^N$. Note that the all-zero codeword must be in $\mathcal{C}$, since $\mathbf{0} = 0\mathbf{u}$ for any $\mathbf{u}$.

Note: For binary codes, where $\mathcal{A}_X = Z_2$, condition (2) will always hold if condition (1) does, since $1\mathbf{u} = \mathbf{u}$ and $0\mathbf{u} = \mathbf{0} = \mathbf{u} + \mathbf{u}$.

# Linear codes from basis vectors

We can construct a linear code by choosing $K$ linearly-independent *basis vectors* from $Z_2^N$.

We'll call the basis vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$. We define the set of codewords to be all those vectors that can be written in the form

$$a_1\mathbf{u}_1 + a_2\mathbf{u}_2 + \cdots + a_K\mathbf{u}_K$$

where $a_1, \ldots, a_K$ are elements of $Z_2$.

The codewords obtained with different $a_1, \ldots, a_K$ are all different. (Otherwise $\mathbf{u}_1, \ldots, \mathbf{u}_K$ wouldn't be linearly-independent.)

There are therefore $2^K$ codewords. We can encode a block consisting of $K$ symbols, $a_1, \ldots, a_k$, from $Z_2$ as a codeword of length $N$ using the formula above.

This is called an $[N, K]$ code. (MacKay's book uses $(N, K)$, but that has another meaning in other books.)

# Vector spaces - counting

- What is a $k$-dim. vector subspace $S \subset V_n(F)$?
- Simply, subspace is determined by $k$ linearly independent vectors in $V_n(F)$

Example Recall our code $C = \{00000, 10110, 01011, 11101\}$. Then any two vectors in $C \setminus \{\mathbf{0}\}$ are linearly independent. E.g. taking as basis $\mathbf{c_1} = 10110, \mathbf{c_2} = 01011$ we get $C$ as,

$$C = a_1 c_1 + a_2 c_2, (a_1, a_2) \in F;\ F = GF(2^2)$$

Three different basis (six up to permutation), same code !

- In general, the number of selecting $k$ lin. ind. vectors is

$$(q^n - 1)(q^n - q)(q^n - q^2) \cdots (q^n - q^{k-1}) = \prod_{i=0}^{k-1} (q^n - q^i).$$

# Counting spaces

- Each $k$-dimensional subspace contains

$$(q^k - 1)(q^k - q)(q^k - q^2)\cdots(q^k - q^{k-1}) = \prod_{i=0}^{k-1}(q^k - q^i)$$

ordered sets of $k$ linearly independent vectors.

- The total number of $k$-dimensional subspaces in $V_n(F)$ is,

$$\frac{\prod_{i=0}^{k-1}(q^n - q^i)}{\prod_{i=0}^{k-1}(q^k - q^i)}$$

Example In our case $q = 2$, $n = 5$, $k = 2$

$$\prod_{i=0}^{k-1}(q^n - q^i) = \prod_{i=0}^{1}(2^5 - 2^i) = 31 \cdot 30 = 960.$$

# Counting subspaces II

$$\prod_{i=0}^{k-1}(q^k-q^i)=\prod_{i=0}^{1}(2^2-2^i)=3\cdot 2=6;\ \frac{\prod_{i=0}^{k-1}(q^n-q^i)}{\prod_{i=0}^{k-1}(q^k-q^i)}=\frac{960}{6}=160.$$

Where does this 6 comes from ?

| | |
|---|---|
| (10000), (01000) | (01000), (10000) |
| (11000), (01000) | (01000), (11000) |
| (11000), (10000) | (10000), (11000) |

All gives the same subspace !

# Linear codes from equations

Another way to define a linear code for $Z_2^N$ is to provide a set of simultaneous equations that must be satisfied for $\mathbf{v}$ to be a codeword.

These equations have the form $\mathbf{c} \cdot \mathbf{v} = 0$, ie

$$c_1 v_1 + c_2 v_2 + \cdots + c_N v_N = 0$$

The set of solutions is a linear code because

1) $\mathbf{c} \cdot \mathbf{u} = 0$ and $\mathbf{c} \cdot \mathbf{v} = 0$ implies $\mathbf{c} \cdot (\mathbf{u} + \mathbf{v}) = 0$.
2) $\mathbf{c} \cdot \mathbf{v} = 0$ implies $\mathbf{c} \cdot (z\mathbf{v}) = 0$.

If we have $N - K$ such equations, and they are independent, the code will have $2^K$ codewords.

# The repetition code over $Z_2$

A repetition code for $Z_2^N$ has only two codewords — one has all 0s, the other all 1s.

This is a linear $[N, 1]$ code, with $(1, \ldots, 1)$ as the basis vector.

The code is also defined by the following $N - 1$ equations satisfied by a codeword $\mathbf{v}$:

$$v_1 + v_2 = 0, \quad v_2 + v_3 = 0, \ \cdots, \quad v_{N-1} + v_N = 0$$

# The single parity-check equation

An $[N, N-1]$ code over $Z_2$ can be defined by the following single equation satisfied by a codeword $\mathbf{v}$:

$$v_1 + v_2 + \cdots + v_N = 0$$

In other words, the *parity* of all the bits in a codeword must be even.

This code can also be defined using $N-1$ basis vectors. One choice of basis vectors when $N = 5$ is as follows:

$$(1, 0, 0, 0, 1)$$
$$(0, 1, 0, 0, 1)$$
$$(0, 0, 1, 0, 1)$$
$$(0, 0, 0, 1, 1)$$

# A [5,2] linear code

Recall the following code from earlier in the lecture:

$$\{\ 00000,\ \ 00111,\ \ 11001,\ \ 11110\ \}$$

Is this a linear code? We need to check that all sums of codewords are also codewords:

$$00111 + 11001 = 11110$$

$$00111 + 11110 = 11001$$

$$11001 + 11110 = 00111$$

We can generate this code using 00111 and 11001 as basis vectors. We then get the four codewords as follows:

$$0 \cdot 00111 + 0 \cdot 11001 = 00000$$

$$0 \cdot 00111 + 1 \cdot 11001 = 11001$$

$$1 \cdot 00111 + 0 \cdot 11001 = 00111$$

$$1 \cdot 00111 + 1 \cdot 11001 = 11110$$

# The [7,4] binary Hamming code

The $[7, 4]$ Hamming code is defined over $Z_2$ by the following equations that are satisfied by a codeword $\mathbf{u}$:

$$\begin{aligned} u_1 + u_2 + u_3 + u_5 &= 0 \\ u_2 + u_3 + u_4 + u_6 &= 0 \\ u_1 + u_3 + u_4 + u_7 &= 0 \end{aligned}$$

Since these equations are independent, there should be 16 codewords.

We can also define the code in terms of the following four basis vectors:

$$1000101, \quad 0100110, \quad 0010111, \quad 0001011$$

There are other sets equations and other sets of basis vectors that define the same code.

We will see later that this code is capable of correcting any single error.

# Generator matrix

We can arrange a set of basis vectors for a linear code in a *generator matrix*, each row of which is a basis vector.

A generator matrix for an $[N, K]$ code will have $K$ rows and $N$ columns.

Here's a generator matrix for the $[5, 2]$ code looked at earlier:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

# Encoding information bits using generator matrix

We can use a generator matrix for an $[N, K]$ code to encode a block of $K$ message bits as a block of $N$ bits to send through the channel.

We regard the $K$ message bits as a row vector, $\mathbf{s}$, and multiply by the generator matrix, $G$, to produce the channel input, $\mathbf{t}$:

$$\mathbf{t} = \mathbf{s}G$$

Every $\mathbf{t}$ that is a codeword will be produced by some $\mathbf{s}$. If the rows of $G$ are linearly independent, each distinct $\mathbf{s}$ will produce a different $\mathbf{t}$.

**Example:** Encoding the message block $(1, 1)$ using the generator matrix for the $[5, 2]$ code given earlier:

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

# Parity-check matrix

Suppose we have specified an $[N, K]$ code by a set of $M = N - K$ equations that any codeword, $\mathbf{v}$, must satisfy:

$$\begin{aligned} c_{1,1}\, v_1 + c_{1,2}\, v_2 + \cdots + c_{1,N}\, v_N &= 0 \\ c_{2,1}\, v_1 + c_{2,2}\, v_2 + \cdots + c_{2,N}\, v_N &= 0 \\ &\vdots \\ c_{M,1}\, v_1 + c_{M,2}\, v_2 + \cdots + c_{M,N}\, v_N &= 0 \end{aligned}$$

We can arrange the coefficients in these equations in a *parity-check matrix*, as follows:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ & & \vdots & \\ c_{M,1} & c_{M,2} & \cdots & c_{M,N} \end{bmatrix}$$

If $\mathcal{C}$ has parity-check matrix $H$, then $\mathbf{v}$ is in $\mathcal{C}$ if and only if $\mathbf{v}H^T = \mathbf{0}$.

Note: Almost all codes have more than one parity-check matrix.

# The [5,2] code

Here is one parity-check matrix for the $[5, 2]$ code used earlier:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

We see that 11001 is a codeword as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

But 10011 isn't a codeword, since

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

# Manipulating the parity check matrix

There are usually many parity-check matrices for a given code. We can get one such matrix from another using the following "elementary row operations":

- Swapping two rows.
- Multipling a row by a non-zero constant (not useful for $Z_2$).
- Adding a row to a different row.

These operations don't alter the solutions to the equations the parity-check matrix represents.

**Example:** This parity-check matrix for the example $[5, 2]$ code:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

can be transformed into this alternative:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

# Manipulating the generator matrix

We can apply the same elementary row operations to a generator matrix for a code, in order to produce another generator matrix, since these operations just convert one set of basis vectors to another.

**Example:** Here is a generator matrix for the $[5, 2]$ code we have been looking at:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Here is another generator matrix, found by adding the first row to the second:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

**Note:** These manipulations leave the set of codewords unchanged, but they *don't* leave the way we encode messages by computing $\mathbf{t} = \mathbf{s}G$ unchanged!

# Equivalent codes

Two codes are said to be *equivalent* if the codewords of one are just the codewords of the other with the order of symbols permuted.

Permuting the order of the columns of a generator matrix will produce a generator matrix for an equivalent code, and similarly for a parity-check matrix.

**Example:** Here is a generator matrix for the $[5, 2]$ code we have been looking at:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

We can get an equivalent code using the following generator matrix obtained by moving the last column to the middle:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

# Systematic form

Using elementary row operations and column permutations, we can convert any generator matrix to a generator matrix for an equivalent code that is in *systematic form*, in which the left end of the matrix is the identity matrix.

Similarly, we can convert to the systematic form for a parity-check matrix, which has an identity matrix in the right end.

For the $[5, 2]$ code, only permutations are needed. The generator matrix can be permuted by swapping columns 1 and 3:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

When we use a systematic generator matrix to encode a block $\mathbf{s}$ as $\mathbf{t} = \mathbf{s}G$, the first $K$ bits will be the same as those in $\mathbf{s}$. The remaining $N - K$ bits can be seen as "check bits".

# Equivalent codes - example

- Want to transform $C$ into $C'$ (equivalent not identical codes)

$$\tilde{G} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}; \; G' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Step 1: $\tilde{G} \to G$ (add row 1 to rows 2 and 3)

$$\tilde{G} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- Step 2: $G' = GP$ (interchange columns 1 and 3)

$$\tilde{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Generator and parity check matrix - relationship

If $G$ and $H$ are generator and parity-check matrices for $\mathcal{C}$, then for every $\mathbf{s}$, we must have $(\mathbf{s}G)H^T = \mathbf{0}$ — since we should only generate valid codewords. It follows that

$$GH^T = \mathbf{0}$$

Furthermore, any $H$ with $N - K$ independent rows that satisfies this is a valid parity-check matrix for $\mathcal{C}$.

Suppose $G$ is in systematic form, so

$$G = [\, I_K \mid P \,]$$

for some $P$. Then we can find a parity-check matrix for $\mathcal{C}$ in systematic form as follows:

$$H = [\, -P^T \mid I_{N-K} \,]$$

since $GH^T = -I_K P + P I_{N-K} = \mathbf{0}$.

# Hamming distance

Recall that the Hamming distance, $d(\mathbf{u}, \mathbf{v})$, of two codewords $\mathbf{u}$ and $\mathbf{v}$ is the number of positions where $\mathbf{u}$ and $\mathbf{v}$ have different symbols.

This is a proper distance, which satisfies the *triangle inequality*:

$$d(\mathbf{u}, \mathbf{w}) \leq d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{w})$$

Here's a picture showing why:

```
u:  0 1 1 0 0 1 1 0 1 1 1 0
                - - - - - -
v:  0 1 1 0 0 1 0 1 0 0 0 1
            - - -         - -
w:  0 1 1 1 1 0 0 1 0 0 1 0
```

Here, $d(\mathbf{u}, \mathbf{v}) = 6$, $d(\mathbf{u}, \mathbf{v} = 5)$, and $d(\mathbf{u}, \mathbf{w}) = 7$.

# Minimum distance and decoding

A code's *minimum distance* is the minimum of $d(\mathbf{u}, \mathbf{v})$ over all distinct codewords $\mathbf{u}$ and $\mathbf{v}$.

If the minimum distance is at least $2t + 1$, a nearest neighbor decoder will always decode correctly when there are $t$ or fewer errors.

Here's why: Suppose the code has distance $d \geq 2t + 1$. If $\mathbf{u}$ is sent and $\mathbf{v}$ is received, having no more than $t$ errors, then

- $d(\mathbf{u}, \mathbf{v}) \leq t$.
- $d(\mathbf{u}, \mathbf{u}') \geq d$ for any codeword $\mathbf{u}' \neq \mathbf{u}$.

From the triangle inequality:

$$d(\mathbf{u}, \mathbf{u}') \leq d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{u}')$$

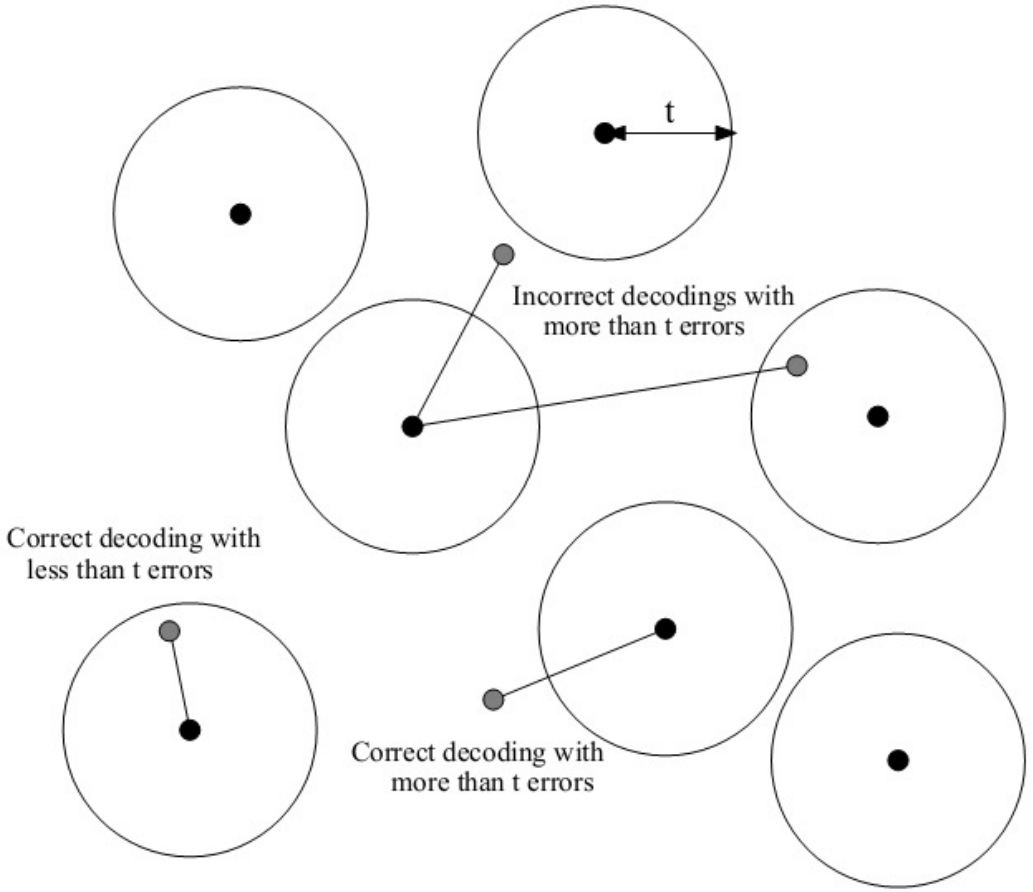
It follows that

$$d(\mathbf{v}, \mathbf{u}') \geq d(\mathbf{u}, \mathbf{u}') - d(\mathbf{u}, \mathbf{v}) \geq d - t \geq (2t + 1) - t \geq t + 1$$

The decoder will therefore decode correctly to $\mathbf{u}$, at distance $t$, rather than to some other $\mathbf{u}'$, at distance at least $t + 1$.

# Distance and decoding

Here's a picture of codewords (black dots) for a code with minimum distance $2t + 1$, showing how some transmissions are decoded:



# Decoding example

Consider the code $C$ (example 6) with codewords:

$$\mathbf{c}_1 = (00000), \mathbf{c}_2 = (10110), \mathbf{c}_3 = (01011), \mathbf{c}_4 = (11101)\}$$

If we would construct the spheres of radius 1 (since $d = 3$)

$$\begin{aligned} S_{c_1} &= \{(00000), (10000), (01000), (00100), (00010), (00001)\} \\ S_{c_2} &= \{(10110), (00110), (11110), (10010), (10100), (10111)\} \\ S_{c_3} &= \{(01011), (11011), (00011), (01111), (01001), (01010)\} \\ S_{c_4} &= \{(11101), (01101), (10101), (11001), (11111), (11100)\} \end{aligned}$$

The set of vectors that are not in spheres is,

$$S^* = \{(11000), (01100), (10001), (00101), (01110), (00111), (10011), (11010)\}.$$

# Decoding example II

- Let $\mathbf{r}$=(00011). Then we compute,

$$d(\mathbf{c}_1, \mathbf{r}) = 2, \quad d(\mathbf{c}_2, \mathbf{r}) = 3, \quad d(\mathbf{c}_3, \mathbf{r}) = 1, \quad d(\mathbf{c}_4, \mathbf{r}) = 4,$$

Decode as $\mathbf{c}_3$.

- Let $\mathbf{r} = (11000) \in S^*$. Then we compute,

$$d(\mathbf{c}_1, \mathbf{r}) = 2, \quad d(\mathbf{c}_2, \mathbf{r}) = 3, \quad d(\mathbf{c}_3, \mathbf{r}) = 3, \quad d(\mathbf{c}_4, \mathbf{r}) = 2.$$

Cannot decode, the receiver knows there are at least 2 errors.

- Suppose $\mathbf{c}_1$ is sent and 2 errors are present so that $\mathbf{r} = (10100)$. Receiver decides in favour of $\mathbf{c}_2$ (closest).

– But cannot detect 2 errors if used for error correcting. Without correcting can detect 2 errors.

# Minimum distance and error correction

Recall the linear $[5, 2]$ code with the following codewords:

$$00000 \quad 00111 \quad 11001 \quad 11110$$

The three non-zero codewords have weights of 3, 3, and 4. This code therefore has minimum distance 3, and can correct any single error.

The single-parity-check code with $N = 4$ has the following codewords:

$$0000 \quad 0011 \quad 0101 \quad 0110$$
$$1001 \quad 1010 \quad 1100 \quad 1111$$

The smallest weight of a non-zero codeword above is 2, so this is the minimum distance of this code. This is too small to guarantee correction of even one error. (Though the presence of a single error can be detected.)

# Minimum distance from a parity check

We can find the minimum distance of a linear code from a parity-check matrix for it, $H$. The minimum distance is equal to the smallest number of linearly-dependent columns of $H$.

Why? A vector $\mathbf{u}$ is a codeword iff $\mathbf{u}H^T = \mathbf{0}$. If $d$ columns of $H$ are linearly dependent, let $\mathbf{u}$ have 1s in those positions, and 0s elsewhere. This $\mathbf{u}$ is a codeword of weight $d$. And if there were any codeword of weight less than $d$, the 1s in that codeword would identify a set of less than $d$ linearly-dependent columns of $H$.

Special cases:

- If $H$ has a column of all zeros, then $d = 1$.
- If $H$ has two identical columns, then $d \leq 2$.
- For binary codes, if all columns are distinct and non-zero, then $d \geq 3$.

# Example: The [7,4] Hamming code

We can define the $[7, 4]$ Hamming code by the following parity-check matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Clearly, all the columns of $H$ are non-zero, and they are all distinct. So $d \geq 3$. We can see that $d = 3$ by noting that the first three columns are linearly dependent, since

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

This produces 1110000 as an example of a codeword of weight three.

Since it has minimum distance 3, this code can correct any single error.

# Hamming codes

We can see that a binary $[N, K]$ code will correct any single error if all the columns in its parity-check matrix are non-zero and distinct.

One way to achieve this: Make the $N - K$ bits in successive columns be the binary representations of the integers 1, 2, 3, etc.

This is one way to get a parity-check matrix for a $[7, 4]$ Hamming code:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

When $N$ is a power of two minus one, the columns of $H$ contain binary representations of all non-zero integers up to $2^{N-K} - 1$.

These are called the *Hamming codes*.

# Encoding Hamming codes

By rearranging columns, we can put the parity-check matrix for a Hamming code in systematic form. For the $[7, 4]$ code, we get

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

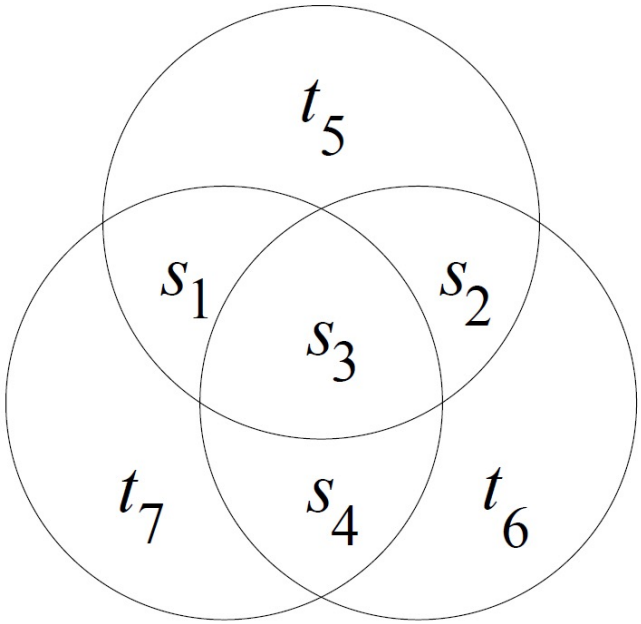
Recall that a systematic parity check matrix of the form $[P^T \mid I_{N-K}]$ goes with a systematic generator matrix of the form $[I_K \mid P]$. We get

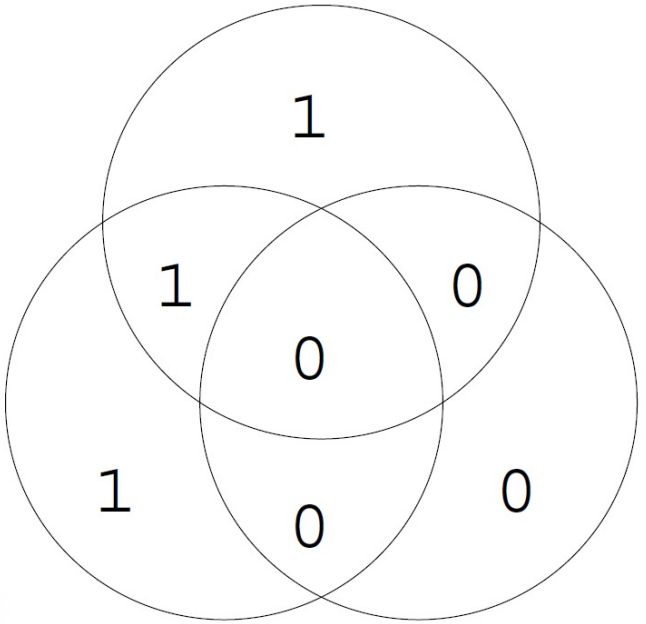
$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

We encode a message block, $\mathbf{s}$, of four bits, by computing $\mathbf{t} = \mathbf{s}G$. The first four bits of $\mathbf{t}$ are the same as $\mathbf{s}$; the remaining three are "check bits". Note: The order of bits may vary depending on how we construct the code.

# Hammingovo kodiranje: Hamming (7,4)

- Coding:




| | |
|---|---|
| Information bits | : $s_1s_2s_3s_4$ |
| Add parity bits | : $t_5t_6t_7$ |
| Codewords | : $s_1s_2s_3s_4$ $t_5t_6t_7$ |

| | |
|---|---|
| Information bits | : 1000 |
| Add | : 101 |
| Codeword | : 1000101 |

Parity: $t_5 = 1$, if the sum $s_1 + s_2 + s_3 =$ *1 mod 2*
$t_5 = 0$, if the sum $s_1 + s_2 + s_3 =$ *0 mod 2*

# Decoding Hamming codes

Consider the non-systematic $[7, 4]$ Hamming code parity-check matrix:

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

Suppose $\mathbf{t}$ is sent, but $\mathbf{r} = \mathbf{t} + \mathbf{n}$ is received. The receiver can compute the *syndrome*, $\mathbf{z} = \mathbf{r}H^T$. Using the fact that $\mathbf{t}H^T = \mathbf{0}$ for a codeword $\mathbf{t}$,

$$\mathbf{z} = \mathbf{r}H^T = (\mathbf{t} + \mathbf{n})H^T = \mathbf{t}H^T + \mathbf{n}H^T = \mathbf{n}H^T$$

If there were no errors, $\mathbf{n} = \mathbf{0}$, so $\mathbf{z} = \mathbf{0}$.

If there is one error, in position $i$, then $\mathbf{n}H^T$ will be the $i$th column of $H$ — which contains the binary representation of the number $i$!

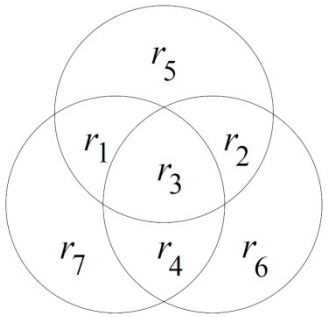
So, to decode, compute the syndrome, $\mathbf{z}$, and if it is non-zero, flip the bit it identifies. If we rearranged $H$ to systematic form, we modify this procedure in corresponding fashion.

# Hammingovo dekodiranje: MLD in sindromi

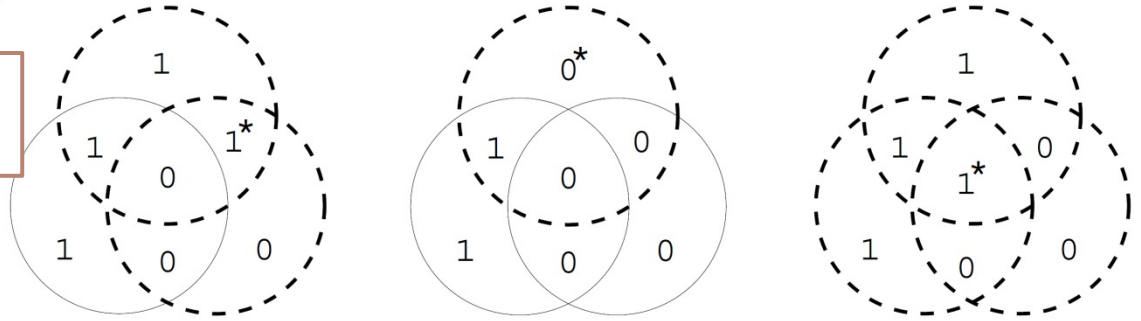
- First decoding option:
  - We compare the received vector after the channel to all the valid codewords in the codebook we are using, in terms of the Hamming distance. The closest codeword to the received vector is the result of decoding. (**Maximum likelyhood decoding**)
  - Problem: computing the distance to all codewords, e.g [30,20] code.

- Second way of decoding: with syndromes
  - Example:
    sent codew. : $\mathbf{t} = 1000101$
    received vector: $\mathbf{r} = 1000101 \oplus 0100000 = 1100101$

    syndrome $\mathbf{z} = (1, 1, 0)$




**disturbed parity check equations – dashed circles**



# Maximum likelyhood decoding

- Again let $\mathbf{C} = \{00000, 10110, 01011, 11101\}$ and $p = 0.1$. If $\mathbf{r} = (11111)$ is received then,

$$\begin{aligned} Pb(\mathbf{r}, 00000) &= (0.1)^5 = 0.00001 \\ Pb(\mathbf{r}, 10110) &= (0.1)^2(0.9)^3 = 0.00729 \\ Pb(\mathbf{r}, 01011) &= (0.1)^2(0.9)^3 = 0.00729 \\ Pb(\mathbf{r}, 11101) &= (0.1)^1(0.9)^4 = 0.06561 \end{aligned}$$

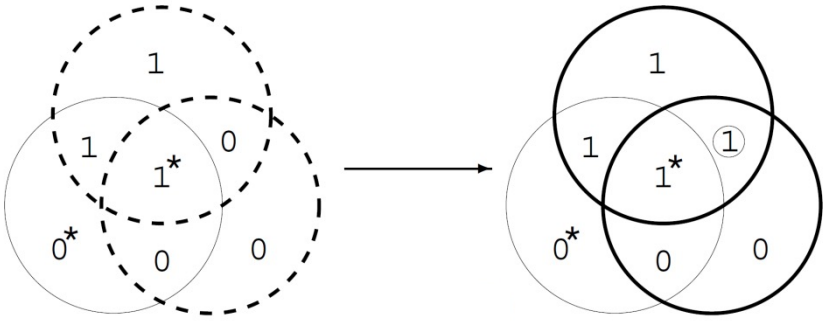
- $Pb(\mathbf{r}, 11101)$ is largest, thus $\mathbf{r}$ is decoded as 11101.

# Hamming decoding: syndroms

- **Main idea**: Construct a table that contains the syndromes if there is a single error, continue filling up $2^{N-K}$ entries. Then, one can correct more than errors if the error pattern is in the table.

| sindrom | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| spremeni bit | nobenega | $r_7$ | $r_6$ | $r_4$ | $r_5$ | $r_1$ | $r_2$ | $r_3$ |

- The same syndroms can be caused by multiple errors but it is quite unlikely
- What if we get 2 errors ?



# Syndrome decoding in general

For any linear code with parity-check matrix $H$, we can find the nearest-neighbor decoding of a received block, $\mathbf{r}$, using the syndrome, $\mathbf{z} = \mathbf{r}H^T$.

We write the received data as $\mathbf{r} = \mathbf{t} + \mathbf{n}$, where $\mathbf{t}$ is the transmitted codeword, and $\mathbf{n}$ is the *error pattern*, so that $\mathbf{z} = \mathbf{n}H^T$.

A nearest-neighbor decoding can be found by finding an error pattern, $\mathbf{n}$, that produces the observed syndrome, and which has the smallest possible weight. Then we decode $\mathbf{r}$ as $\mathbf{r} - \mathbf{n}$.

# Building a syndrome decoding table

We can build a table indexed by the syndrome that gives the error pattern of minimum weight for each syndrome.

We initialize all entries in the table to be empty.

We then consider the non-zero error patterns, $\mathbf{n}$, in some order of non-decreasing weight. For each $\mathbf{n}$, we compute the syndrome, $\mathbf{z} = \mathbf{n}H^T$, and store $\mathbf{n}$ in the entry indexed by $\mathbf{z}$, *provided* this entry is currently empty. We stop when the table has no empty entries.

**Problem:** The size of the table is exponential in the number of check bits — it has $2^{N-K} - 1$ entries for an $[N, K]$ code.

# Hamming sphere's packing bound

We'd like to make the minimum distance as large as possible, or alternatively, have as many codewords as possible for a given distance. There's a limit, however.

Consider a binary code with $d = 3$, which can correct any single error. The "spheres" of radius one around each codeword must be disjoint — so that any single error leaves us closest to the correct decoding.

For codewords of length $N$, each such sphere contains $1+N$ points. If we have $m$ codewords, the total number of points in all spheres will be $m\,(1+N)$, which can't be greater than the total number of points, $2^N$.

So for binary codes that can correct any single error, the number of codewords is limited by

$$m \;\leq\; 2^N/(1+N)$$

# A more general version of the  bound

A binary code of length $N$ that is guaranteed to correct any pattern of up to $t$ errors can't have more than this number of codewords:

$$2^N \left( 1 + \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{t} \right)^{-1}$$

The $k$th term in the brackets is the number of possible patterns of $k$ errors in $N$ bits:

$$\binom{N}{k} = \frac{N!}{k!\,(N-k)!}$$

If the above bound is actually reached, the code is said to be *perfect*. For a perfect code, the disjoint spheres of radius $t$ around codewords cover all points.

Very few perfect codes are known. Usually, we can't find a code with as many codewords as would be allowed by this bound.

# Hamming codes are perfect

For each positive integer $c$, there is a binary Hamming code of length $N = 2^c - 1$ and dimension $K = N - c$. These codes all have minimum distance 3, and hence can correct any single error.

They are also perfect, since

$$2^N/(1+N) \;=\; 2^{2^c-1}/(1+2^c-1) \;=\; 2^{2^c-1-c} \;=\; 2^K$$

which is the number of codewords.

One consequence: A Hamming code can correct any single error, but if there is more than one error, it will not be able to give any indication of a problem — instead, it will "correct" the wrong bit, making things worse.

The *extended Hamming codes* add one more check bit (ie, they add one more row of all 1s to the parity-check matrix). This allows them to detect when two errors have occurred.

# Gilbert Varshamov bound (optional)

The sphere-packing bound is an *upper* limit on how many codewords we can have. There's also a *lower* limit, showing there *is* a code with at least a certain number of codewords.

There is a binary code of length $N$ with minimum distance $d$ that has at least the following number of codewords:

$$2^N \left( 1 + \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{d-1} \right)^{-1}$$

Why? Imagine spheres of radius $d-1$ around codewords in a code with fewer codewords than this. The number of points in each sphere is the sum above in brackets, so the total number of points in these spheres is less than $2^N$. So there's a point outside these spheres where we could add a codeword that is at least $d$ away from any other codeword.

# How good are simple codes

Shannon's noisy coding theorem says we can get the probability of error in decoding a block, $p_B$, arbitrarily close to zero when transmitting at any rate, $R$, below the capacity, $C$ — if we use good codes of large enough length, $N$.

For repetition codes, as $N$ increases, $p_B \to 0$, but $R \to 0$ as well.

For Hamming codes, as $N = 2^c - 1$ increases, $R \to 1$, but $p_B \to 1$ as well, since there's bound to be more than one error in a really big block.

# Practical codes -Coding gain

- Back in 1969 Mariners (Voyagers etc.) were supposed to send pictures from Mars to Earth
- The problem was a thermal noise to send pixels with grey scale of 64 level.


- Redundancy was introduced - 6 bits (64 scale grades) encoded as a 32-bit tuple.

# Mariner story encoding

- Such an encoding could correct up to 7 errors in transmission.
- Correcting errors is **not for free**- we have to send bits 32/6 times faster.
- This means that the total energy per bit is reduced - this causes increased probability of (bit) error !
- Have we overpaid the capability of correcting errors ?
- The answer lies in computing coding gain - if positive then we save energy (reduce the probability of error).

# Error probability in noisy chanels

- Assume that a transmitter has a total energy per bit $E_b$ available. E.g. to send "1" a signal with amplitude $s = \sqrt{E_b}$ is sent and $s = -\sqrt{E_b}$ for "0".
- In presence of AWGN (Additive White Gaussian Noise) the received signal is

$$r = s + n,$$

  $n$ has zero mean and variance $\sigma^2$.
- Hard decision decoding: $r > 0$ "1" sent; "0" otherwise. Then the bit error probability is,

$$p_e = \int_{\sqrt{E_b}}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-y^2}{2\sigma^2}) dy = Q\left(\sqrt{\frac{E_b}{\sigma^2}}\right).$$

# Error probability for Mariner

- Assumption: Each block of 6 bits may be wrong with probability $P_E < 10^{-4}$.
- In case of no coding we need $E_b/\sigma^2 = 17.22$ as,

$$p_e = Q(\sqrt{17.22}) \approx 10^{-4}/6 \text{ and } P_E = 1 - (1 - p_e)^6 \approx 10^{-4}.$$

- Compute $p_e$ for given $P_E$ and get SNR$=E_b/\sigma^2$.
- In Mariner 6 bits encoded as 32 bits, i.e. energy per bits decreases:

$$p'_e = Q\left(\sqrt{\frac{6E_b}{32\sigma^2}}\right)$$

- For given $SNR = 17.22$ $p'_e = 0.036$ – 2000 times larger than $p_e$

# Coding gain for Mariner

- The benefit is in error correction. After decoding 32 bits to 6 bits,

$$P'_E = \sum_{i>7} \binom{32}{i} (p'_e)^i (1-p'_e)^{32-i} \approx 1.4 \cdot 10^{-5}.$$

- Even better results if *soft decoding* is used.
- The use of coding may be viewed as saving the energy ! The code used in Mariner was a $[32, 6]$ Reed-Muller code.
- For Mariner example to get $P'_E = 10^{-4}$ an SNR of 14.83 is required (instead of 17.22).

Definition The ratio between SNR (uncoded) and SNR (coded) for equal error probability after decoding is called the coding gain.

# Zveza med kodiranjem informacije in kanala

- Kaj smo do sedaj pokazali?

  Za kodiranje informacije (stiskanje podatkov) smo ugotovili, da je faktor stiskanja $R > H$, ki še zagotavlja enovito dekodiranje.

  Za kodiranje podatkov za prenos preko kanala (kodiranje kanala) pa smo ugotovili, da moramo kodirati z $R < C$, če hočemo zagotoviti dekodiranje, kjer je maksimalna napaka dekodiranja poljubno majhna.

- Kako moramo podatke zakodirati, da jih lahko 'varno' pošljemo preko kanala?

- In kaj je boljše, če hočemo poslati podatke preko kanala:
  - ali jih je npr. potrebno najprej zakodirati (da jih stisnemo) in potem še enkrat kodirati kodirane podatke, da jih prenesemo brez napak po kanalu z motnjami?
  - ali moramo razmišljati o hkratnem kodiranju podatkov tako, da informacijo stisnemo in jo brez težav prenesemo preko kanala?

# Zveza med kodiranjem informacije in kanala

- Kako moramo podatke zakodirati, da jih lahko ‘varno’ pošljemo preko kanala?
- Primer:
  - Denimo, da želimo poslati neko slovensko besedilo preko binarnega kanala z izgubo.
  - 1 .možnost:
    - Zakodiramo besedilo z najboljšim postopkom stiskanja v neke binarne kode.
    - Te binarne kode potem pošljemo preko kanala.
    - Če imamo napake, jih zelo težko dekodiramo.
  - 2. možnost:
    - Ne kodiramo besedila, ampak pošljemo kar tekst.
    - Tudi če izgubimo polovico črk, bomo lahko še nekako razbrali besedilo.
- Nekako je potrebno zagotoviti, da moramo podatke predstaviti tako, da ustrezajo kanalu.

# Zveza med kodiranjem informacije in kanala

- Za kodiranje informacije (stiskanje podatkov) smo ugotovili, da je faktor stiskanja R > H, ki še zagotavlja enovito dekodiranje.

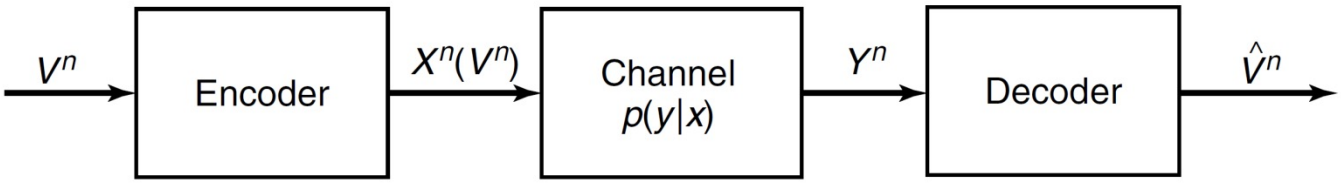
  Za kodiranje podatkov za prenos preko kanala (kodiranje kanala) pa smo ugotovili, da moramo kodirati z R < C, če hočemo zagotoviti dekodiranje, kjer je maksimalna napaka dekodiranja skoraj 0.

- **Izrek o kodiranju vira in kanala:**
  **Pogoj H < C je potreben in zadosten pogoj za zanesljivo prenašanje informacije po kanalih.**

# Zveza med kodiranjem informacije in kanala

Denimo, da imamo vir informcij $V$, ki generira simbole iz abecede $\mathcal{V}$. Denimo, da želimo poslati preko kanala zaporedje simbolov $V^n = V_1, V_2, \ldots, V_n$. Sprejemnik mora rekonstruirati to zaporedje.

Postopek je naslednji: zaporedje $V^n$ zakodiramo z $X^n(V^n)$ in pošljemo preko kanala. Na drugi strani sprejmemo zaporedje $Y^n$ in dekodiramo v zaporedje $\hat{V}^n$. Napaka se zgodi, ko je $\hat{V}^n \neq V^n$.



# Zveza med kodiranjem informacije in kanala

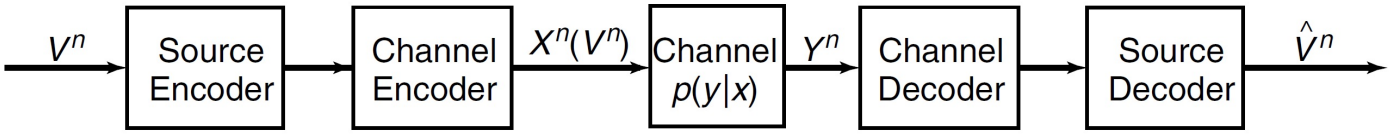
## Izrek o kodiranju vira informacije in kanala

Če je $V^n$ poljubno zaporedje sestavljeno iz simbolov končne abecede $\mathcal{V}$, ki zadoščajo lastnostim AEP in $H(V) < C$, potem obstaja takšno kodiranje, ki zagotavlja poljubno majhno verjetnost napake kodiranja, torej $P(\hat{V}^n \neq V^n) \to 0$, ko gre $n \to \infty$.

Obrat: Za vsak vir, ki zadošča pogoju $H(V) > C$, ne moremo zagotoviti kodiranja z verjetnostjo napake $P(\hat{V}^n \neq V^n)$, ki je poljubno majhna.

# Zveza med kodiranjem informacije in kanala

- Posledica:
  - Izrek pove, da lahko ločeno obravnavamo kodiranje vira informacije in kodiranje informacije za prenašanje podatkov po kanalu.




- Praktična uporaba:
  - uporaba enakega kodiranja za prenašanje podatkov po internetu ne glede na tip podatkov (zvok, slika, tekst, ...)

# Ponovimo: Shannonov izrek o kodiranju kanala, 1948

*Theorem 11:* Let a discrete channel have the capacity $C$ and a discrete source the entropy per second $H$. If $H \leq C$ there exists a coding system such that the output of the source can be transmitted over the channel with an arbitrarily small frequency of errors (or an arbitrarily small equivocation). If $H > C$ it is possible to encode the source so that the equivocation is less than $H - C + \epsilon$ where $\epsilon$ is arbitrarily small. There is no method of encoding which gives an equivocation less than $H - C$.