Huffman Encoding

Anubhav Rathore - Recurssion Based Not Tree

Input: Probabilites of symbols

Output: Encodings for each symbol

Contents

- Defaults
- Inputs
- Condition for faithful M-ary Huffman
- Descending Sorting
- Results
- Algorithm

Defaults

```
clear all;
close all;
clc;
```

Inputs

```
probabs = [0.4 0.3 0.2 0.1];
M = 2;

N = length(probabs);

groups = num2cell(1:N);
codes = repmat({''}, 1, N);
```

Condition for faithful M-ary Huffman

```
while mod((N-1),(M-1)) ~= 0
    probabs(end+1) = 0;
    groups{end+1} = [];   % dummy symbol
    N = N + 1;
end
```

Descending Sorting

```
[probabs, order] = sort(probabs, 'descend');
groups = groups(order);
```

Results

```
codes = Huffman_Encoding(probabs, groups, M, codes);

disp('Symbol   Probability   Code');
for i = 1:length(codes)
    fprintf('%3d        %.2f        %s\n', i, probabs(i), codes{i});
end
```

Algorithm

```
function codes = Huffman_Encoding(probabs, groups, M, codes)

    % Base case: one node left
    if numel(probabs) == 1
        return
    end

    picked_probs  = probabs(end-M+1:end);
    picked_groups = groups(end-M+1:end);

    for d = 0:M-1
        digit = M-1-d;
        symbols = picked_groups{end-d};
        for s = symbols
            codes{s} = strcat(num2str(digit), codes{s});
        end
    end

    new_prob  = sum(picked_probs);
    new_group = [picked_groups{:}];

    probabs(end-M+1:end) = [];
    groups(end-M+1:end)  = [];

    probabs(end+1) = new_prob;
    groups{end+1}  = new_group;

    [probabs, order] = sort(probabs, 'descend');
    groups = groups(order);

    codes = Huffman_Encoding(probabs, groups, M, codes);
end
```

```
Symbol   Probability   Code
  1        0.40        1
  2        0.30        01
  3        0.20        000
  4        0.10        001
```

Published with MATLAB® R2024a