EECE 580B Modern Coding Theory | Homework assignment #8
WWW: http://dde.binghamton.edu/filler/mct/hw/8 | Due date: December 3, 2:50pm

Belief propagation algorithm over the BSC channel

In this assignment, you are asked to implement the Belief propagation algorithm and observe its performance over the BSC channel.

Problem 1: (Log-likelihood ratio for the BSC channel)
Let $y \in \{0,1\}^n$ be a binary sequence obtained through the BSC channel with flipping probability f. Implement a function calculating the log-likelihood ratio of every bit of the received sequence y

$$LLR_i = log \frac{P(y_i|x_i = 0)}{P(y_i|x_i = 1)}.$$

```
function [ LLR ] = bsc_llr( y, f )
%BSC_LLR calculates the log-likelihood ratio of the received vector y for the BSC
%channel with crossover probability f.
% LLR is a vector of the same size as y with elements LLR_i
% this needs to be done ...
end
```

Problem 2: (Belief propagation algorithm)
For a given parity-check matrix H of size $n - k \times n$ and a vector of log-likelihood ratios $LLR \in R^n$, implement a function "ldpc_decode.m" which performs i iterations of the Belief propagation algorithm and outputs binary word $\hat{x} \in \{0,1\}^n$ representing the final bitwise ML estimate of every bit. Use the "tanh" version of the BP algorithm. Use the function from Problem 1 to obtain the vector LLR.

I expect the function "ldpc_decode.m" in the following form:

```
function x_hat = ldpc_decode(H, LLR, i)
%LDPC_DECODE implements the Belief propagation algorithm for
% LDPC codes with initial log-likelihood ratios LLR.
% Function performs i iterations of the BP algorithm, calculates final ML estimate
% and outputs this bitwise estimate as x_hat

% this needs to be done ...
end
```

Use the following pseudo-code to implement the "ldpc_decode.m" function:

```
Initialize all the check->variable messages to 0
for iter = 1 to i
  update all the variable->check messages using the update rule (sum)
  update all the check->variable messages using the update rule (tanh)
end
Calculate the bitwise estimate of every bit, x_hat.
```

Use the handout from Lecture 22 or the slides to implement the update equations. Use the "tanh" version of the BP algorithm.

To test your code, we will use the following setup. In the "data.mat" file you find two parity-check matrices H_1 and H_2 and 6 noisy words obtained by sending the all-zero codeword over the BSC with different probability of flipping. I used the following Matlab code to obtain the test data:

```
L = [0 0 1]; R = [0 0 0 0 0 1];       % (3,6) regular code
H1 = ldpc_create_matrix(L, R, 1000);
L = [0 0.4824 0.2979 0.067 0 0 0 0 0.1527]; R = [0 0 0 0 0 0 1]; % irregular code
H2 = ldpc_create_matrix(L, R, 1000);

f = [0.03 0.04 0.04 0.05 0.04 0.05];  % parameters of the BSC channel
for i=1:6
    y{i} = double(rand(1,1000)<f(i)); % send the all-zero codeword over the BSC
end
```

By using a fixed number of iterations of the BP algorithm, the number of errors in these noisy words should be decreased to zero. Use the following function to plot the number of remaining errors:

```
function draw_bp_error_iter(H, y, f)
    max_iter = 0:15;
    p = [];
    for i=1:numel(max_iter)
        p(i) = mean(ldpc_decode(H, bsc_llr(y,f), max_iter(i) ));
    end
    figure; plot(max_iter, p); xlabel('i');
    title('Bit error probability after i iterations of the BP algorithm');
end
```

where H is the parity-check matrix of the code, y is the noisy word and f is the flipping probability of the BSC channel. Use this function in the following script to run the experiments

```
clc; clear; load('data.mat');

draw_bp_error_iter(H1, y{1}, f(1)); saveas(gcf, 'my_H1_1.png');
draw_bp_error_iter(H1, y{2}, f(2)); saveas(gcf, 'my_H1_2.png');
draw_bp_error_iter(H1, y{3}, f(3)); saveas(gcf, 'my_H1_3.png');
draw_bp_error_iter(H1, y{4}, f(4)); saveas(gcf, 'my_H1_4.png');
draw_bp_error_iter(H2, y{1}, f(1)); saveas(gcf, 'my_H2_1.png');
draw_bp_error_iter(H2, y{2}, f(2)); saveas(gcf, 'my_H2_2.png');
draw_bp_error_iter(H2, y{3}, f(3)); saveas(gcf, 'my_H2_3.png');
draw_bp_error_iter(H2, y{4}, f(4)); saveas(gcf, 'my_H2_4.png');

draw_bp_error_iter(H1, y{5}, f(5)); saveas(gcf, 'my_H1_5.png');
draw_bp_error_iter(H1, y{6}, f(6)); saveas(gcf, 'my_H1_6.png');
draw_bp_error_iter(H2, y{5}, f(5)); saveas(gcf, 'my_H2_5.png');
draw_bp_error_iter(H2, y{6}, f(6)); saveas(gcf, 'my_H2_6.png');
```

As a reference for you, the first 8 graphs are provided (download the ZIP file from the course web site). Your graphs should be almost identical to those in the ZIP file if the algorithm is implemented correctly.

As an output of this problem I want the following:

- Complete code listing of the functions “ldpc_decode.m”, “bsc_llr.m”.
- The last four graphs from the above script.