Information Theory. 2nd Chapter Problems

Boris Kudryashov

ITMO University

October 24, 2016

Problems

1. Construct a Huffman code for the source with probability distribution:
   $(0,3, 0,25, 0,15, 0,1, 0,1, 0,05, 0,05)$
   Compare the average codeword length with the entropy.

Problems

1. Construct a Huffman code for the source with probability distribution:
   $(0,3, 0,25, 0,15, 0,1, 0,1, 0,05, 0,05)$
   Compare the average codeword length with the entropy.
2. Consider Discrete Memoryless Source (DMS) $X = \{a, b, c\}$ with probability distribution:
   1. $p(a) = 1/2, p(b) = 1/4, p(c) = 1/4$;
   2. $p(a) = p(b) = p(c) = 1/3$.

   Find Huffman codes for X, X^2, X^3. Compare code rate with the entropy of the source.

Problems

3 Analyse behaviour of the Huffman code rate as a function of length n of encoded blocks for the Binary Memoryless Source (BMS) with probability of 1 equal to 0.1.

Problems

4 *Optimal coding for uniform distributions* Find a Huffman code for source, which chooses letters from alphabet of size M with uniform distribution. Calculate the average length of codewords and redundancy as a function of M, derive upper and lower bounds of redundancy.

4 (hint) *Intermediate steps* Huffman code for this source contains $D = 2 \cdot 2^{\lfloor \log M \rfloor} - M$ words of length $\lfloor \log M \rfloor$ and $M - D$ words of length $\lfloor \log M \rfloor + 1$. Average length of codewords and redundancy are equal to

$$\bar{l} = \lfloor \log M \rfloor + 2 - \frac{2}{M} 2^{\lfloor \log M \rfloor},$$

$$r = \bar{l} - \log M = 2 - d - 2 \cdot 2^{-d},$$

where $d = \log M - \lfloor \log M \rfloor$, $d \in [0, 1)$. Differentiation with respect to d shows, that maximum of redundancy is achieved when $d = 1 - \log \log e$ and is equal to $1 + \log \log e - \log e \approx 0,0861$.

5 Consider Markov chain with the probability transition matrix:

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 1/4 & 1/4 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}.$$

Calculate $H(X)$, $H_n(X)$, $n = 1, 2, ...$, $H(X/X^n)$, $n = 1, 2, ...$, assuming that the initial distribution on letters is the stationary distribution. Find code rate for coding ensembles X and X^2 using Huffman coding. Suggest such a method for encoding, that code rate is equal to the information rate of the source.

6 *Non-uniform series-length coding* Sequence at the output of binary source is uniquely represented in the form of concatenation of subsequences $1, 01, ..., 0^{L-1}1, 0^{L}$ for some integer R. Then a Huffman code is used for encoding indices of subsequences from this set. For a BMS with probability of 1 equal to p=0,1, analyse the dependence between code rate and the value L .
Code rate in this case is a value $R = \bar{n}/\tau$, where τ – average length of sequence to be encoded; $\bar{n}$ – average length of codewords.

7 *Variable-to-Variable (VV) codiing. Tunstall code.*
Recall that the redundancy of the Huffman code is determined by the maximum probability of letters.
Let $X = p(x)$ and $x_0 \in X$ is a most probable letter. Introduce a new ensemle X_1, which consists of all letters from $X \setminus x_0$ and of all pairs (x_0, x), $x \in X$. Probabilities of pairs of letters are calculated like products of probabilities of corresponding letters. Such extension of the alphabet can be continued, and on each next step, the maximal probability of letters of the extended alphabet becomes smaller. Thus the redundancy of code applied to the extended alphabet also becomes smaller.
Compare this way of encoding to series-length coding, considered in previous problem for a random binary ensemble

Problems

8 Construct Shannon code of the source from task 1. Compare its average length with the average length of Huffman code codewords and with entropy of the source.

Problems

8 Construct Shannon code of the source from task 1. Compare its average length with the average length of Huffman code codewords and with entropy of the source.

9 Construct Gilbert-Moore code of the source from task 1. Compare its average length with the average length of Huffman code codewords and with entropy of the source.

Problems

10 Use arithmetic coding to encode the sequence 01001 from the BMS with probability of 1 is 0.4
Compare the length of code sequence with the amount of information of the sequence.
Compare the length of code sequence with the total length of code sequence assuming independent encoding of characters with Shannon code.

Problems

10 Use arithmetic coding to encode the sequence 01001 from the BMS with probability of 1 is 0.4 Compare the length of code sequence with the amount of information of the sequence. Compare the length of code sequence with the total length of code sequence assuming independent encoding of characters with Shannon code.

11 Explain, why completing step of encoding in the program (p. 87, pic. 2.16 in the book) is equivalent to conversion of the codeword of Shannon code to codeword Gilbert-Moore code.

Problems

12 What changes should be done in the algorithms and programs of arithmetic coding and decoding, to apply them to source, described by Markov chain.

Problems

12 What changes should be done in the algorithms and programs of arithmetic coding and decoding, to apply them to source, described by Markov chain.

13 In programs at the pictures (p. 87, pic. 2.16) and (p. 88, pic. 2.17) in the book appear division operations which did not exist in algorithms of arithmetic coding and decoding. Why? Is it possible to avoid divisions here?

Problems

```
function y=int_arithm_encoder(x,q);
% x is input data sequence,
% q is cumulative distribution (model)
% y is binary output sequence

% Constants
k=16;
R4=2^(k-2); R2=R4*2; R34=R2+R4;% half,quarter,e
R=2*R2;                        % Precision

% Initialization
Low=0;          % Low
High=R-1;       % High
btf=0;          % Bits to Follow
y=[ ];          % code sequence

% Encoding
for i=1:length(x);
   Range=High-Low+1;
   High=Low+fix(Range*q(x(i)+1)/q(m))-1;
   Low=Low+fix(Range*q(x(i))/q(m));

   % Normalization
   while 1
      if High<R2
         y=[y 0 ones(1,btf)]; btf=0;
         High=High*2+1; Low=Low*2;
      else
         if Low>=R2
            y=[y 1 zeros(1,btf)]; btf=0;
            High=High*2-R+1; Low=Low*2-R;
         else
            if Low>=R4  &  High<R34
               High=2*High-R2+1; Low=2*Low-R2;
               btf=btf+1;
            else
               break;
            end;
         end;
      end;
   end; % while
end; % for

% Completing
if Low<R4
   y=[y 0 ones(1,btf+1)];
else
   y=[y 1 zeros(1,btf+1)];
end;
```

```
function x=int_arithm_decoder(y,q,n);
% y is binary encoded data sequence,
% q is cumulative distribution (model)
% x is output sequence
% n is number of messages to decode

% Constants
k=16; R4=2^(k-2); R2=R4*2; R34=R2+R4; R=2*R2;
m=length(q);

% Start decoding. Reading first k bits
Value=0; y=[y zeros(1,k)];
for ib=1:k
   Value=2*Value+y(ib);
end;

% Initialization
Low=0;  High=R-1;

% Decoding
for j=1:n
   Range=High-Low+1;
   aux=fix(( (Value-Low+1)*q(m)-1)/Range);
   i=1; % message index
   while q(i+1)<=aux, i=i+1; end;
   x(j)=i;
   High=Low+fix(Range*q(i+1)/q(m))-1;
   Low=Low+fix(Range*q(i)/q(m));

   % Normalization
   while 1
      if High<R2
         High=High*2+1; Low=Low*2;
         ib=ib+1;
         Value = 2*Value+y(ib);
      else
         if Low>=R2
            High=High*2-R+1; Low=Low*2-R;
            ib=ib+1;
            Value = 2*Value-R+y(ib);
         else
            if Low>=R4  &  High<R34
               High=2*High-R2+1;  Low=2*Low-R2;
               ib=ib+1;
               Value = 2*Value-R2+y(ib);
            else
               break;
            end;
         end;
      end
   end; % while
end; % for
```