Table of Contents


```
clc; clear

hidden_layer_1 = 3;
hidden_layer_2 = 3;
input_layer = 2;
output_layer = 1;
epoch_length = 3000;
learning_rate = 0.1;

x = [0,0;0,1;1,0;1,1];
y = [1;0;0;1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%sigmoid_function
g = @(z) 1.0./(1.0+exp(-z));
sigma_prime = @(a) a.*(1-a);

%%hyperbolic tangent
% g = @(z) tanh(z);
% sigma_prime = @(a) 1+(tanh(a).^2);

%%ReLu activation
% g = @(z) max(z,0);
% sigma_prime = @(a) 1*(a>0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%RMS Error
delta_C = @(a,y) a-y

%%Logloss Error
% delta_C = @(a,y,z) z(:,1).*(a-y)
% delta_C = @(a,y,z) ((1-y)./(1-a))-(y./a)

weights = {(rand(input_layer,hidden_layer_1)),...
    (rand(hidden_layer_1,output_layer))};
biases = {(rand(1,hidden_layer_1)),...
    (rand(1,output_layer))};


delta_C =
```

```
@(a,y)a-y
```

Test Set

```
weights = {[0,0.5;-1,1;0,1]',[0,1,-1]'};
biases = {[1;1;1]',1};
x = [1,0];
y = [0];

for epoch = 1:epoch_length

    z = x;
```

Forward Propogation

```
    for layer = 1:2

        zs{layer} = z*weights{layer}+biases{layer};
        a{layer} = g(zs{layer});
        z = a{layer};

    end
```

Cost Function

```
    err(epoch) = (1/2) .* sum((z-y).^2);
```

logloss function

```
        err(epoch) = (1/4) .* sum(-y.*log(z)-(1-y).*log(1-z));
```

Backward Propogation

```
    for layer = 2:-1:1

        if layer == 2
            delta = delta_C(a{layer},y).*sigma_prime(a{layer});
            nabla_b{layer} = delta;
            nabla_w{layer} = a{layer-1}'*delta;
        else
            delta = delta*weights{layer+1}'.*sigma_prime(a{layer});
            nabla_b{layer} = delta;
            nabla_w{layer} = x'*delta;
%         else
%             delta = delta*weights{layer+1}'.*sigma_prime(a{layer});
%             nabla_b{layer} = delta;
%             nabla_w{layer} = a{layer-1}'*delta;
        end
```

```
    end
```

Update weights

```
    for layer = 2:-1:1
        weights{layer} = weights{layer} -
 (learning_rate.*nabla_w{layer});
        biases{layer} = biases{layer} -
 (learning_rate.*nabla_b{layer});
    end

end

plot(1:1:epoch_length,err)
```

Published with MATLAB® R2016a