CSC420

Assignment 5

Vibhavi Peiris
Student Number: 1000597687

December 6, 2017

Note*: **All Coding done on Matlab R2016a.**

Question 1 – Tracking:

a. Instead of using a list I decided to use a struct with the format seen below.
   tracks =
   struct(previous_frame,previous_frame_val,occurences,occurences_val,dets,dets_val,frames_field,frames_val);
   I loop through the frames find the sim matrix (using the code provided)
   Then I have a while loop that stops once there are only 0's left in the sim matrix.
   Each iteration the loop find the max value in the sim matrix, returns its k and t value.
   Then checks if that current detection is in the tracks. I do this by getting an array containing the last added detection for each track. If the current detection is in the track I add it to that specific track(i). When adding it I increase the occurrences field by 1, add the new detection(next_det) to the end of the detections list. I save the current frame value + 1, and I replace the prev_det_val with next_det (this is the latest detection on this track). The I updated that track(i) with this struct. If the current detection was not found in tracks list then I create a new struct and add it to the track. With the fields (occurunces = 2, dets = [cur_dets next_dets] frame = current frame value+1, prev_det = next_dets).

   Outside of the frame's loop I use find, to find tracks that have occurrences > 2 and >5 for part 1.2

```
function track_objects()
% this is a very simple tracking algo
% for more serious tracking, look-up the papers in the projects pdf
FRAME_DIR = '../data/frames/';
DET_DIR = '../data/detections/';
start_frame = 62;
end_frame = 71;
%tracks = zeros(0,13);   %formart [dets_cur dets_next occurences]

%field names
previous_frame = 'prev_frame';  previous_frame_val = zeros(4,0);  %had to transpose so ismember would work
occurences = 'occur';  occurences_val = {0};
dets = 'dets';  dets_val = zeros(0,6);
frames_field = 'frames'; frames_val = zeros(0,0);  % the frames a detection occurs in
tracks = struct(previous_frame,previous_frame_val,occurences,occurences_val,dets,dets_val,frames_field,frames_val);

    for idx = start_frame:end_frame
        im_cur = imread(fullfile(FRAME_DIR, sprintf('%06d.jpg', idx)));
        data = load(fullfile(DET_DIR, sprintf('%06d_dets.mat', idx)));
        dets_cur = data.dets;

        im_next = imread(fullfile(FRAME_DIR, sprintf('%06d.jpg', idx+1)));
        data = load(fullfile(DET_DIR, sprintf('%06d_dets.mat', idx+1)));
        dets_next = data.dets;

        % sim has as many rows as dets_cur and as many columns as dets_next
        % sim(k,t) is similarity between detection k in frame i, and detection
        % t in frame j
        % sim(k,t)=0 means that k and t should probably not be the same track
        sim = compute_similarity(dets_cur, dets_next, im_cur, im_next);

        %greedy approach
        finding = true;   %loop until no values in sim are > 0
        while finding
            [maxValue, idxs] = max(sim(:));  %find current max
            [i,j] = ind2sub(size(sim),idxs);  % get its i and j

            if maxValue == 0
                finding = false; %stop loop all sims>0 are found
            else
                %get previous dets locations from tracks
                prev_dets = [tracks.prev_frame]'; % transpose so that ismember function could find row

                %see if current det is in tracks by comparing with prev_dets
                trans_dets_cur = dets_cur(i,1:4); % get current sims detection  box
                [Result,LocResult] = ismember(trans_dets_cur,prev_dets,'rows');

                if Result  %if current det is in tracks, then add to that track
                    curr_occurs = tracks(LocResult).occur + 1; %increase number of occurence for current track by 1
                    curr_dets = tracks(LocResult).dets;
                    previous_frame_val = dets_next(j,1:4)';   %set prev_fram value to det_next
                    dets_val = [curr_dets;dets_next(j,1:4)];  %append the next det to the tracks dets list
                    frames_val = [tracks(LocResult).frames idx+1];  %save the frames this track was seen in

                    tracks(LocResult) =
struct(previous_frame,previous_frame_val,occurences,curr_occurs,dets,dets_val,frames_field,frames_val);
                    %tracks(LocResult,:) = [dets_cur(i,:) dets_next(j,:) curr_occurs+1];   %save the detections, occured twice
(current & next)

                else
                    %add current det and next det to a new track
                    track_size = size(tracks,2);
                    curr_occurs = 2;
                    previous_frame_val = dets_next(j,1:4)';
                    dets_val = [dets_cur(i,1:4);dets_next(j,1:4)];
                    frames_val = [idx idx+1];

                    tracks(track_size+1) =
struct(previous_frame,previous_frame_val,occurences,curr_occurs,dets,dets_val,frames_field,frames_val);
                    %tracks(track_size+1) = [dets_cur(i,:) dets_next(j,:) 2];   %save the detections, occured twice (current &
next)
                end
                sim(i,j)=0; %set the current sim to 0 so it wont be choosen again
            end
        end
    end;
```

```
%only take tracks with occurences > 2
    idx = find([tracks.occur]>2);
    correct_tracks = tracks(idx);

    %visualize tracks with more than 5
    idx = find([tracks.occur]>5);
    visualize_tracks = tracks(idx);
    track_size = size(visualize_tracks,2);
    colors = {'r','g','b', 'y', 'm', 'c', 'w','k'}; %set colors for modified showboxes function

    %%loop through each frame and show detections
    for idx = start_frame:end_frame
        im_cur = imread(fullfile(FRAME_DIR, sprintf('%06d.jpg', idx)));
        boxes = [];
        frame_colors = {};

        %get the boxes (loop through each track)
        for i = 1:track_size
            track_dets = visualize_tracks(i).dets; %get dets for all frams on current track
            track_frames = visualize_tracks(i).frames;
            location = find(track_frames == idx);

            if location  %if current track has current frame
                boxes = [boxes track_dets(location,:)];
                frame_colors{size(frame_colors,2)+1} = colors{i}; %set a color got current box
            end
        end

        figure,showboxes(im_cur, boxes, frame_colors) %run modified showbox function

    end
end


function sim = compute_similarity(dets_cur, dets_next, im_cur, im_next)

n = size(dets_cur, 1);
m = size(dets_next, 1);
sim = zeros(n, m);



area_cur = compute_area(dets_cur);
area_next = compute_area(dets_next);
c_cur = compute_center(dets_cur);
c_next = compute_center(dets_next);
im_cur = double(im_cur);
im_next = double(im_next);
weights = [1,1,2];

for i = 1: n
    % compare sizes of boxes
    a = area_cur(i) * ones(m, 1);
    sim(i, :) = sim(i, :) + weights(1) * (min(area_next, a) ./ max(area_next, a))';

    % penalize distance (would be good to look-up flow, but it's slow to
    % compute for images of this size)
    sim(i, :) = sim(i, :) + weights(2) * exp((-0.5*sum((repmat(c_cur(i, :), [size(c_next, 1), 1]) - c_next).^2, 2)) / 5^2)';

    % compute similarity of patches
    box = round(dets_cur(i, 1:4));
    box(1:2) = max([1,1],box(1:2));
    box(3:4) = [min(box(3),size(im_cur, 2)), min(box(4),size(im_cur, 1))];
    im_i = im_cur(box(2):box(4),box(1):box(3), :);
    im_i = im_i / norm(im_i(:));
    for j = 1 : m
       d = norm(c_cur(i, :) - c_next(j, :));
       if d>60  % distance between boxes too big
           sim(i,j) = 0;
           continue;
       end;
       box = round(dets_next(j, 1:4));
       box(1:2) = max([1,1],box(1:2));
       box(3:4) = [min(box(3),size(im_cur, 2)), min(box(4),size(im_cur, 1))];
       im_j = im_next(box(2):box(4),box(1):box(3), :);
       im_j = double(imresize(uint8(im_j), [size(im_i, 1), size(im_i, 2)]));
       im_j = im_j / norm(im_j(:));
       c = sum(im_i(:) .* im_j(:));
       sim(i,j) = sim(i,j) + weights(3) * c;
    end;
end;
end

function area = compute_area(dets)
   area = (dets(:, 3) - dets(:, 1) + 1).* (dets(:, 4) - dets(:, 2) + 1);
end

function c = compute_center(dets)

c = 0.5 * (dets(:, [1:2]) + dets(:, [3:4]));
end
```

b. The code for this is at the bottom of the code from part A. I also changed the input of showBoxes(im, boxes, colors,out). So that I can send in the colors relative to the boxes. Then in the showboxes function's main loop I set c = color{i}. the code is below.

```
function showboxes(im, boxes, colors,out)  %changed this added colors input
if nargin > 3
  % different settings for producing pdfs
  print = true;
  %wwidth = 2.25;
  %cwidth = 1.25;
  cwidth = 1.4;
  wwidth = cwidth + 1.1;
  imsz = size(im);
  % resize so that the image is 300 pixels per inch
  % and 1.2 inches tall
  scale = 1.2 / (imsz(1)/300);
  im = imresize(im, scale, 'method', 'cubic');
  %f = fspecial('gaussian', [3 3], 0.5);
  %im = imfilter(im, f);
  boxes = (boxes-1)*scale+1;
else
  print = false;
  cwidth = 2;
end

image(im);
if print
  truesize(gcf);
end
axis image;
axis off;
set(gcf, 'Color', 'white');

if ~isempty(boxes)
  numfilters = floor(size(boxes, 2)/4);
  if print
    for i = 1:numfilters
      x1 = boxes(:,1+(i-1)*4);
      y1 = boxes(:,2+(i-1)*4);
      x2 = boxes(:,3+(i-1)*4);
      y2 = boxes(:,4+(i-1)*4);
      % remove unused filters
      del = find(((x1 == 0) .* (x2 == 0) .* (y1 == 0) .* (y2 == 0)) == 1);
      x1(del) = [];
      x2(del) = [];
      y1(del) = [];
      y2(del) = [];
      if i == 1
        w = wwidth;
      else
        w = wwidth;
      end
      line([x1 x1 x2 x2 x1]', [y1 y2 y2 y1 y1]', 'color', c, 'linewidth', w);
    end
  end
  % draw the boxes with the detection window on top (reverse order)

  if 1
  for i = numfilters:-1:1
  %for i = 1:1
    x1 = boxes(:,1+(i-1)*4);
    y1 = boxes(:,2+(i-1)*4);
    x2 = boxes(:,3+(i-1)*4);
    y2 = boxes(:,4+(i-1)*4);
    % remove unused filters
    del = find(((x1 == 0) .* (x2 == 0) .* (y1 == 0) .* (y2 == 0)) == 1);
    x1(del) = [];
    x2(del) = [];
    y1(del) = [];
    y2(del) = [];
    if i == 1
      c = 'r'; %[160/255 0 0];
      s = '-';
%     elseif i ==  13+1 || i == 14+1
%       c = 'c';
%       s = '--';
    else
      c = 'b';
      s = '-';
    end
    c = colors{i}; %ADDED THIS LINE set color to be realtive to current box
    line([x1 x1 x2 x2 x1]', [y1 y2 y2 y1 y1]', 'color', c, 'linewidth', cwidth, 'linestyle', s);
  end  end;  end

% save to pdf
if print
  % requires export_fig from http://www.mathworks.com/matlabcentral/fileexchange/23629-exportfig
  export_fig([out]);
end
```

c. You could try using a sliding window to find detections from current window in next frame, and keep ones that are above a certain similarity threshold. You could also try checking similarity at different level using a gaussian pyramid.

d. First option would be if there is more than one camera and you have the internal and external camera matrix information. You can compute the 3d word coordinates for the center of the 3d bounding box for each player. See how much that center moves through each frame. This will give you a distance per frame, which you can convert to distance per second depending on how long a frame is. If you have only one camera. You could have the user select the 4 corners of the field, and perform homography to get a better representation of distances (like in a2 with the shoe) then you can just find the difference between each center point of detection box between frames, to get distance travelled.

2. Question 2 – Deep Learning

a. Part 2.1 find derivatives

$$L\left(w,b\right)=-\frac{1}{M}\sum_{i=1}^{M}\left[y_i\log\left(h\left(w^Tx_i+b\right)+\left(1-y_i\right)\log\left(1-h\left(w^Tx_i+b\right)\right)\right)\right]$$

$$h\left(w^Tx_i+b\right)=\frac{1}{1+e^{-(w^Tx_i+b)}}$$

$$\frac{\partial h}{\partial(w^Tx_i+b)}=h\left(w^Tx_i+b\right)\left(1-h\left(w^Tx_i+b\right)\right)$$

$$\frac{\partial h}{\partial w_i}=h\left(w_ix_i+b\right)\left(1-h\left(w_ix_i+b\right)\right)\left(x_i\right)$$

$$\frac{\partial h}{\partial b}=h\left(w_0x_0+b\right)\left(1-h\left(w_0x_0+b\right)\right)$$

can ignore sum because of linerity and also since we are only finding deravtive for w_1

$$\frac{\partial L}{\partial w_1}=-\frac{1}{M}\left(-y_1\frac{1}{h\left(w_1x_1+b\right)}+\left(1-y_1\right)\frac{1}{1-h\left(w_1x_1+b\right)}\right)\frac{\partial h}{\partial w_1}$$

$$=-\frac{1}{M}\left(-y_1\frac{1}{h\left(w_1x_1+b\right)}+\left(1-y_1\right)\frac{1}{1-h\left(w_1x_1+b\right)}\right)h\left(w_1x_1+b\right)\left(1-h\left(w_1x_1+b\right)\right)\left(x_1\right)$$

$$=-\frac{1}{M}\left(-y_1\left(1-h\left(w_1x_1+b\right)\right)+\left(1-y_1\right)h\left(w_1x_1+b\right)\right)\left(x_1\right)$$

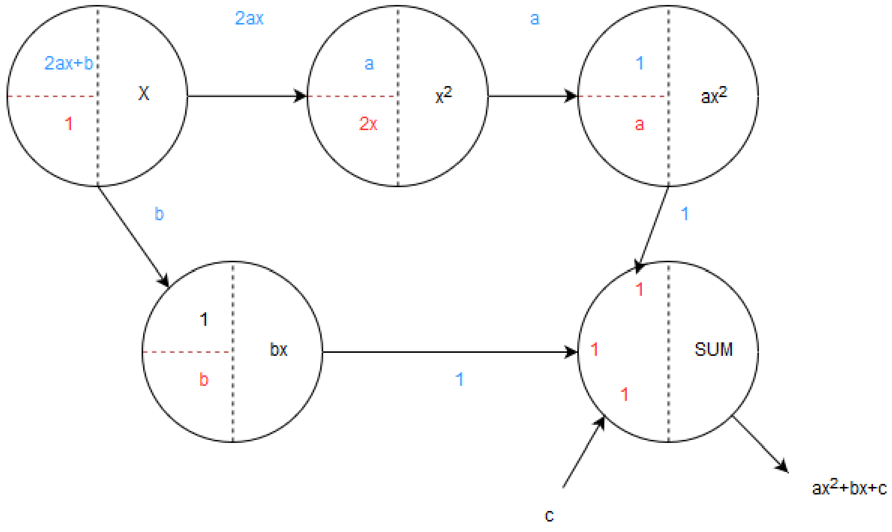
$$=-\frac{1}{M}\left(-y_1+y_1h\left(w_1x_1+b\right)+h\left(w_1x_1+b\right)-y_1h\left(w_1x_1+b\right)\right)\left(x_1\right)$$

$$=-\frac{1}{M}\left(-y_1+h\left(w_1x_1+b\right)\right)\left(x_1\right)$$

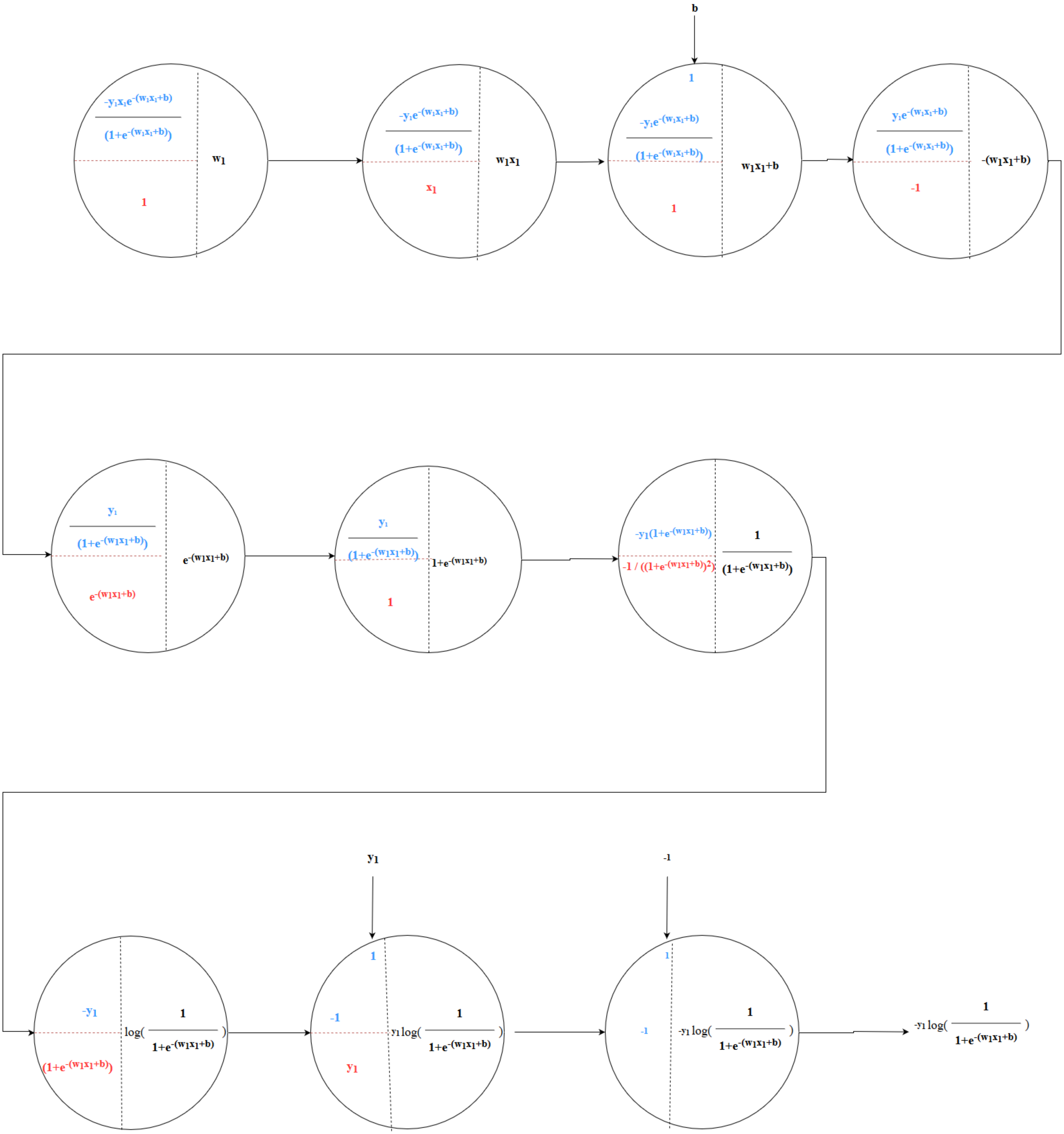
$$\begin{aligned}\frac{\partial L}{\partial w_2} &= -\frac{1}{M}\left(-y_2\frac{1}{h(w_2x_2+b)}+(1-y_2)\frac{1}{1-h(w_2x_2+b)}\right)\frac{\partial h}{\partial w_2}\\ &= -\frac{1}{M}\left(-y_2\frac{1}{h(w_2x_2+b)}+(1-y_2)\frac{1}{1-h(w_2x_2+b)}\right)h(w_2x_2+b)\big(1-h(w_2x_2+b)\big)(x_2)\\ &= -\frac{1}{M}\Big(-y_2\big(1-h(w_2x_2+b)\big)+(1-y_2)h(w_2x_2+b)\Big)(x_2)\\ &= -\frac{1}{M}\big(-y_2+y_2h(w_2x_2+b)+h(w_2x_2+b)-y_2h(w_2x_2+b)\big)(x_2)\\ &= -\frac{1}{M}\big(-y_2+h(w_2x_2+b)\big)(x_2)\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial b} &= -\frac{1}{M}\left(-y_0\frac{1}{h(w_0x_0+b)}+(1-y_0)\frac{1}{1-h(w_0x_0+b)}\right)\frac{\partial h}{\partial b}\\ &= -\frac{1}{M}\left(-y_0\frac{1}{h(w_0x_0+b)}+(1-y_0)\frac{1}{1-h(w_0x_0+b)}\right)h(w_0x_0+b)\big(1-h(w_0x_0+b)\big)\\ &= -\frac{1}{M}\Big(-y_0\big(1-h(w_0x_0+b)\big)+(1-y_0)h(w_0x_0+b)\Big)\\ &= -\frac{1}{M}\big(-y_0+y_2h(w_0x_0+b)+h(w_0x_0+b)-y_0h(w_0x_0+b)\big)\\ &= -\frac{1}{M}\big(-y_0+h(w_0x_0+b)\big)\end{aligned}$$

b. Question 2.2



c. Question 2.3 & 2.4



b
$\frac{-y_1 x_1 e^{-(w_1x_1+b)}}{(1+e^{-(w_1x_1+b)})}$
1
w_1
$\frac{-y_1 e^{-(w_1x_1+b)}}{(1+e^{-(w_1x_1+b)})}$
x_1
w_1x_1
1
$\frac{-y_1 e^{-(w_1x_1+b)}}{(1+e^{-(w_1x_1+b)})}$
1
w_1x_1+b
$\frac{y_1 e^{-(w_1x_1+b)}}{(1+e^{-(w_1x_1+b)})}$
-1
$-(w_1x_1+b)$
$\frac{y_1}{(1+e^{-(w_1x_1+b)})}$
$e^{-(w_1x_1+b)}$
$e^{-(w_1x_1+b)}$
$\frac{y_1}{(1+e^{-(w_1x_1+b)})}$
1
$1+e^{-(w_1x_1+b)}$
$-y_1(1+e^{-(w_1x_1+b)})$
$-1 / ((1+e^{-(w_1x_1+b)})^2)$
$\frac{1}{(1+e^{-(w_1x_1+b)})}$
$-y_1$
$(1+e^{-(w_1x_1+b)})$
$\log(\frac{1}{1+e^{-(w_1x_1+b)}})$
y_1
1
-1
y_1
$y_1\log(\frac{1}{1+e^{-(w_1x_1+b)}})$
-1
1
-1
$-y_1\log(\frac{1}{1+e^{-(w_1x_1+b)}})$
$-y_1\log(\frac{1}{1+e^{-(w_1x_1+b)}})$