```
%The following back-propagation algorithm has been written using both
%professor Bibbona lectures and Gosavi's book notes

function[W,v,b,k,o,SSE_new]=back_propagation_NN(x,y_n,tol)
% the function returns the matrix of weights W,the hidden neurons v,
 the
% bias node b, the vector of weights k, the output vector and the
 SSE_new
% through the iterations
M=max(x)-min(x);

n=length(x); %number of rows of our dataset
%we didn't normalise the input as it was already in [0,1]. In case you
 need
%to normalise you can do it in the main script before calling this
%function
X =[ones(n,1) x'];%we add a bias node set to 1 to the input layer
N=length(x(:,1));
H=4; %number of neurons in the hidden layer, without counting the bias
 node

%%normalising the output
for p=1:length(y_n)
    y(p)=(y_n(p)-min(y_n))/(max(y_n)-min(y_n));
end

%STEP 1 - ALGORITHM setting all weights and kh to small random numbers
%between 0 and 0.5, fix tol to a small number
b=unifrnd(0,0.5); %setting the bias node in the hidden layer as a
 uniform random sample
W=unifrnd(0,0.5,N+1,H);
k=unifrnd(0,0.5,H,1); %H+1 there is a bias node in the hidden layer

%STEP 2 - ALGORITHM setting SSE_old to a large value, then set m=0.
SSE_old=1000;
%mu=0.005;
mu=0.05; % we set the learning rate as a constant as we encountered
 problems
%STEP 3 - ALGORITHM while using a learning rate depending on m
for m=1:1000

%sigma=@(u) (1+exp(-u)).^(-1); %sigmoid function defined as a handle
 function

%gosavi improvement of sigmoid function to avoid overfitting and norm
 issues.
sigma=@(u) (1+exp(-u/M)).^(-1);

%we compute v_star componentwise as defined in the algorithm
v_star=zeros(n,H);
for p = 1:n
    for h=1:H
```

```
        for j= 1:N+1
           v_star(p,h)=v_star(p,h)+W(j,h)*X(p,j);
        end
    end
end

v=sigma(v_star);
%STEP 4 - ALGORITHM computation of the output componentwise
o=b*ones(n,1);
for p=1:n
    for h = 1:H
        o(p)=o(p)+k(h)*v(p,h);
    end
end
%UPDATE BIAS ON THE HIDDEN LAYER
sum=0;
for p=1:n
    sum=sum+y(p)-o(p);
end
b=b+mu*sum;

%STEP 5 - ALGORITHM update weights w_ih
for i = 1:N+1
    for h = 1:H
        sum=0;
        for p = 1:n
            sum=sum+(y(p)-o(p))*k(h)*v(p,h)*(1-v(p,h))*X(p,i);
        end
        W(i,h)=W(i,h)+mu*sum;

    end
end


%%STEP 6 - ALGORITHM update weights k_h
for h=1:H
    sum=0;
    for p =1:n
        sum=(y(p)-o(p))*v(p,h);
    end
    k(h)=k(h)+mu*sum;
end

SSE_new(m)=(norm(y-o,2))^2;%saving the SSE in a vector

if SSE_new(m)<tol
    break

end

end
%we apply the inverse of the normalising function in order to resize
 the
%output
```

```
for i=1:length(y)
    o(i)=(max(y_n)-min(y_n))*o(i)+min(y_n);

end
figure('name','comparising the output of the NN')
plot(x,y_n,'o',x,o,'*')%plot
legend('exact value','interpolated value')

figure('name', 'Plot of SSE changes through the iterations')
plot(linspace(0,m,m),SSE_new) %optional plot to check how the SSE
 changes
%through the iterations
end
```

```
Not enough input arguments.

Error in back_propagation_NN (line 8)
M=max(x)-min(x);
```

Published with MATLAB® R2021a