UNIVERSITÀ DEGLI STUDI DI MILANO

Introduction to Neural Networks

Exercises

Dipartimento di Informatica
via Bramante 65, 26013 Crema (CR), Italy
ruggero.donida@unimi.it

Execise 1

One-dimensional fitting

Goals

- Implement a Matlab application based on feedforward neural networks
- Tune network parameters
- Experimentally evaluate the output variability
- Experimentally evaluate the overfitting phenomena

Problem

1Ddata.mat contains data describing the connections between the enzymes X and Y. The vectors describing the enzymes are X_train and Y_train. Feedforward neural networks should be used to learn the connections between the enzymes X and Y. The generalization capability of trained neural networks should be evaluated on X_test and Y_test.

Solution

Load and analyze data

```
load('1Ddata.mat ');

% minimum and maximum of the "enzymes"
[min(X_train')' max(X_train')']
```

Create a neural network and tune parameters

```
% create a neural network
neuronsXLayer = 5; % number of neurons per layer
neuronTopology{1} = 'logsig';
% neuronTopology{2} = 'tansig';


net_ff = feedforwardnet(neuronsXLayer);


% training and testing data
net_ff.divideParam.trainRatio = 1;
net_ff.divideParam.testRatio  = 0;
net_ff.divideParam.valRatio   = 0;

net_ff.trainParam.epochs = 100;
net_ff.trainParam.goal = 0.00001;
for iL = 1: size(neuronsXLayer,2)
    net_ff.layers{iL}.transferFcn = neuronTopology{iL};
end
```

Train the created neural network and analyze the results

```
%Training:
net_ff = train (net_ff, X_train, Y_train);

% Show NN
view(net_ff)
```

Test the accuracy

```
%Test:
Y_nn = net_ff(X_val);
% It is sufficient to call the object net_ff
% Input and output are matrices!!!


% Plot the results
figure
plot(X_train, Y_train, '+');  % training samples
hold on ;
plot(X_val, Y_nn, 'r-'); hold off  % estimated values


% Numerical results
errorT= sum(abs( net_ff(X_train)-Y_train) ) /size(Y_train,2)*100;
errorV= sum(abs( net_ff(X_val)- Y_val) ) /size(Y_val,2)*100;

disp(' ------------------------------------------------------------')
disp(' -----TRAINIG ERROR-------------------')
fprintf('Error: %6.2f %%\n',errorT);
disp(' -----VALIDATION ERROR-------------------')
fprintf('Error: %6.2f %%\n',errorV);
disp(' ------------------------------------------------------------')
```

Tests

- Analyze the results
- Re-run the script and evaluate the results
- Different number of neurons
- Different neuron topology
- Different number of hidden layers
- Different number training epochs
- Different threshold
- Overfitting phenomena

Execise 2

Two-dimensional fitting

Individual test

Problem

2Ddata.mat contains data describing the connections of the enzymes X1 and X2 with Y. The matrices describing the enzymes are X_train and Y_train. Feedforward neural networks should be used to learn the connections between the enzymes X and Y. The generalization capability of trained neural networks should be evaluated on X_test and Y_test.

Use `plot(x,y)` to analyze the results.

Suggestions

Many parts of the script are similar to the previous one.

Execise 3

N-dimensional fitting

Goals

- Use of a more formal notation
- Divide datasets into training and testing sets
- Analysis of N-dimensional problems
- Accuracy evaluation for N-dimensional datasets

Problem

LOAD bodyfat_dataset.

Train a neural network to estimate the bodyfat of someone from various measurements.
Bodyfat_dataset contains two variables.

1) bodyfatInputs - a 13x252 matrix defining thirteen attributes for 252 people:
    1. Age (years)
    2. Weight (lbs)
    3. Height (inches)
    4. Neck circumference (cm)
    5. Chest circumference (cm)
    6. Abdomen 2 circumference (cm)
    7. Hip circumference (cm)
    8. Thigh circumference (cm)
    9. Knee circumference (cm)
    10. Ankle circumference (cm)
    11. Biceps (extended) circumference (cm)
    12. Forearm circumference (cm)
    13. Wrist circumference (cm)

2) bodyfatTargets - a 1x252 matrix of associated body fat percentages, to be estimated from the inputs.

Solution

Load and analyze data

```
load('bodyfat_dataset.mat');
```

P & T

```
% INPUT vector
P = bodyfatInputs;
% TARGET vector
```

```
T = bodyfatTargets;
```

Create a neural network and tune parameters

```
% create a neural network
neuronsXLayer = 5; % number of neurons per layer
neuronTopology{1} = 'logsig';
% neuronTopology{2} = 'tansig';

net.trainParam.epochs = 100;
net.trainParam.goal = 0.00001;
for iL = 1: size(neuronsXLayer,2)
    net.layers{iL}.transferFcn = neuronTopology{iL};
end



net = feedforwardnet(neuronsXLayer);
```

Divide the input dataset into two subsets

```
%-----------------------------------------
net.divideParam.trainRatio = 0.5;
net.divideParam.testRatio  = 0.5;
net.divideParam.valRatio   = 0;
%-----------------------------------------
```

Train the created neural network and analyze the results

```
% train a neural network
[net,tr,Y,E] = train(net,P,T);


% show network
view(net)

plotperform(tr)
```

Compute the training and testing results

```
% Training and testing data
trainP = P(:,tr.trainInd);
trainT = T(:,tr.trainInd);
trainResult = net(trainP);


testP = P(:,tr.testInd);
testT = T(:,tr.testInd);
testResult = net(testP);
```

Plot the results

```
%plot
figure,
plot(trainT)
hold on
plot(trainResult, '--r')
title('training results')

figure,
plot(testT)
hold on
plot(testResult, '--r')
title('testing results')
```

Compute the error using the required figures of metric

```
trainErr = abs(trainT - trainResult);
testErr = abs(testT - testResult);
```

Compute figures of merit

```
%Results
numTrainErr = size(find(trainErr > 0),2);
meanTrainErr = mean(trainErr);
stdTrainErr = std(trainErr);


numTestErr = size(find(testErr > 0),2);
meanTestErr = mean(testErr);
stdTestErr = std(testErr);


totalErr = [trainErr, testErr];
numTotalErr = size(find(totalErr > 0),2);
meanTotalErr = mean(totalErr);
```

```
stdTotalErr = std(totalErr);


fprintf('\nRESULTS \n')
fprintf('TRAINING\n')
fprintf('n. of el. = %i \n', size(trainErr,2))
fprintf('n. of errors = %i \n', numTrainErr)
fprintf('mean error = %f \n', meanTrainErr)
fprintf('std error = %f \n\n', stdTrainErr)
fprintf('TESTING\n')
fprintf('n. of el. = %i \n', size(testErr,2))
fprintf('n. of errors = %i \n', numTestErr)
fprintf('mean error = %f \n', meanTestErr)
fprintf('std error = %f \n\n', stdTestErr)
fprintf('TOTAL\n')
fprintf('n. of el. = %i \n', size(totalErr,2))
fprintf('n. of errors = %i \n', numTotalErr)
fprintf('mean error = %f \n', meanTotalErr)
fprintf('std error = %f \n\n', stdTotalErr)
```