**mymain.m**

mysource

myencoder

mydecoder

load("sourcevars.mat")

load('Lmin.mat')

fprintf("Entropy H[X/S] of dependent source=%f\n",entropy);

fprintf("Lmin with window elements included=%f\n",length(totalencodestr)/length(source));

fprintf("Lmin without window elements included=%f\n",Lmin);

**mysource.m**

rng(1);

alphabetset = 'a':char('a'+3);%param to change

fixedlengthentropy=2;%param to change

%numOfStates=length(alphabetset)+1;

numOfStates=4;%param to change

%creates a markov source with random transition matrix

mc=mcmix(numOfStates,Zeros=2\*numOfStates);

fprintf("Is my markov chain ergodic=%d\n",isergodic(mc));%tells whether the generated chain is ergodic or not

requiredmatchlength=9;%param to change%typical match length

windowsize=2^(requiredmatchlength\*fixedlengthentropy);%approx typical set size

numofLetters=windowsize\*8;%source length is 8 times the window size

X = simulate(mc,numofLetters);

mc.P;%(i,j) element specifies prob from ith state to jth state

xFix = asymptotics(mc);%each state probability q(s)

%H[X/S] calculation

entropy=0;

for j=1:numOfStates

H=0;

symbols = (1:numOfStates); % Alphabet vector

prob = mc.P(j,:); % alphabet probability vector at a given state

prob=prob(prob~=0);

%[dict,avglen] = huffmandict((1:length(prob)),prob);

%entropyPerState=avglen;%E[L]

entropyPerState=log2(prob)\*transpose(-prob);%H[X/s]

entropy=entropy+xFix(j)\*entropyPerState;% H[X/S]

end

%disp(entropy);

%creates a random symbol for each transition of state

%T=alphabetset(randi([1 length(alphabetset)],numOfStates,numOfStates))

%creates a symbol for each transition of state

T=[ 'abcd'

'abcd'

'abcd'

'abcd'];

source='';

for i=2:length(X)

%source=strcat(source,T(X(i-1),X(i)));%alphabetset(X(i)));

source(i-1)=T(X(i-1),X(i));

end

%source=num2str(transpose(X));

%source=source(~isspace(source));

%disp(source);

save('sourcevars.mat','source','entropy','alphabetset','mc','X','xFix','windowsize','T');

%simplot(mc,X);

%figure;

%graphplot(mc,'LabelEdges',true,'ColorEdges',true,'ColorNodes',true);

**myencoder.m**

clear

load('sourcevars.mat');

windowstartptr=1;

windowendptr=windowsize;

encodeptr=1;

totalencodestr='';

%source string has to be greater than window size

assert(strlength(source)>windowsize,'source string length has to be greater than window size')

fprintf("LZ encoding started\n");

%filling the initial window for rcver

for i=1:windowsize

%matchedlength=1;

%ntobin=dec2bin(matchedlength,2\*floor(log2(matchedlength))+1);

matchstr=source(encodeptr);

utobin=dec2bin(strfind(alphabetset,source(encodeptr))-1,ceil(log2(strlength(alphabetset))));

%torcver=strcat(ntobin,utobin);

torcver=utobin;

%totalencodestr=strcat(totalencodestr,torcver);

totalencodestr=append(totalencodestr,torcver);

encodeptr=encodeptr+1;

end

initialprefixlength=strlength(totalencodestr);

while encodeptr~=strlength(source)+1

hasMatchfound=0;

matchedlength=1;%length of the match

matchindex=1;%where match has occured in the window

matchstr='';%string being matched

%search by increasing the length of match str till window size

for complength=2:windowsize

out=[];%set of match indexes

if encodeptr+complength-1<=strlength(source)

%virtually window length will be till last but one index of match str

out=strfind(source(windowstartptr:windowendptr+complength-1),source(encodeptr:encodeptr+complength-1));

else

break;%implies no source str is present beyond this length for matching

end

if isempty(out)==1

break;%implies no match for this in window

else

hasMatchfound=1;%record the matched str details and proceed for a bigger length

matchedlength=complength;

if ismatrix(out)==true%implies multiple matches are found

matchindex=out(length(out));%taking the length closer to the window's end

else%implies single match is found

matchindex=out;

end

matchstr=source(encodeptr:encodeptr+complength-1);

end

end

if hasMatchfound

[torcver,windowstartptr,windowendptr]=encodeInp(matchedlength,matchindex,windowstartptr,windowsize,source,alphabetset,encodeptr);

else

[torcver,windowstartptr,windowendptr]=encodeInp(1,0,windowstartptr,windowsize,source,alphabetset,encodeptr);

end

%{

if hasMatchfound==1

matchstr;

ntobin=dec2bin(matchedlength,2\*(floor(log2(matchedlength)))+1);%matched length is n %unary binary code

matchdistfromencptr=windowsize- matchindex+1;%calculating u

if matchdistfromencptr==windowsize

utobin=dec2bin(0,ceil(log2(windowsize)));%assigning zero to the highest value

else

utobin=dec2bin(matchdistfromencptr,ceil(log2(windowsize)));%fixed length encoding

end

torcver=strcat(ntobin,utobin);

%sliding the window

windowstartptr=windowstartptr+matchedlength;

if strlength(source)>=windowstartptr+windowsize-1

windowendptr=windowstartptr+windowsize-1;

else

windowendptr=strlength(source);

end

else%encoding using fixed length coding

complength=1;

matchedlength=1;%setting n to 1

matchstr=source(encodeptr);

ntobin=dec2bin(matchedlength,2\*(floor(log2(matchedlength)))+1);%fixed length encoding

utobin=dec2bin(strfind(alphabetset,source(encodeptr))-1,ceil(log2(strlength(alphabetset))));%fixed length encoding

torcver=strcat(ntobin,utobin);

windowstartptr=windowstartptr+complength;

if strlength(source)>=windowstartptr+windowsize-1

windowendptr=windowstartptr+windowsize-1;

else

windowendptr=strlength(source);

end

end

%}

encodeptr=encodeptr+matchedlength;

%totalencodestr=strcat(totalencodestr,torcver);

totalencodestr=append(totalencodestr,torcver);

%Lmin=(strlength(totalencodestr))/(encodeptr)

end

torcver=totalencodestr;

%disp(totalencodestr)

Lmin=(strlength(torcver)-initialprefixlength)/(strlength(source)-windowsize);

%disp(Lmin);

save('Lmin.mat','Lmin','totalencodestr');

%this function converts n and u values in binary and updates window

%parameters

function [torcver,windowstartptr,windowendptr]=encodeInp(matchedlength,matchindex,windowstartptr,windowsize,source,alphabetset,encodeptr)

%matchedlength

ntobin=dec2bin(matchedlength,2\*(floor(log2(matchedlength)))+1);%matched length is n %unary binary code

matchdistfromencptr=windowsize- matchindex+1;%calculating u

if matchedlength==1

utobin=dec2bin(strfind(alphabetset,source(encodeptr))-1,ceil(log2(strlength(alphabetset))));%fixed length encoding

else

if matchdistfromencptr==windowsize

utobin=dec2bin(0,ceil(log2(windowsize)));%assigning zero to the highest value

else

utobin=dec2bin(matchdistfromencptr,ceil(log2(windowsize)));%fixed length encoding

end

end

%torcver=strcat(ntobin,utobin);

torcver=append(ntobin,utobin);

%sliding the window

windowstartptr=windowstartptr+matchedlength;

if strlength(source)>=windowstartptr+windowsize-1

windowendptr=windowstartptr+windowsize-1;

else

windowendptr=strlength(source);

end

end

**mydecoder.m**

clearvars -except windowsize source torcver alphabetset Lmin

%filling rcver window

%rcverwindow=source(1:windowsize);

assert(strlength(source)~=0,'Generate source string')

assert(strlength(torcver)~=0,'Run encoder')

fprintf("LZ decoding started\n");

rcverwindow='';

for i=1:windowsize

currentparselength=0;

rcvedstr=alphabetset(bin2dec(torcver(1:ceil(log2(strlength(alphabetset)))))+1);

currentparselength=currentparselength+ceil(log2(strlength(alphabetset)));

rcverwindow=append(rcverwindow,rcvedstr);

rcvedstr=[];

torcver(1:currentparselength)=[];

end

totalrcvd=rcverwindow;

mlen=0;

totallength=0;

abc=0;

while ~isempty(torcver)

currentparselength=0;

k=strfind(torcver,'1');%finding the first 1 in unary binary code

e=k(1);%consider the nearest match for parsing%also indicates no. of zeros present in unary binary code

%parsing and decoding a match

if e==1%rcved match of length 1

currentparselength=currentparselength+1;%as n is 1

%decode based on fixed length coding from alphabet set

rcvedstr=alphabetset(bin2dec(torcver(2:ceil(log2(strlength(alphabetset)))+1))+1);

%parsed this length from encoded string

currentparselength=currentparselength+ceil(log2(strlength(alphabetset)));

else%decode n and u based on the location first 1 of n

%extending parse length till unary zeros

currentparselength=currentparselength+e-1;

%decoding n

lenatrcver=bin2dec(torcver(e:e\*2-1));

%k=strfind(torcver(e\*2:strlength(torcver)),'1');

%r=k(1);

%parsing and decoding u based on windowing size length

distatrcver=bin2dec(torcver(e\*2:e\*2+ceil(log2(windowsize))-1));

if distatrcver==0

distatrcver=windowsize;

end

%decoding the orginal str based on decoded n and u

if distatrcver<lenatrcver%if u < n implies that part of the string is sent and needs to be multiplied

rcvedstrmain=rcverwindow(windowsize-distatrcver+1:windowsize);

remlen=lenatrcver;

while remlen~=0

if remlen>=strlength(rcvedstrmain)

%add string present in window until the length to be

%added is less than u

rcvedstr=strcat(rcvedstr,rcvedstrmain);

remlen=remlen-strlength(rcvedstrmain);

else

%if length to be added is less than u only part of the

%string found in window has to be added

rcvedstr=strcat(rcvedstr,rcvedstrmain(1:remlen));

remlen=0;

end

end

rcvedstr;

else

rcvedstr=rcverwindow(windowsize-distatrcver+1:windowsize-distatrcver+lenatrcver);

end

currentparselength=currentparselength+ceil(log2(windowsize))+e;%extending parse to n,u pair

end

%adding the newly decoded str to the window

rcverwindow=append(rcverwindow,rcvedstr);

rcverwindow(1:strlength(rcvedstr))=[];

%adding to output string

totalrcvd=append(totalrcvd,rcvedstr);

%totallength=totallength+strlength(rcvedstr);

%abc=abc+1;

%mlen=totallength/abc

rcvedstr=[];

%deleting the parsed part from the binary string

torcver(1:currentparselength)=[];

end

totalrcvd;

isoutmatched=strcmp(totalrcvd,source);%check if decoded output matches with source

fprintf("\nIs decoded string matched with source=%d\n",isoutmatched);

References:

1.<https://in.mathworks.com/help/econ/mcmix.html>

**ReadMe**

This code is run based on 4 files mymain.m mysource.m myencoder.m mydecoder.m

When mymain.m is run it will call the other files also from it.