# 332:521 – Digital Signal Analytics – Spring 2021
## NN Experiments – S. J. Orfanidis

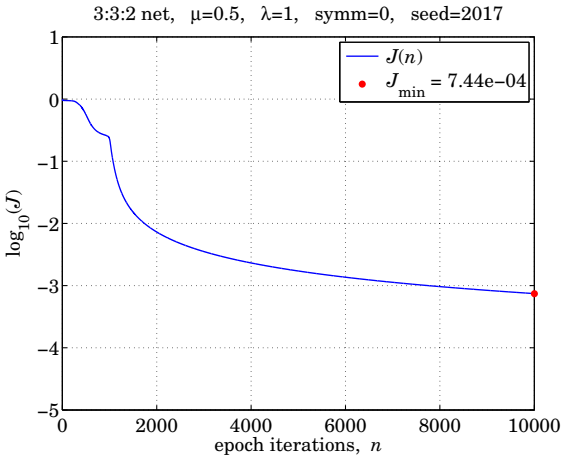
### XOR problem with 3:3:2 network

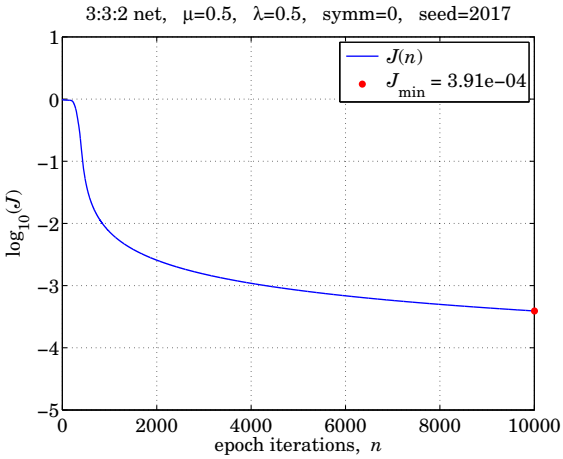
```
%     input      | XOR output
% --------------------------
%  0    0    0   |   0    1
%  0    0    1   |   1    0
%  0    1    0   |   1    0
%  0    1    1   |   0    1
%  1    0    0   |   1    0
%  1    0    1   |   0    1
%  1    1    0   |   0    1
%  1    1    1   |   1    0
```

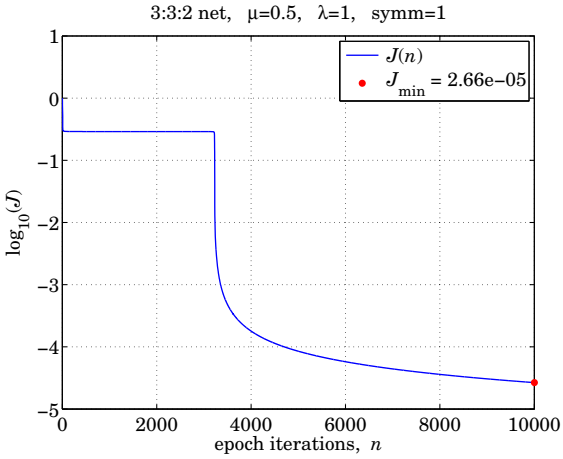
```
% s21xor1.m - XOR problem - 3:3:2 network
% DSA - Spring 2021 - S. J. Orfanidis

symm = 0;                           % symm=0, sigmoid

if symm==0,
   f = @(u) 1./(1+exp(-u));         % standard sigmoid
  df = @(u) diag(f(u).*(1-f(u)));   % sigmoid derivative
else
   f = @(u) tanh(u/2);              % symmetric sigmoid, 2*f(u)-1
  df = @(u) diag((1-f(u).^2)/2);    % sigmoid derivative
end

X0 = [0  0  0  0  1  1  1  1       % input layer, training patterns
      0  0  1  1  0  0  1  1
      0  1  0  1  0  1  0  1];

 D = [0  1  1  0  1  0  0  1       % output layer, training patterns
      1  0  0  1  0  1  1  0];

if symm==1,
    X0  = 2*X0 - 1;
    D = 2*D - 1;
end

P = size(X0,2);           % no. patterns

M0 = size(X0,1);          % M0:M1:M2 network
M1 = 3;
M2 = size(D,1);           % 3:3:2 network

randn('state',2017);      % initial seed, try also seed=20

A=1; B=0;
if symm==1,
   A=2; B=1;
end
W0 = A*randn(M1,M0)-B;    % initialize weights & biases
b0 = A*randn(M1,1)-B;
W1 = A*randn(M2,M1)-B;
b1 = A*randn(M2,1)-B;

mu = 0.5;     % adaptation parameter

% la = 0.5;   % pattern updating la<1
la=1;         % epoch updating la=1

Niter = 10000;                % no. epoch iterations
```

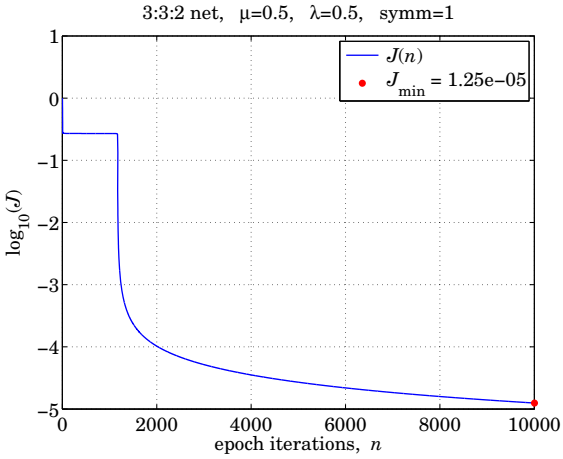
```
for n = 1:Niter,               % training epoch iterations

    J(n) = 0;                  % performance index
    dW1=0; db1=0; dW0=0; db0=0;

    for p = 1:P,               % iterate over patterns
        x0 = X0(:,p);          % input/output training pair
        d = D(:,p);

        u1 = W0*x0 + b0;       % forward pass
        x1 = f(u1);
        u2 = W1*x1 + b1;
        x2 = f(u2);            % network output

        J(n) = J(n) + norm(d-x2)^2;     % accumulate index

        e2 = df(u2)*(d-x2);     % back-propagation pass
        e1 = df(u1)*W1'*e2;

        dW1 = la*dW1 + mu * e2 * x1';  % weight corrections
        db1 = la*db1 + mu * e2;
        dW0 = la*dW0 + mu * e1 * x0';
        db0 = la*db0 + mu * e1;

        if (la<1)              % pattern updating
            W1 = W1 + dW1;     % update weights
            b1 = b1 + db1;
            W0 = W0 + dW0;
            b0 = b0 + db0;
        end

    end   % pattern loop

    if (la==1)                 % epoch updating
        W1 = W1 + dW1;         % update weights
        b1 = b1 + db1;
        W0 = W0 + dW0;
        b0 = b0 + db0;
    end

end                            % stop iterations

X2 = [];
for p=1:P,                % output of trained network
    x0 = X0(:,p);
    d = D(:,p);

    u1 = W0*x0 + b0;
    x1 = f(u1);
    u2 = W1*x1 + b1;
    x2 = f(u2);

    X2 = [X2, x2];        % collect outputs for printing
end

if symm==1,
    X0 = (X0 + 1)/2;
    D = (D + 1)/2;
    X2 = (X2 + 1)/2;
end

fprintf('symm = %1d\n', symm);
fprintf('-------------------------------\n');
fprintf('   X0   |   D  |       X2\n')
fprintf('-------------------------------\n');
fprintf('%1d  %1d  %1d | %1d  %1d | %1.4f  %1.4f\n',[X0; D; X2]);
```

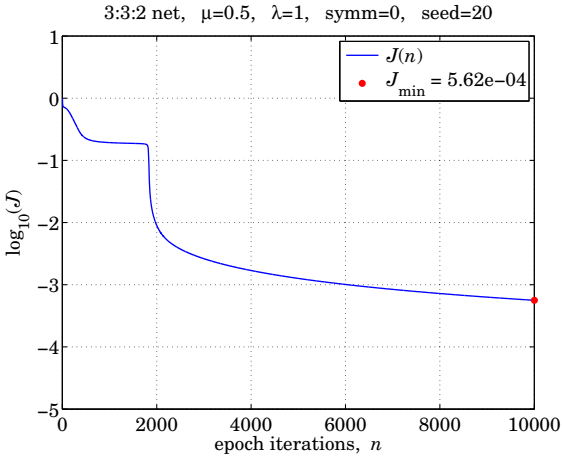
```
% symm = 0
% -------------------------------
%    X0   |   D  |       X2
% -------------------------------
% 0  0  0 | 0  1 | 0.0136  0.9866
% 0  0  1 | 1  0 | 0.9846  0.0152
% 0  1  0 | 1  0 | 0.9863  0.0136
% 0  1  1 | 0  1 | 0.0119  0.9883
% 1  0  0 | 1  0 | 0.9863  0.0136
% 1  0  1 | 0  1 | 0.0116  0.9885
% 1  1  0 | 0  1 | 0.0198  0.9803
% 1  1  1 | 1  0 | 0.9889  0.0110

% symm = 1
% -------------------------------
%    X0   |   D  |       X2
% -------------------------------
% 0  0  0 | 0  1 | 0.0040  0.9970
% 0  0  1 | 1  0 | 0.9958  0.0032
% 0  1  0 | 1  0 | 0.9957  0.0032
% 0  1  1 | 0  1 | 0.0013  0.9991
% 1  0  0 | 1  0 | 0.9964  0.0027
% 1  0  1 | 0  1 | 0.0045  0.9966
% 1  1  0 | 0  1 | 0.0046  0.9965
% 1  1  1 | 1  0 | 0.9970  0.0023

J = J/J(1);

mOm1m2 = [num2str(M0),':',num2str(M1),':',num2str(M2)];
Jmin = J(end);

n = 1:Niter;
figure; plot(n,log10(J),'b-', Niter,log10(Jmin),'r.','markersize',20);
yaxis(-5,1,-5:1); grid
title([mOm1m2,' net,   \mu=',num2str(mu), ',   \lambda=',num2str(la), ',   symm=', num2str(symm)])
xlabel('epoch iterations,  {\itn}')
ylabel('log_{10}({\itJ})')
legend(' {\itJ}({\itn})', [' {\itJ}_{min} = ', num2str(Jmin,'%3.2e')], 'location','ne')
```

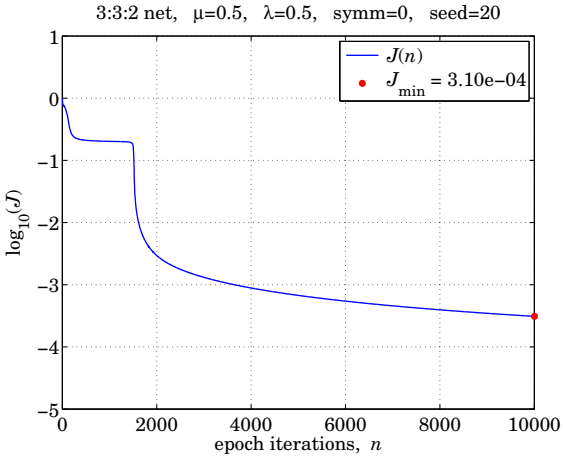

3:3:2 net, μ=0.5, λ=1, symm=0, seed=2017
J(n)
J_min = 7.44e−04
log10(J)
epoch iterations, n


3:3:2 net, μ=0.5, λ=0.5, symm=0, seed=2017
J(n)
J_min = 3.91e−04
log10(J)
epoch iterations, n


3:3:2 net, μ=0.5, λ=1, symm=1
J(n)
J_min = 2.66e−05
log10(J)
epoch iterations, n


3:3:2 net, μ=0.5, λ=0.5, symm=1
J(n)
J_min = 1.25e−05
log10(J)
epoch iterations, n


3:3:2 net, μ=0.5, λ=1, symm=0, seed=20
J(n)
J_min = 5.62e−04
log10(J)
epoch iterations, n


3:3:2 net, μ=0.5, λ=0.5, symm=0, seed=20
J(n)
J_min = 3.10e−04
log10(J)
epoch iterations, n

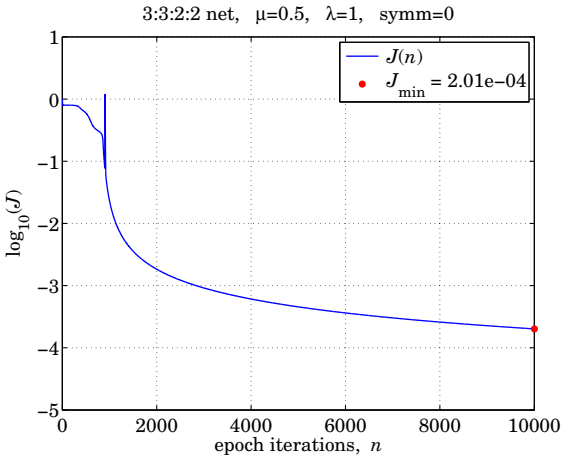
## XOR problem with 3:3:2:2 network

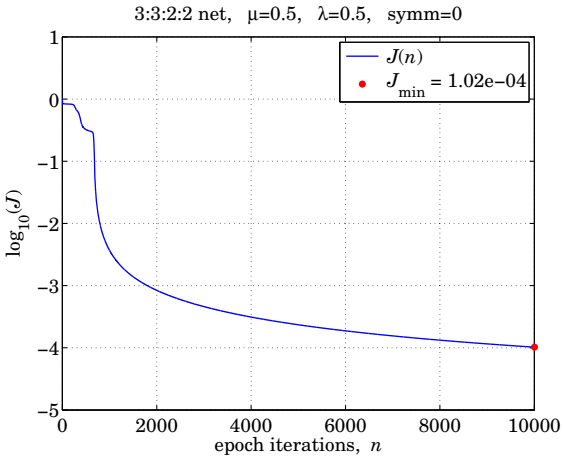
| input | | | XOR output | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

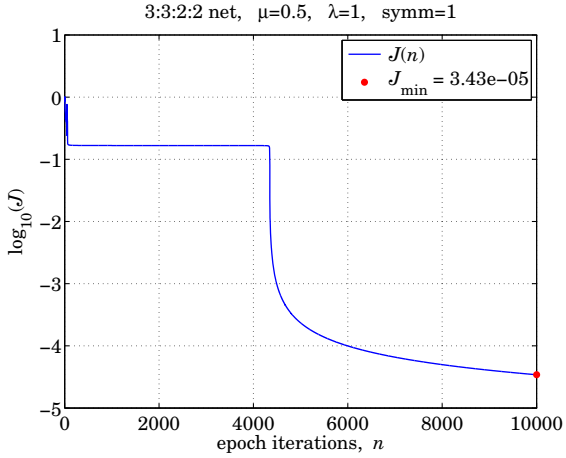
```
% s21xor2.m - XOR problem - 3:3:2:2 network
% DSA - Spring 2021 - S. J. Orfanidis

symm = 0;                           % symm=0, sigmoid

if symm==0,
   f = @(u) 1./(1+exp(-u));         % standard sigmoid
  df = @(u) diag(f(u).*(1-f(u)));   % sigmoid derivative
else
   f = @(u) tanh(u/2);              % symmetric sigmoid, 2*f(u)-1
  df = @(u) diag((1-f(u).^2)/2);    % sigmoid derivative
end

X0 = [0  0  0  0  1  1  1  1       % input layer, training patterns
      0  0  1  1  0  0  1  1
      0  1  0  1  0  1  0  1];

 D = [0  1  1  0  1  0  0  1       % output layer, training patterns
      1  0  0  1  0  1  1  0];


if symm==1,
    X0  = 2*X0 - 1;
    D = 2*D - 1;
end

P = size(X0,2);           % no. patterns

M0 = size(X0,1);          % M0:M1:M2:M3 network
M1 = 3;
M2 = 2;
M3 = size(D,1);           % 3:3:2:2

randn('state',2017);      % initial seed

A=1; B=0;
if symm==1, A=2; B=1; end
W0 = A*randn(M1,M0)-B;       % initialize weights & biases
b0 = A*randn(M1,1)-B;
W1 = A*randn(M2,M1)-B;
b1 = A*randn(M2,1)-B;
W2 = A*randn(M3,M2)-B;
b2 = A*randn(M3,1)-B;

mu = 0.5;        % adaptation parameter
% la = 0.5;      % pattern updating la<1
la = 1.0;        % epoch updating la=1

Nit = 10000;                    % no. iterations/epochs

for n = 1:Nit,                  % training iterations

    J(n) = 0;                   % performance index
    dW1=0; db1=0; dW0=0; db0=0; dW2=0; db2=0;
```

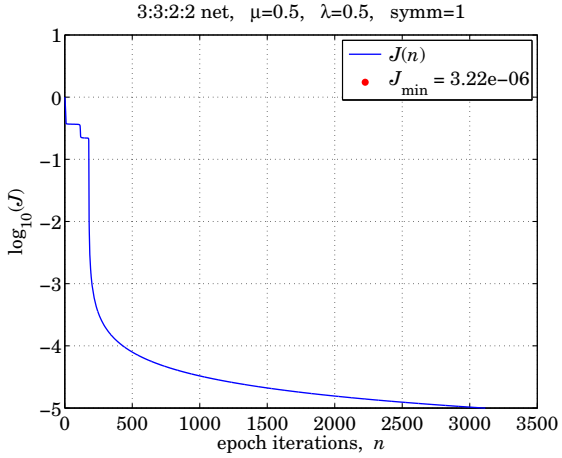
```
    for p = 1:P,                % iterate over patterns
        x0 = X0(:,p);           % input/output training pair
        d = D(:,p);

        u1 = W0*x0 + b0;        % forward pass
        x1 = f(u1);
        u2 = W1*x1 + b1;
        x2 = f(u2);
        u3 = W2*x2 + b2;
        x3 = f(u3);

        J(n) = J(n) + norm(d-x3)^2;   % accumulate index

        e3 = df(u3)*(d-x3);            % back-propagation pass
        e2 = df(u2)*W2'*e3;
        e1 = df(u1)*W1'*e2;

        dW2 = la*dW2 + mu * e3 * x2';    % weight corrections
        db2 = la*db2 + mu * e3;
        dW1 = la*dW1 + mu * e2 * x1';
        db1 = la*db1 + mu * e2;
        dW0 = la*dW0 + mu * e1 * x0';
        db0 = la*db0 + mu * e1;

        if (la<1)               % pattern updating
            W2 = W2 + dW2;      % update weights
            b2 = b2 + db2;
            W1 = W1 + dW1;
            b1 = b1 + db1;
            W0 = W0 + dW0;
            b0 = b0 + db0;
        end

    end      % pattern loop

    if (la==1)                  % epoch updating
        W2 = W2 + dW2;          % update weights
        b2 = b2 + db2;
        W1 = W1 + dW1;
        b1 = b1 + db1;
        W0 = W0 + dW0;
        b0 = b0 + db0;
    end

end                     % stop epoch iterations

X3 = [];
for p=1:P,              % output of trained network
    x0 = X0(:,p);
    d = D(:,p);

    u1 = W0*x0 + b0;     % forward pass
    x1 = f(u1);
    u2 = W1*x1 + b1;
    x2 = f(u2);
    u3 = W2*x2 + b2;
    x3 = f(u3);

    X3 = [X3, x3];        % collect outputs for printing
end

if symm==1,
    X0 = (X0 + 1)/2;
    D = (D + 1)/2;
    X3 = (X3 + 1)/2;
end
```

```
fprintf('symm = %1d\n', symm);
fprintf('------------------------------\n');
fprintf('   X0   |   D  |       X2\n')
fprintf('------------------------------\n');
fprintf('%1d  %1d  %1d | %1d  %1d | %1.4f  %1.4f\n',[X0; D; X3]);

% symm = 0
% ------------------------------
%    X0   |   D  |       X2
% ------------------------------
% 0  0  0 | 0  1 | 0.0044  0.9953
% 0  0  1 | 1  0 | 0.9852  0.0150
% 0  1  0 | 1  0 | 0.9934  0.0070
% 0  1  1 | 0  1 | 0.0041  0.9957
% 1  0  0 | 1  0 | 0.9957  0.0045
% 1  0  1 | 0  1 | 0.0055  0.9942
% 1  1  0 | 0  1 | 0.0112  0.9886
% 1  1  1 | 1  0 | 0.9959  0.0044

% symm = 1
% ------------------------------
%    X0   |   D  |       X2
% ------------------------------
% 0  0  0 | 0  1 | 0.0024  0.9965
% 0  0  1 | 1  0 | 0.9954  0.0068
% 0  1  0 | 1  0 | 0.9973  0.0039
% 0  1  1 | 0  1 | 0.0024  0.9965
% 1  0  0 | 1  0 | 0.9973  0.0040
% 1  0  1 | 0  1 | 0.0019  0.9971
% 1  1  0 | 0  1 | 0.0026  0.9962
% 1  1  1 | 1  0 | 0.9973  0.0040


J = J/J(1);

mOm1m2 = [num2str(M0),':',num2str(M1),':',num2str(M2), ':', num2str(M3)];

Jmin = J(end);

n = 1:Nit;
figure; plot(n,log10(J),'b-', Nit,log10(Jmin),'r.','markersize',20);
yaxis(-5,1,-5:1); grid
title([mOm1m2,' net,   \mu=',num2str(mu), ',   \lambda=',num2str(la), ',   symm=', num2str(symm)])
xlabel('epoch iterations,  {\itn}')
ylabel('log_{10}({\itJ})')
legend(' {\itJ}({\itn})', [' {\itJ}_{min} = ', num2str(Jmin,'%3.2e')], 'location','ne')
```

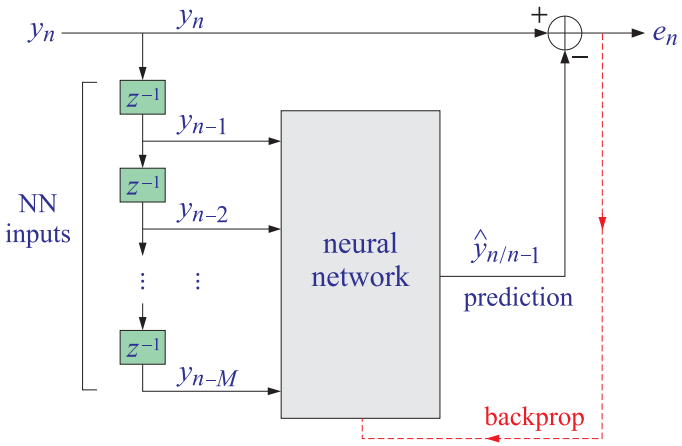

3:3:2:2 net, μ=0.5, λ=1, symm=0
J(n)
J_min = 2.01e−04
log10(J)
epoch iterations, n


3:3:2:2 net, μ=0.5, λ=0.5, symm=0
J(n)
J_min = 1.02e−04
log10(J)
epoch iterations, n


3:3:2:2 net, μ=0.5, λ=1, symm=1
J(n)
J_min = 3.43e−05
log10(J)
epoch iterations, n


3:3:2:2 net, μ=0.5, λ=0.5, symm=1
J(n)
J_min = 3.22e−06
log10(J)
epoch iterations, n

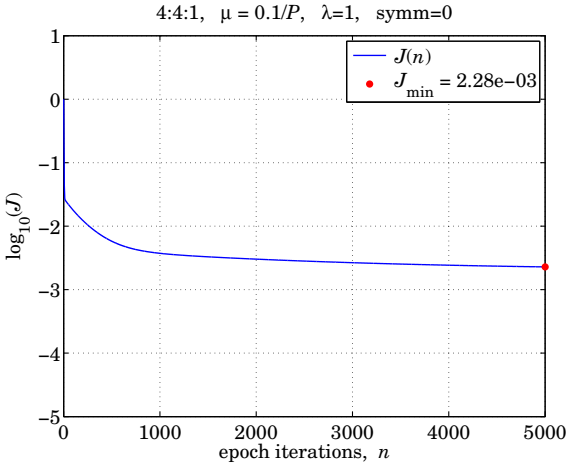
## NN prediction of suspot data

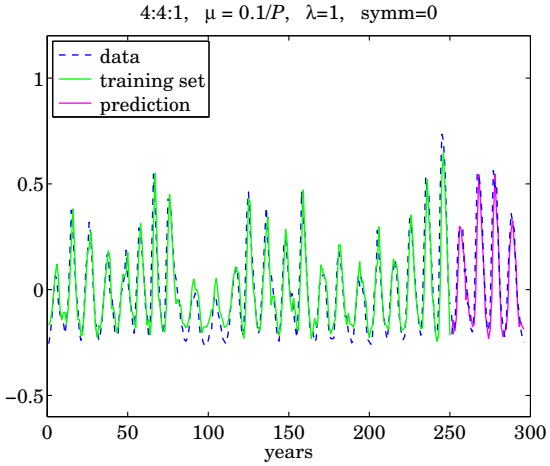



```
% s21nnsun.m - NN prediction of sunspot data
% DSA - Spring 2021 - S. J. Orfanidis

symm = 0;                               % symmetric sigmoid

if symm==0,
    f = @(u) 1./(1+exp(-u));            % standard sigmoid
    D = @(u) diag(f(u).*(1-f(u)));      % sigmoid derivative
elseif symm==1
   f = @(u) tanh(u);                    % symmetric sigmoid
   D = @(u) diag((1-tanh(u).^2));       % sigmoid derivative
else
   c = 0.01;
   f = @(u) max(c*u,u);                 % leaky ReLU
   D = @(u) diag((u>=0) + c*(u<0));
end

Y = loadfile('sunspots.dat');

i = find(Y(:,1)==1709);    % start in the year 1709
y = Y(i:end,2);
maxy = max(abs(y));
y = y/maxy;                % optional normalization

N = length(y);             % here, N=300
my = mean(y);
s2 = std(y)^2;

p = 4;       % 4:4:1 network

X = datamat(y-my,p,1)';                     % cov method
% X = flipud(buffer(y-my,p+1,p,'nodelay'));  % equivalent
%
% rearranges data in the form (for p=4):
%
%  y4  y5  y6  y7        y(n)     = NN desired output
%  y3  y4  y5  y6        y(n-1)   =
%  y2  y3  y4  y5  etc.  y(n-2)   = NN inputs
%  y1  y2  y3  y4        y(n-3)   =
%  y0  y1  y2  y3        y(n-4)   =

Nt = 250;                  % size of training set
Xin = X(2:end, 1:Nt);      % NN inputs of training set
Xout = X(1,1:Nt);          % NN outputs of training set
```

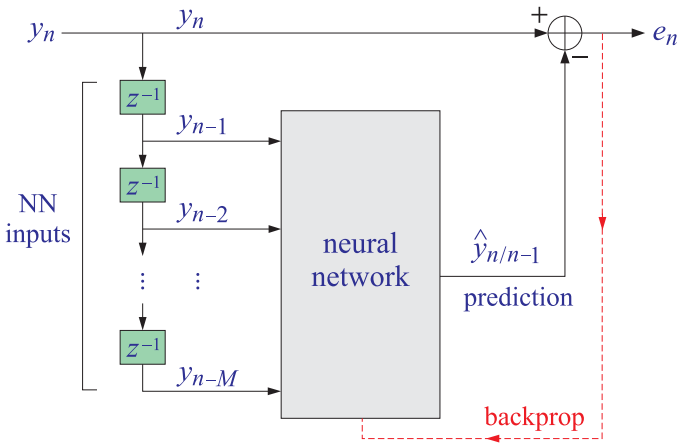
```
P = size(Xin,2);           % no.trainign patterns, here, P = 250

M0 = size(Xin,1);          % M0:M1:M2 network, here M0 = p
M1 = p;                    % p:p:1 network, M1 = 2*p+1 is OK, too
M2 = size(Xout,1);         % here M2 = 1

seed = 2015;
randn('state',seed);       % initial seed

A=1; B=0;
% if symm==1, A=2; B=1; end
W0 = A*randn(M1,M0)-B;     % initialize weights & biases
b0 = 0*A*randn(M1,1)-B;    % bias weights set to zero, optional
W1 = A*randn(M2,M1)-B;
b1 = 0*A*randn(M2,1)-B;

mu = 0.1/P;    % here, mu = 0.0004
la = 1;        % epoch updating la=1, pattern updating la<1

Nit = 5000;                    % no. epoch iterations

for n = 1:Nit,                 % training iterations

    J(n) = 0;                  % performance index
    dW1=0; db1=0; dW0=0; db0=0;

    for p = 1:P,               % iterate over patterns
        x0 = Xin(:,p);         % input/output training pair
        d = Xout(:,p);

        u1 = W0*x0 + b0;       % forward pass
        x1 = f(u1);
        u2 = W1*x1 + b1;
%       x2 = f(u2);            % network output with activation function
        x2 = u2;               % w/o activation function

        J(n) = J(n) + norm(d-x2)^2;   % accumulate index

%       e2 = D(u2)*(d-x2);     % back-propagation pass
        e2 = (d-x2);           % back-propagation pass
        e1 = D(u1)*W1'*e2;

        dW1 = la*dW1 + mu * e2 * x1';   % weight corrections
        db1 = la*db1 + mu * e2;
        dW0 = la*dW0 + mu * e1 * x0';
        db0 = la*db0 + mu * e1;

        if (la<1)              % pattern updating
            W1 = W1 + dW1;     % update weights
            b1 = b1 + db1;
            W0 = W0 + dW0;
            b0 = b0 + db0;
        end

    end

    if (la==1)                 % epoch updating
        W1 = W1 + dW1;         % update weights
        b1 = b1 + db1;
        W0 = W0 + dW0;
        b0 = b0 + db0;
    end

end                        % stop training iterations
```

```
mOm1m2 = [num2str(M0),':',num2str(M1),':',num2str(M2)];

J = J/J(1);

Jmin = J(end);
n = 1:Nit;
figure; plot(n,log10(J),'b-', Nit,log10(Jmin),'r.','markersize',20);
yaxis(-5,1,-5:1); grid
title([mOm1m2,', \mu = ',num2str(mu*P), '/{\itP}, \lambda=',num2str(la), ', symm=', num2str(symm)])
xlabel('epoch iterations,  {\itn}')
ylabel('log_{10}({\itJ})')
legend(' {\itJ}({\itn})', [' {\itJ}_{min} = ', num2str(Jmin,'%3.2e')], 'location','ne')


for t = 1:length(Xout),     % iterate over patterns
    x0 = Xin(:,t);          % input/output training pair

    u1 = W0*x0 + b0;        % forward pass
    x1 = f(u1);
    u2 = W1*x1 + b1;
%     x2 = f(u2);           % network output
    Xpred(t) = u2;          % outputs over the training set
end


Xin2 = X(2:end, Nt+1:end);  % prediction set
Xout2 = X(1,Nt+1:end);      % observed output of prediction set

for t = 1:length(Xout2),     % iterate over patterns
    x0 = Xin2(:,t);          % input/output training pair

    u1 = W0*x0 + b0;        % forward pass
    x1 = f(u1);
    u2 = W1*x1 + b1;
%     x2 = f(u2);           % network output
    Xpred2(t) = u2;         % predicted output
end


t = 1:length(Xout);             % time range of training set
t2 = Nt+(1:length(Xout2));      % time range of prediction horizon

figure;  plot(t,Xout,'b--', t,Xpred,'g-')     % plot training set
hold on
plot(t2,Xpred2,'m-', t2,Xout2,'b--')          % plot predictions
hold off
title([mOm1m2,', \mu = ',num2str(mu*P), '/{\itP}, \lambda=',num2str(la), ', symm=', num2str(symm)])
xlabel('years')
legend(' data', ' training set', ' prediction', 'location','nw')
yaxis(-0.6,1.2, -0.5:0.5:1.0)
xaxis(0,300,0:50:300);
```


4:4:1, μ = 0.1/P, λ=1, symm=0
J(n)
J min = 2.28e−03
log10(J)
epoch iterations, n


4:4:1, μ = 0.1/P, λ=1, symm=0
data
training set
prediction
years

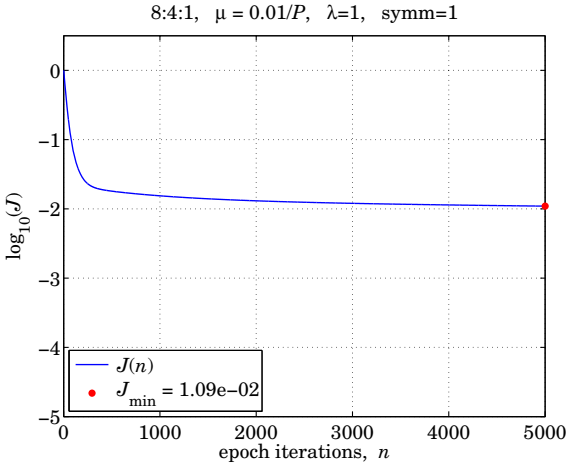
## NN prediction of airline data

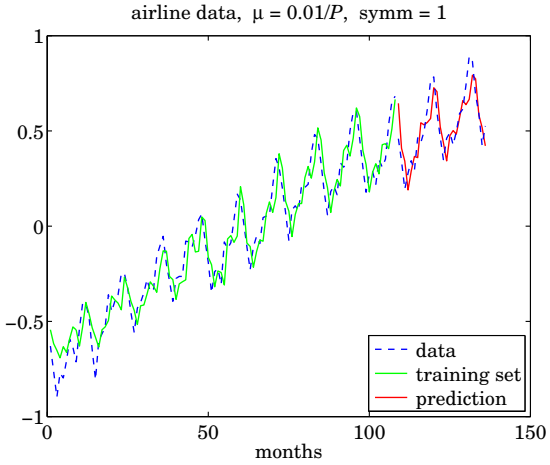



```
% s21nnair.m - NN prediction of airline data
% DSA - Spring 2021 - S. J. Orfanidis

symm = 1;                               % symmetric sigmoid

if symm==0,
    f = @(u) 1./(1+exp(-u));            % standard sigmoid
    D = @(u) diag(f(u).*(1-f(u)));      % sigmoid derivative
elseif symm==1
   f = @(u) tanh(u);                    % symmetric sigmoid
   D = @(u) diag((1-tanh(u).^2));       % sigmoid derivative
else
   c = 0.01;
   f = @(u) max(c*u,u);                 % leaky ReLU
   D = @(u) diag((u>=0) + c*(u<0));
end

Y = log(loadfile('airline.dat'));
y = Y'; y = y(:)';
yd = y;
N = length(yd); t = 0:N-1;       % N = 144 months
P0 = 36;                         % prediction distance, 36 months
n0 = N-P0;                       % training distance, 108 months
my = mean(yd);

p = 8;
X = flipud(buffer(yd-my,p+1,p,'nodelay'));

Nt = n0;
Xin = X(2:end, 1:Nt);
Xout = X(1,1:Nt);

P = size(Xin,2);              % no. patterns

M0 = size(Xin,1);             % M0:M1:M2 network
M1 = 4;                       % 8:4:1 network
M2 = size(Xout,1);

randn('state',2017);          % initial seed

A=1/2; B=0;
W0 = A*randn(M1,M0)-B;        % initialize weights & biases
b0 = A*randn(M1,1)-B;
W1 = A*randn(M2,M1)-B;
b1 = A*randn(M2,1)-B;
```

```
mu = 0.01/P;                % adaptation parameter
la = 1;        % epoch updating la=1, pattern updating la<1

Nit = 5000;                 % no. epoch iterations

for n = 1:Nit,              % training iterations

    J(n) = 0;               % performance index
    dW1=0; db1=0; dW0=0; db0=0;

    for p = 1:P,            % iterate over patterns
        x0 = Xin(:,p);      % input/output training pair
        d = Xout(:,p);

        u1 = W0*x0 + b0;    % forward pass
        x1 = f(u1);
        u2 = W1*x1 + b1;
%         x2 = f(u2);       % network output
        x2 = u2;

        J(n) = J(n) + norm(d-x2)^2;     % accumulate index

%         e2 = D(u2)*(d-x2);    % back-propagation pass
        e2 = (d-x2);        % back-propagation pass
        e1 = D(u1)*W1'*e2;

        dW1 = la*dW1 + mu * e2 * x1';    % weight corrections
        db1 = la*db1 + mu * e2;
        dW0 = la*dW0 + mu * e1 * x0';
        db0 = la*db0 + mu * e1;

        if (la<1)           % pattern updating
            W1 = W1 + dW1;      % update weights
            b1 = b1 + db1;
            W0 = W0 + dW0;
            b0 = b0 + db0;
        end

    end

    if (la==1)              % epoch updating
        W1 = W1 + dW1;      % update weights
        b1 = b1 + db1;
        W0 = W0 + dW0;
        b0 = b0 + db0;
    end

end                     % stop iterations

m0m1m2 = [num2str(M0),':',num2str(M1),':',num2str(M2)];

J = J/J(1);

Jmin = J(end);
n = 1:Nit;
figure; plot(n,log10(J),'b-', Nit,log10(Jmin),'r.','markersize',20);
yaxis(-5,0.5,-5:0); grid
title([m0m1m2,', \mu = ',num2str(mu*P), '/{\itP}, \lambda=',num2str(la), ', symm=', num2str(symm)])
xlabel('epoch iterations,  {\itn}')
ylabel('log_{10}({\itJ})')
legend(' {\itJ}({\itn})', [' {\itJ}_{min} = ', num2str(Jmin,'%3.2e')], 'location','sw')
```

```
for t = 1:length(Xout),     % iterate over patterns
    x0 = Xin(:,t);          % input/output training pair

    u1 = W0*x0 + b0;        % forward pass
    x1 = f(u1);
    u2 = W1*x1 + b1;
%     x2 = f(u2);           % network output
    Xpred(t) = u2;
end


Xin2 = X(2:end, Nt+1:end);
Xout2 = X(1,Nt+1:end);

for t = 1:length(Xout2),     % iterate over patterns
    x0 = Xin2(:,t);          % input/output training pair

    u1 = W0*x0 + b0;         % forward pass
    x1 = f(u1);
    u2 = W1*x1 + b1;
%     x2 = f(u2);            % network output
    Xpred2(t) = u2;
end

t = 1:length(Xout);          % time range of training set
t2 = Nt+(1:length(Xout2));   % time range of prediction set

figure;
plot(t,Xout,'b--', t,Xpred,'g-')
hold on
plot(t2,Xpred2,'r-', t2,Xout2,'b--')
hold off
xaxis(0,150, 0:50:150)
title('airline data,  \mu = 0.01{/\itP},  symm = 1')
xlabel('months')
legend(' data', ' training set', ' prediction', 'location','se')
```


8:4:1, μ = 0.01/P, λ=1, symm=1
log10(J)
epoch iterations, n
J(n)
J min = 1.09e−02


airline data, μ = 0.01/P, symm = 1
months
data
training set
prediction