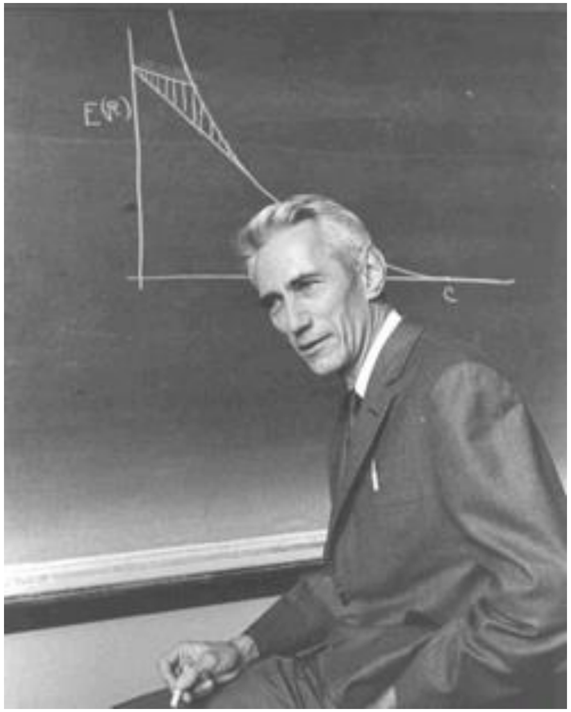
lossless compression

Introduction to Information Theory

- Claude Elwood Shannon, the founder of information theory



C. E. Shannon 1916-2001

C. E. Shannon, the mathematical theory of communication (part 1), *Bell System Technical Journal,* 1948,27(7):379-423
C. E. Shannon, the mathematical theory of communication (part 2), *Bell System Technical Journal,* 1948,27(10): 623-656.

Introduction to Information Theory

In the papers, Shannon answered some important problems

- How can we do the measurement of information
- how does one measure the capacity of a communication channel?
- what are the characteristics of an efficient coding process?
- when the coding is as efficient as possible, at what rate can the channel convey information?
- ……

Introduction to Information Theory

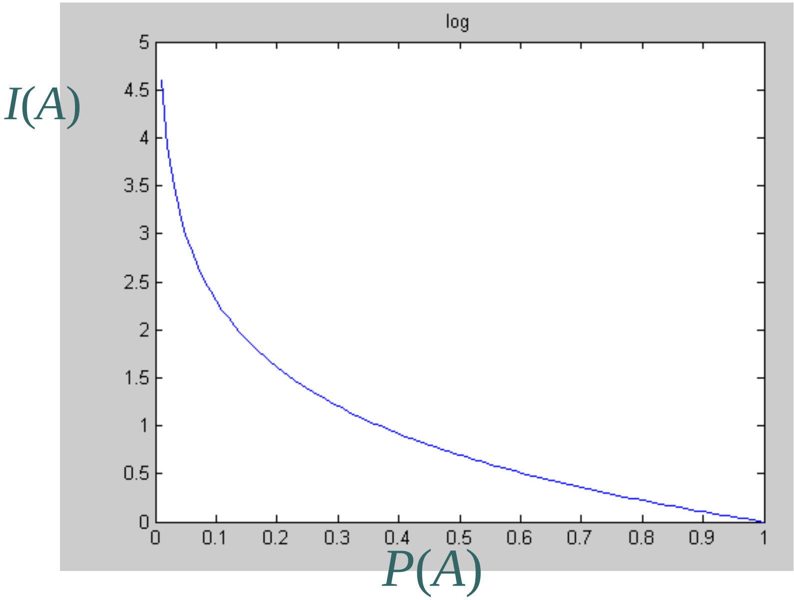
- Self-information
  - If P(A) is the probability that the event A will occur, then the self-information associated with A is given by

$$I(A) = \log_b \frac{1}{P(A)} = -\log_b P(A)$$

- The base of the log function has several values:
  - b=2, the self-information unit is bits, log
  - b=e (means we will take the natural log), the unit is nats 1Nat=1.44bit, ln
  - b=10, the unit is hartleys. 1Hart=3.32bit,lg

$$\log_2 X = 1.443 \log_e X = 3.322 \log_{10} X$$

Self-information



- As shown in the figure
  - I(A) decreases with the increase of P(A). When P(A)=1, I(A)=0; when P(A)=0, I(A)=∞.
  - self information and Probability are inversely related
  - Therefore, if the P(A) is high, the information associated with it is low.
    - A: a dog bites a person B: a person bites a dog
    - I(A) < I(B)

Quality of Self-information

- Suppose A and B are two independent events. The self information associated with the occurrence of both event A and event B is:

$$I(AB) = -\log P(AB)$$

As A and B are independent

$$= -\log P(A) - \log P(B)$$

$$= I(A) + I(B)$$

Example 2.2.1

- Let H and T be the outcomes of flipping a coin.
  - If the coin is fair, then $P(H)=P(T)=1/2$
    - $I(H)=?$, $I(T)=?$
  - If the coin is not fair, the frequency of occurrence of each event is different. Suppose $P(H)=1/8, P(T)=7/8$
    - $I(H)=?$, $I(T)=?$

3bits 0.193bits

Average self-information

- If we have a set of independent events A_i, which are outcomes of some experiment, and $\bigcup A_i = S$, where S is the sample space
- The average self-information associated with the experiment is

$$H = \mathbb{E}\left[I\left(A_i\right)\right] = -\sum P\left(A_i\right)\log P\left(A_i\right)$$

- The average self-information is also called the *entropy*
- Shannon showed that entropy is the best that a lossless compression scheme can do to encode the output of a source
- That is, rate>=entropy for lossless compression

The entropy of a source

- Assume we have alphabet source A. For a general source S with alphabet A={1,2,…,m} that generate a sequence $\{X_1,X_2,\ldots\}$, the entropy is given by

$$H(\mathcal{S}) = \lim_{n\to\infty} \frac{1}{n} G_n$$

- where

$$G_n = -\sum_{i_1=1}^{i_1=m} \sum_{i_2=1}^{i_2=m} \cdots \sum_{i_n=1}^{i_n=m} P(X_1=i_1, X_2=i_2, \ldots, X_n=i_n) \log P(X_1=i_1, X_2=i_2, \ldots, X_n=i_n)$$

- $\{X_1,X_2,\ldots,X_n\}$ is a sequence of length n from the source.
- If each element in the sequence is independent and identically distributed (iid), then we have

$$G_n = -n \sum_{i_1=1}^{i_1=m} P(X_1=i_1) \log P(X_1=i_1)$$

First order entropy

- The entropy will become $H(S) = -\sum P(X_1) \log P(X_1).$

Example

- Consider the sequence：

  1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10
- Assuming the frequency of occurrence of each number is reflected accurately in the number of times it appears in the sequence, then
  - P(i), i=1,2,…,10 ?
  - Entropy of the sequence?
- Can we give a model and compression the sequence?
- If yes, compute the entropy of compressed sequence

Example

- Consider the sequence:
  - 1 2 1 2 3 3 3 3 1 2 3 3 3 3 1 2 3 3 1 2
- If we look at it one symbol at a time, entropy? Total number of bits?
- If we look at it in blocks of two, entropy? Total number of bits?
- What we have learned from this example?
  - P18
- What's the true entropy of source

True entropy

For a general source S with alphabet A={1,2,…m} that generates a sequence $\{X_1,X_2,\ldots\}$, the entropy is given by

$$H(\mathcal{S}) = \lim_{n\to\infty} \frac{1}{n} G_n,$$

$$G_n = -\sum_{i_1=1}^{i_1=m} \sum_{i_2=1}^{i_2=m} \cdots \sum_{im=1}^{i_1=m} P\left(X_1 = a_{i_1}, X_2 = a_{i_2}, \ldots, X_n = a_{i_n}\right) \log P\left(X_1 = a_{i_1}, X_2 = a_{i_2}, \ldots, X_n = a_{i_n}\right)$$

$\{X_1,X_2,\ldots X_n\}$ is a sequence of length n from the source

model

- As shown in above mentioned examples,having a good model for the data can be useful in estimating the entropy/the average number of bits per symbol of the source.
  - Good model:
    - The better the model is, the closer the model matches the aspects of reality.
- There are several approaches to building mathematical models
  - Physical models
  - Probability models
  - Markov models
  - Composite source model

Physical models

- If we know something about the physics of the data generation process, we can use that information to construct a model.
  - For example, in speech-related applications, knowledge about the physics of voice generation can be used to construct a mathematical model. Sampled speech can then be encoded using this model.
- However, the physics of data generation is always too complicated to both understand and use.
  - Or we can obtain a model based on empirical observation of the statistics of the data

Probability/statistics models

- The simplest statistical model → **Ignorance model**
  - Two assumption:
    - each letter is independent
    - each letter occurs with the same probability
- Probability model
  - Two assumption:
    - each letter is independent
    - each letter occurs with the different probability
- Markov model
  - each letter is dependent
    - need to describe the correlation of elements of the data sequence

Entropy of models

- The entropy of probability model

$$H = \mathbb{E}\left[I\left(A_i \right) \right] = -\sum P\left(A_i \right) \log P\left(A_i \right)$$

Entropy of models

Markov models

- One of the most popular ways of representing dependence in the data is through the use of Markov models
- In lossless compression, we use a discrete-time Markov chain.
- Let $\{x_n\}$ be a sequence of observations. And it is said that the sequence follow a *k*th-order Markov model if

$$P\left(x_n \mid x_{n-1}, x_{n-2}, \ldots, x_{n-k}\right) = P\left(x_n \mid x_{n-1}, \ldots., x_{n-k}, \ldots\right),$$

- Knowledge of the past k symbols is equivalent to the knowledge of the entire past history of the process

Markov models

$$P\left(x_n \mid x_{n-1}, x_{n-2}, \ldots, x_{n-k}\right) = P\left(x_n \mid x_{n-1}, \ldots., x_{n-k}, \ldots\right),$$

- The values taken on by the set $\{x_{n-1}, \ldots, x_{n-k}\}$ are called the states of the process.
- The most commonly used Markov model is the first-order Markov model,

$$P\left(x_n \mid x_{n-1}\right) = P\left(x_n \mid x_{n-1}, x_{n-2}, \ldots\right)$$

- These two equations indicate the existence of dependence between samples. However they do not describe the form of the dependence.

Markov models

- The dependence: linear and nonlinear
- linear dependence
  - we could view the data sequence as the output of a linear filter driven by white noise.

$$x_n = \rho x_{n-1} + \varepsilon_n$$

ρ is a constant

where ε_n is a white noise process.

  - This model is often used when developing coding algorithms for speech and images.

Markov models

- Nonlinear dependence
  - The dependence can be described by the transition probability
  - The Markov model can be represented by the state transition diagram
  - The entropy of a finite state process with states S_i is simply the average value of the entropy at each state:

$$H = \sum_{i=1}^{M} P(S_i) H(S_i)$$

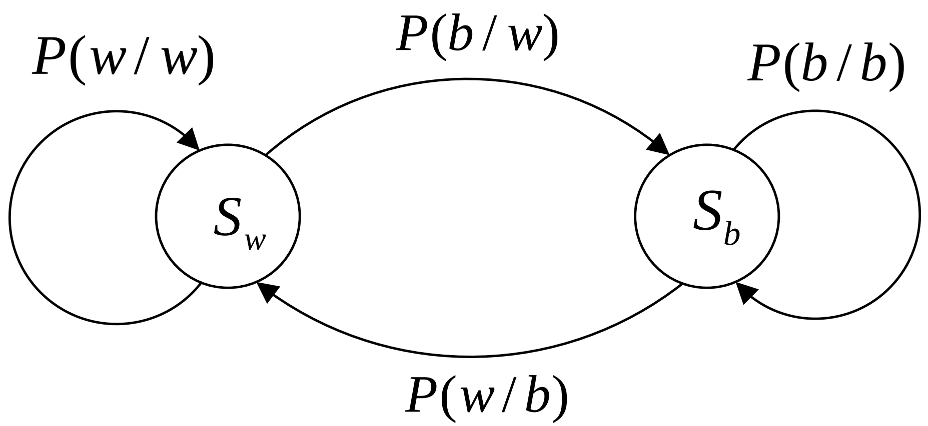
conditional entropy

Markov model

- Consider a binary image. The image has only two types of pixel: white and black.
- We know that the appearance of a white pixel depends to some extent on whether the current pixel is white or black.
- Therefore we can model the pixel process as a discrete time Markov chain.
- Define two states： S_b & S_w
- the probility of each state： $P(S_b)$ $P(S_w)$
- The transition probabilities
  - $P(b|b)$, $P(b|w)$
  - $P(w|b)$, $P(w|w)$

Markov model

the two state Markov model for binary images



$$H(S_w) = -P(b/w)\log(b/w) - P(w/w)\log(w/w)$$

$$H(S_b) = -P(w/b)\log(w/b) - P(b/b)\log(b/b)$$

$$P(w/w) = 1 - P(b/w), P(b/b) = 1 - P(w/b)$$

$$H = P(S_b)H(S_b) + P(S_w)H(S_w)$$

$$P(S_w) = \frac{P(w/b)}{P(b/w)+P(w/b)} = 1 - P(S_b)$$

Markov model-example

- assume

$$P(w/w) = 0.99 \quad P(b/w) = 0.01 \quad P(b/b) = 0.7 \quad P(w/b) = 0.3$$

- As for probability model and iid assumption

$$H = -\frac{30}{31}\log\frac{30}{31} - \frac{1}{31}\log\frac{1}{31} = 0.206\ bits$$

- Now using Markov model

$$H(S_b) = -0.3\log 0.3 - 0.7\log 0.7 = 0.881\ bits$$

$$H(S_w) = -0.01\log 0.01 - 0.99\log 0.99 = 0.081\ bits$$

$$H_{Markov} = \frac{30}{31}0.081 + \frac{1}{31}0.881 = 0.107$$

Markov models in text compression

- Markov models are particularly useful in text compression, where the probability of the next letter is heavily influenced by the preceding letters.

Markov models in text compression

- In English， we consider 26 letters and 1 space, total 27 symbols.
- （1）ignorance model (Independent, equal probability）：

$$H_0 = \log 27 = 4.76(bits / symbol)$$

  - Output： XFMOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

C.E.Shannon. Prediction and Entropy of printed English. Bell System Technical Journal, 30:5-064, January 1951

Markov models in text compression

- （2） probability model (Independent, different probability)

Probabilities of 27 symbols

symbol	probability		symbol	probability		symbol	probability
space	0.2		S	0.052		Y,W	0.012
E	0.105		H	0.047		G	0.011
T	0.072		D	0.035		B	0.0105
O	0.0654		L	0.029		V	0.008
A	0.063		C	0.023		K	0.003
N	0.059		F,U	0.0225		X	0.002
I	0.055		M	0.021		J,Q	0.001
R	0.054		P	0.0175		Z	0.001

Markov models in text compression

- Entropy of probability model

$$H_1 = -\sum_{i=1}^{27} p(e_i)\log_2 p(e_i) = 4.03(\text{比特}/\text{字符})$$

- Output：
  - **OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL**

Markov models in text compression

- To make ‘real english’, we consider the relationship between letters, and model the English as 1th order Markov , 2th order Markov,…, the entropy is

$$H_2 = 3.32(\text{bit/sym})$$
$$H_3 = 3.01(\text{bit/sym})$$
$$H_4 = 2.8(\text{bit/sym})$$

Markov models in text compression

- **Output of 2th order Markov Model:**
  - **ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE**
- **The words with length ≤3 are meaning, while those with length >3 are not real words.**
- **Why?**

Markov models in text compression

- **Output of 3th order Markov Model:**
  - THE GENERATED JOB PROVIDUAL BETTER TRAND THE DISPLAYED CODE ABOVERY UPONDULTS WELL THE CODERST IN THESTICAL IT DO HOCK BOTHE MERG INSTATES CONS
- **Output of 4th order Markov Model:**
  - THE GENERATED JOB PROVIDUAL BETTER TRAND THE DISPLAYED CODE ABOVERY UPONDULTS WELL THE CODERST IN THESTICAL IT DO HOCK BOTHE MERG INSTATES CONS

Composite source model

- In many applications, it is not easy to use one signal model the source.
- In this cases, we can define a composite source, which can be viewed as a combination or composition of several sources.
- As shown in Fig 2.3, in this composite source model, there are n sources with a switch, only one source is active at any given time.
- A composite source can be represented as a number of individual sources S_i, each with its own model M_i, and a switch that selects a source S_i with probability P_i.

Coding

- When we talk about coding, we always mean code the symbol or letter with binary sequences.
  - Code: the set of binary sequences
  - Codewords: the individual members of the set
  - Alphabet
    - ASCII code, A->1000001,a->1000011 etc.

Code words

$$\underbrace{W}_{\text{code}} = \underbrace{\{w_1, w_2, \ldots, w_n\}}_{\text{Code words}}$$

Coding

- Some Codes :
  - fixed length code
    - e.g. a = 00, b = 01, c = 10, d = 11
  - variable-length code: assigns a bit string (codeword) of variable length to every message value
    - e.g. a = 1, b = 01, c = 101, d = 011
  - uniquely decodable code: is a variable length code in which bit strings can always be uniquely decomposed into its codewords.
    - What if you get the sequence of bits 1011 ?
    - a = 1, b = 01, c = 101, d = 011 aba, ca, ad? => Not uniquely decodable
    - a=1,b=01,c=000,d=001 aba =>uniquely decodable
  - Instantaneous code: the decoder knows the moments a code is complete (the code can be decoded at the moment it is complete)

Coding

- The average length l for each code:

$$l = \sum P(a_i)n(a_i) \quad \textbf{bits/symbol}$$

$n(a_i)$ is the code length for letter a_i

Coding

- Alphabet $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$
  - $P(a_1) = 1/2$, $P(a_2) = 1/4$, $P(a_3) = P(a_4) = 1/8$
  - H = 1.75 bits
  - $l(a_i)$ = length (codeword(a_i)), i=1...4
  - $\boldsymbol{l} = \Sigma_{i=1..4} P(a_i)\, l(a_i)$
- Codes：

	probability	Code1	Code2	Code3	Code4	Code5
a_1	0.500	0	0	0	0	00
a_2	0.250	0	1	10	01	01
a_3	0.125	1	00	110	011	10
a_4	0.125	10	11	111	0111	11
l		1.125	1.250	1.750	1.875	2

	probability	Code1	Code2	Code3	Code4	Code5
a_1	0.500	0	0	0	0	00
a_2	0.250	0	1	10	01	01
a_3	0.125	1	00	110	011	10
a_4	0.125	10	11	111	0111	11
l		1.125	1.250	1.750	1.875	2

- **Code 1：Code a_1 ＝Code a_2 ==> decode('00') = ???**
- **Code 2： Code $a_1 \neq$ Code a_2 , decode('00'/'11') = ???**
- **Code 3：uniquely decodable， Instantaneous code**
- **Code 4： uniquely decodable，near instantaneous code（have to wait to decode till the beginning of the next codeword）**
- **Code 5: same length and different codewords, uniquely decodable**
- **Which code is the best one?**

Uniquely decodable code & instantaneous code

- Unique decodability !=instantaneous decoding
- Instantaneous codes must be unique decodable codes, unique decodable codes doesn’t have to be instantaneous codes
  - **Uniquely decodable codes**

Instantaneous codes

Non-instantaneous codes

Test for unique decodability

Code 6

Letter	Code word
a1	0
a2	01
a3	10

- Is code 6 unique decodable codes?
- The sequency:0101010101010101010
- Decode:a2 a2 a2…a1(eight a2s) or

  a1 a3 a3……(eight a3s)
- So code 6 is not unique decodable.
- How to determine whether the codes are uniquely decodable?

Test for unique decodability

- It is not immediately evident whether the code is uniquely decodable or not.
- Dangling suffix
  - Suppose we have two binary codewords *a* and *b*, where *a* is *k* bits long, *b* is *n* bits long, and $k<b$. If the first *k* bits of *b* are identical to *a*, then *a* is called a *prefix* of *b*. The last *n-k* bits of *b* are called the *dangling suffix*.
  - For example, if *a*=010 and *b*=01011
    - Prefix? **010**
    - Dangling suffix? **11**

Test for unique decodability

- Test for unique decodability
  - 1.Examine all pairs of codewords to see if any codeword is a prefix of another codeword
  - 2.If yes, add the dangling suffix to the code list
  - 3.repeat 2,until one of the following two things happens:
    - (1) you get a dangling suffix that is a codeword, that is, the dangling suffix is itself a codeword;
    - (2) there are no more unique dangling suffixes.
  - 4.If you get the first outcome, the code is not uniquely decodable. However if you get the second outcome, the code is uniquely decodable.

Example 2.4.1

Step	Description	Example
1	Construct a list of all codewords in a code	*[0,01, 11]*
2	Examine all pairs of codewords to see if any codeword is a prefix of another codeword.	*Code word 0 is a prefix of 01*
3	If any pair exist, add the dangling prefix to the list (if not added already in previous iterations)	*[0,01, 11, 1]*
4	Repeat step 2 – 3 until any of these condition occur: a) A Dangling suffix that is a codeword is obtained: **RESULT:** *Code NOT Uniquely Decodable* b) No more Unique Dangling Suffixes: **RESULT:** *Code is Uniquely Decodable*	✓ *1 is a prefix of 11* ✓ *Dangling suffix 1 already in the list* ✓ *No more prefix pairs* ***RESULT:*** ***<u>Code is Uniquely</u>***

Example 2.4.1

- Consider the codewords {0,01,11}
  - The codeword 0 is the prefix for the code 01, then the dangling suffix is 1. There are no other prefix;
  - Add the suffix 1 into the codeword list, we obtain a new codeword list {0,01,11,1}.;
  - examine the new list (1 and 11, the suffix is 1, which is already in the list). There are no other pairs that would generate a dangling suffix, so there are no more unique dangling suffix in the augmented list;
  - We get the second outcome. Therefore, it is uniquely decodable.

Example 2.4.2

- Consider the codewords {0,01,10}
  - The codeword 0 is a prefix for codeword 01, the dangling suffix is 1;
  - Add the suffix 1in the list and we obtain a new list {0,01,10,1};
  - Examine the new list, and we find that 1 is a prefix for 10, the dangling suffix for this pair is 0, which is the codeword for a1;
  - so we get the first outcome;
  - Therefore, it is not uniquely decodable.

Prefix codes

- One type of code in which we will never face the possibility of the first outcome must be uniquely decodable.
- Prefix codes are such code.
- Prefix codes: a variable length code in which no codeword is a prefix to another codeword.
  - e.g., a = 0, b = 110, c = 111, d = 10

Prefix codes

- Some Prefix Codes for Integers

n	Binary	Unary	Gamma
1	..001	0	0|
2	..010	10	10|0
3	..011	110	10|1
4	..100	1110	110|00
5	..101	11110	110|01
6	..110	111110	110|10

other fixed prefix codes:
Golomb, phased-binary, subexponential, ...

Prefix codes

- Check if a code is a prefix code: drawing the rooted binary tree corresponding to the code.
- How to draw a rooted binary tree?

Draw a rooted binary tree

- Start form a single node (the root node) and has a maximum of two possible branches at each node;
- One of these branches corresponds to a 1 and the other branch corresponds to a 0;
- In this book, the left branch corresponds to a 0 and the right branch corresponds to a 1.

Example of binary tree

- Code 2:{0,1,00,11},code 3:{0,10,110,111},
- Code4:{0,01,011,0111}
  - Draw binary tree(P31,figure 2.4)
  - The trees have two kinds of nodes
    - internal nodes: nodes that give rise to other nodes
    - external nodes (leaves): nodes that do not give rise to other nodes
  - The code for any symbol can be obtained by traversing the tree from the root to the external node, each branch on the way contributes a bit to the codeword: 0 for each left branch and 1 for each right branch.
  - In a prefix code, the codeword only associated with the external nodes.
    - Code 4 is not a prefix code, code 3 is prefix code.

Conclusion of prefix codes

- For any nonprefix uniquely decodable code, we can always find a prefix code with the same codeword lengths.

Kraft-Mcmillan inequality

- If a code is UD, then

$$K(\mathcal{C}) = \sum_{i=1}^{N} 2^{-l_i} \leq 1$$

- Equality is complete code

Algorithmic information theory

- *Kolmogorov* complexity /2.5/
- MDL principle /2.6/

homework

- P38-39,3,5,7