Lab1, Task 4

0 Votes

question posted 3 days ago by bcottier

Lab1, Task 4 I think I am making determining the number of white_runs and the number of black_runs more difficult than it must be. We are already provided with the number of runs of ones and zeros and we must determine if they are runs of zeros or runs of ones. Can we assume the data starts from zero? Can someone make a suggestion? Thanks

This post is visible to everyone.

5 responses

Add a Response

clarenceho

3 days ago

0 Votes

To my understanding, Matlab indexing starts with 1.

We know that the array has alternating white and black length. There is an easy way to extract alternating elements, e.g.:

```
>> list = {'white', 'black', 'white',
'black', 'white'};
>> disp(list(1:2:end))
    'white'    'white'    'white'

>> disp(list(2:2:end))
    'black'    'black'
```

From the help page:

Lab 1 task 3 9
FOLLOWING

Lab1, Task 4 12
FOLLOWING

```
>> help colon
 :  Colon.
    J:K  is the same as [J, J+1, ...,
J+m], where m = fix(K-J). In the
    case where both J and K are
integers, this is simply [J, J+1, ...,
K].
    This syntax returns an empty matrix
if J > K.

    J:I:K  is the same as [J, J+I, ...,
J+m*I], where m = fix((K-J)/I).
    This syntax returns an empty matrix
when I == 0, I > 0 and J > K, or
    I < 0 and J < K.

    colon(J,K) is the same as J:K and
colon(J,I,K) is the same as J:I:K.

    The colon notation can be used to
pick out selected rows, columns
    and elements of vectors, matrices,
and arrays.  A(:) is all the
    elements of A, regarded as a single
column. On the left side of an
    assignment statement, A(:) fills A,
preserving its shape from before.
    A(:,J) is the J-th column of A.
A(J:K) is [A(J),A(J+1),...,A(K)].
    A(:,J:K) is
[A(:,J),A(:,J+1),...,A(:,K)] and so on.

    The colon notation can be used with
a cell array to produce a comma-
    separated list.  C{:} is the same
as C{1},C{2},...,C{end}.  The comma
    separated list syntax is valid
inside () for function calls, [] for
    concatenation and function return
arguments, and inside {} to produce
    a cell array.  Expressions such as
S(:).name produce the comma separated
    list
S(1).name,S(2).name,...,S(end).name for
```

```
the structure S.

    For the use of the colon in the FOR
statement, See FOR.
    For the use of the colon in a comma
separated list, See VARARGIN.

    Reference page for colon
    Other functions named colon
```

Hope this helps.

Add a comment

pjjurado
2 days ago

0 Votes

Hi

To my understanding is not that easy, since there could be 255 0 x statements which correspond to a single white or black.

Anyhow, I ask the professor to check the grader. It seems there is something weird on it. I am pretty sure I am separating correcting the white and black numbers considering the case with 255 0 x. I was able to reconstruct runs from my white_runs and black_runs, but the grader complains that my probabilities are not OK.

Could you please provide the first 3 numbers for probabilities of white_runs and black_runs to try to compare and see if I am right? I can also paste my probabilities since this is not the answer to the problem ;-)

white_prob = 0.0180 0.0425 0.2498 0.1877 0.1946 0.0795 0.0454 0.0451 0.0191 0.0074 0.1108
black_prob = 0 0.7444 0.1303 0.0689 0.0350 0.0113 0.0001 0.0090 0.0004 0.0003 0.0001

Are my probabilities correct? Any help is welcomed since I'm stuck on this problem lab 1 task 4.

Add a comment

aredirl

about 23 hours ago

0 Votes

Just a guess at the issue . You said you had "reconstructed the runs" by that may I infer that you found the elements corresponding to runs over 255 and summed them ? (e.g. 255 0 3 become 258). The fact that there is a zero element (with associated probability) for the set of runs values is the giveaway, i.e. **DON'T** reconstruct the runs - merely separate the black and white values stored in the **runs** array.

Apologies if I misinterpreted your post, for comparison sake I've added a partial list of the calculated probabilities below. Hope this helps

white_prob = 0 0.0433 0.2543 .....
black_prob = 0.0184 0.7308 0.1279 .....

I got the same probabilities as you have posted above aredirl. The huffman code for 0 probability symbol should be [ ], i.e. there is no code for it. However, the grader says that the length of the code for the first element of the white_prob is incorrect. Can someone help? Thanks.

posted about 12 hours ago by **Googlypk**

Thanks, it was a stupid bug on the code.

Cheers

posted about 6 hours ago by **pjjurado**

Hi, Googlypk Since the first value is 0, that means it is the lowest value. Then when start executing the Huffman algorithm, it is member of the first couple of the two probabilities you must choose. Consequently, it will have the longest code in the white dictionnary. Hope this helps you :D

posted about 5 hours ago by **m_s_william**

Hi pjjurado,

I am getting the same probabilities as you got, from your post. Since these are not the correct probabilities, could you give me hint what I might be doing wrong here.

I applied the following approach:

1. loop over the runs for white runs
2. assume runs starts from white runs.
3. copy the white runs into white_runs. if it includes 255 0 then include both 255 0 in the white_run
4. loop over the runs for black runs
5. copy the black runs into black_runs

Note that I use two different loops for white and black runs.

I hope I am not violating Honor Code rules by providing my incorrect psudocodes :)

Thanks in Advance

posted 43 minutes ago by **sraut**

Add a comment

aredirl

about an hour ago

0 Votes

Look at the image that's being encoded black text on white background. Intuitively in such a 500x500 image there will be no black runs more than 255.Thus, with the given encoding method, there will be no zero elements in the set of white values when you separate the runs . Conversely there are large tracts of white with runs exceeding 255 consecutive white pixels and hence there **will** be zero values in the black.
Remember the basics for information, the lower the probability the higher the information content. Thus lowest probability events are encoded with the most bits in a variable length encoding schema .

Add a comment

KarenWest

36 minutes ago

0 Votes

I have a question on syntax - I'm still working on getting the answer correct, having completed tasks 1-3, but task 4 is giving me a syntax error that is confusing me - have not looked at MATLAB in over a year - does anyone see it? I'm still working on separating out the black and white runs. Here is my code and the syntax error is listed at the top. Note that this is NOT working yet - just started working on it and I do not see the syntax error. Here is the snippet - up to the part where you separate the white and black runs - the error is at the line - trying to concatenate the 2 lists - there's probably an easier way - but thought this should work too?

```
white_runs = [white_runs run_value];

%Error: Line: 38 Column: 25 Unbalanced
or unexpected parenthesis or bracket.
% Load the input image
lorem_img = imread('lorem_img.png');

% display the raw image
figure(1);
imshow(lorem_img);
title('Original image');

% run-length encode
run_length_code =
runlength_encode(lorem_img);
% convert the binary array into an
decimal array of runs
runs = bin2decArray(run_length_code);

% huffman encode
% set the histogram
rlen_list = [0:10,255];

% % % % Revise the following code  % %
% %

% separate the black and white runs
len_runs = length(runs);
white_runs = [];
black_runs = [];

%white_runs = runs(1:len_runs);
%black_runs = runs(1:len_runs);
pixel_value = 1;
run_value = 0;
for run = 1:len_runs, %as in task 2,
encode assumes pixel_value = 1 (white
to start)
    if pixel_value == 1,
        if runs[run] ~= 255,
            pixel_value = 0;
        else
            if runs[run] == 255, %we
have more than 255 1's
```

```
                pixel_value = 1;
            end
        end
        run_value = runs[run];
        white_runs = [white_runs
run_value];
    end
    if pixel_value == 0,
        if runs[run] ~= 255,
            pixel_value = 1;
        else
            if runs[run] == 255, %we
have more than 255 1's
                pixel_value = 0;
            end
        end
        run_value = runs[run];
        black_runs = [black_runs
run_value];
    end
end
```

@KarenWest, in Matlab, if a =[1, 2 3, 4]; then, to refer to the first element, we use a(1) and not a[1]. Hope it helps.

posted 2 minutes ago by dwdrajesh

I bet that is it - I'll try it again - thank you. ;-)

posted less than a minute ago by KarenWest

Add a comment

Showing all responses

Post a response:

PREVIEW

Submit