Abhijeet %%

Table of Contents

Convention Followed :

CAPSLOCK : Vector input : Input Layer hidden_ : Hidden Layer output_ : Output Layer

NEURAL NET BATCH BACK-PROPAGATION WITH 3 LAYERS%%

activation fucntion is assumed sigmoid INPUT input_X :: original data target_T :: expected output of data(target parameter result of preprocessing) eta :: ita d_cap :: number of units in layer2

OUTPUT w :: weights from input to middle layer s :: weigths from middle to final layer

Preporcessing.

Convention used for 2 output units : [1,0] -> 0 [0,1] -> 1

Open the Validation file %

```
f_optdigits = fopen('optdigits-orig.cv');

% Iterate over file chars, till End Of File. %
i = 1;

% Check for end of file.%
while ~feof(f_optdigits)

    % Store Labels, when i is divisible by 33. %
    if mod(i,33)==0
        validation_label(i/33,:)=fgetl(f_optdigits);

    % Store line otherwise. %
    else
        data_validation_line(i,:)=fgetl(f_optdigits);

    end
```

```
    % Increment, i and save next line. %
    i=i+1;

end

%Open Training file. %
f_optdigits = fopen('optdigits-orig.tra');

% Iterate over training file chars, till End of File. %
i = 1;

% Check for end of file. %
while ~feof(f_optdigits)

    % Store Labels, when i is divisible by 33. %
    if mod(i,33)==0
        training_label(i/33,:)=fgetl(f_optdigits);

    % Store line otherwise. %
    else
        data_training_line(i,:)=fgetl(f_optdigits);
    end

    % Increment, i and save next line. %
    i=i+1;

end


% Remove space before lables in label line.%
training_label = training_label(:,2);
validation_label = validation_label(:,2);
```

```
        Assignment has more non-singleton rhs dimensions than non-singleton subscr

        Error in Question3_GeneralizedNN (line 36)
                validation_label(i/33,:)=fgetl(f_optdigits);
```

Char to int conversion training_data.

From 1 and 0, convert this to binary image. % For 1, change indexes to int(255). %

```
f_optdigits = data_training_line == '1';
data_training_line(f_optdigits) = 255;

% For 0, change indexes to int(0) %
f_optdigits = data_training_line == '0';
data_training_line(f_optdigits) = 0;
```

Char to int conversion validation_data.

From 1 and 0, convert this to binary image. % For 1, change indexes to int(255). %

```
f_optdigits = data_validation_line == '1';
data_validation_line(f_optdigits) = 255;

% For 0, change indexes to int(0) %
f_optdigits = data_validation_line == '0';
data_validation_line(f_optdigits) = 0;
```

Convert to double.

```
data_training_line = double(data_training_line);
data_validation_line = double(data_validation_line);
```

Question Specifications, consider data only for 7 and 0.

Find indexes for labels 7 and 0 % We have a two-category system. %

```
f_optdigits = find(training_label == '7' | training_label == '0');

% Declare training data to be used %
train = zeros(8,8,length(f_optdigits));
train_label = zeros(2,length(f_optdigits));

for i = 1:size(f_optdigits,1)

    % Assign 1 for 7%
    if training_label(f_optdigits(i)) == '7'
        b = [0,1];
    % Assign 0 for 0%
    else
        b = [1,0];
    end

    % Make train data, line by line.%
    temp_window = data_training_line((f_optdigits(i)-1)*33+1:f_optdigits(i)*33-1,:
    train(:,1:8,i) = imresize(double(temp_window), 0.25);
    train_label(:,i) = b;

end

% Find indexes for labels 7 and 0 %
% We have a two-category system. %
f_optdigits = find(validation_label == '7' | validation_label == '0');

% Declare validation data to be used %
validate = zeros(8,8,length(f_optdigits));
validate_label = zeros(2, length(f_optdigits));

for i = 1:length(f_optdigits)

    % Assign 1 for 7%
    if validation_label(f_optdigits(i)) == '7'
```

```
        b = [0,1];
    else
    % Assign 0 for 0%
        b = [1,0];
    end

    % Make validate data, line by line.%
    temp_window = data_validation_line((f_optdigits(i)-1)*33+1:f_optdigits(i)*33-1
    validate(:,1:8,i) = imresize(double(temp_window), 0.25);
    validate_label(:,i) = b;

end
```

BackPropogation section.

```
input_X = zeros(65, size(train,3));

for i=1:size(train,3)
    temp_window = train(:,:,i);
    temp_window = [1;temp_window(:)];
    input_X(:,i) = 100*double(temp_window(:))/sum(temp_window(:).*temp_window(:));
end

input_X = input_X';
target_T = train_label;
eta = 0.5;
d_cap = 20;

% Initialize w (weights of input-to-hidden layer)%
input_size = size(input_X);
w = rand(d_cap, input_size(2));

% Initialize s (weights of hidden-to-output layer)%
s = rand(2, d_cap+1);

% Initialize hidden_Y. %
hidden_Y = double(ones(input_size(1), d_cap+1));

% Compute hidden_Y. %
hidden_Y(:,1:d_cap) = (w*input_X')';
hidden_Y_size = size(hidden_Y);
for i=1:hidden_Y_size(1)

    for j=1:hidden_Y_size(2)-1

        % Apply activation function, which is sigmoid funcation here. %
        temp = sigmf(hidden_Y(i,j),[1,0]);
        % Update the values in hidden_Y%
        hidden_Y(i,j) = temp;

    end

end
```

```
% Initialize output_Z. %
output_Z = zeros(input_size(1),2);

% Compute output_Z. %
output_Z = (s*hidden_Y')';
output_Z_size = size(output_Z);
for i=1:output_Z_size(1)

    for j=1:output_Z_size(2)

        % Apply activation function, which is sigmoid function here. %
        temp = sigmf(output_Z(i,j), [1,0]);

        % Update the values in output_Z%
        output_Z(i,j) = temp;

    end

end

% delta_s is delta and delta_w is my mu. %
delta = zeros(2,1);
mu = zeros(size(hidden_Y,2),1);

for iter=1:1000
    for i=1:2
        t = target_T(i);
        z = output_Z(i);
        delta(i) = (t-z)*(z*(1-z));
    end
    %delta
    s = s - eta*repmat(delta, [1, input_size(1)])*hidden_Y;

    for i=1:d_cap
        y = hidden_Y(i);
        sub = 0;
        for j=1:2
            sub = sub + delta(j)*s(j,i);
        end
        mu(i) = sub*(y*(1-y));
    end
    %mu
    w = w - eta*repmat(mu(1:d_cap), [1, input_size(1)])*input_X;


    % Compute Z and compare again. %

    hidden_Y(:,1:d_cap) = (w*input_X')';
    for i=1:hidden_Y_size(1)

        for j=1:hidden_Y_size(2)-1

            % Apply activation function, which is sigmoid funcation here. %
```

```
            temp = sigmf(hidden_Y(i,j), [1,0]);
            % Update the values in hidden_Y%
            hidden_Y(i,j) = temp;
        end

    end

    output_Z = (s*hidden_Y')';
    for i=1:output_Z_size(1)

        for j=1:output_Z_size(2)

            % Apply activation function, which is sigmoid function here. %
            temp = sigmf(output_Z(i,j),[1,0]);
            % Update the values in output_Z%
            output_Z(i,j) = temp;

        end

end

end
```

Published with MATLAB® R2013a