Mathematics

AIMS Mathematics, 9(9): 25223–25231.
DOI: 10.3934/math.20241230
Received: 27 August 2024
Revised: 27 August 2024
Accepted: 28 August 2024
Published: 29 August 2024

https://www.aimspress.com/journal/Math

Correction

Correction: Developing mathematical models and intelligent sustainable supply chains by uncertain parameters and algorithms

Massoumeh Nazari[1], Mahmoud Dehghan Nayeri[2,*] and Kiamars Fathi Hafshjani[1]

[1] Department of Industrial Management, Islamic Azad University, Tehran South Branch, Faculty of Management, Tehran, Iran

[2] Department of Industrial Management, Tarbiat Modares University, Faculty of Management and Economics, Tehran, Iran

* **Correspondence:** Email: mdnayeri@modares.ac.ir.

A correction on

Developing mathematical models and intelligent sustainable supply chains by uncertain parameters and algorithms
by Massoumeh Nazari, Mahmoud Dehghan Nayeri and Kiamars Fathi Hafshjani. AIMS Mathematics, 2024, 9(3): 5204–5233. DOI: 10.3934/math.2024252

The authors would like to revise a small mistake in Figure 1 by changing the direction of the fifth elbow arrow from h to j, and add the artificial intelligence codes to Appendix section of the published paper [1]. The updated Figure 1 and Appendix are as follows,

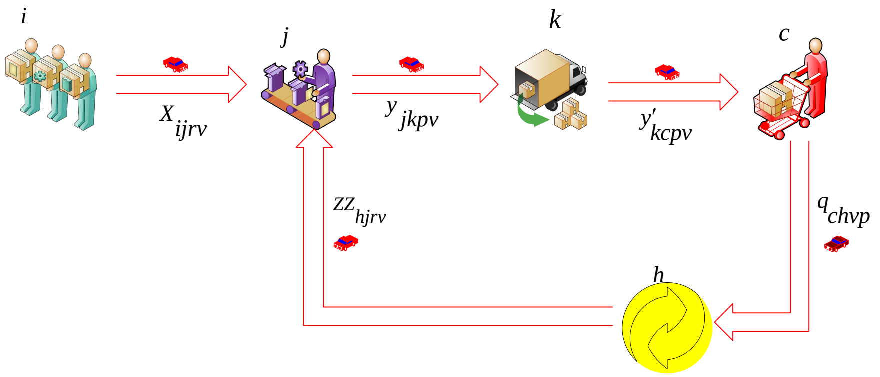

Figure 1. Conceptual models.

Appendix

A.1. The shipping cost parameter from the polymer material supplier to the factory

```
T = tonndata(y,true,false);
trainFcn = 'trainlm';
feedbackDelays = 1:3;
hiddenLayerSize = 25;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);
net.input.processFcns = {'removeconstantrows','mapminmax'};
 [x,xi,ai,t] = preparets(net,{},{},T);
net.divideFcn = 'dividerand';
net.divideMode = 'time';
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 20/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate', 'ploterrhist', ...
   'plotregression', 'plotresponse', 'ploterrcorr', 'plotinerrcorr'};
 [net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
view(net)
netc = closeloop(net);
```

```
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,{},{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)
 [x1,xio,aio,t] = preparets(net,{},{},T);
[y1,xfo,afo] = net(x1,xio,aio);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(cell(0,5),xic,aic);
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,{},{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)
if (false)
   genFunction(net,'myNeuralNetworkFunction');
   y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
   genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
   x1 = cell2mat(x(1,:));
   xi1 = cell2mat(xi(1,:));
   y = myNeuralNetworkFunction(x1,xi1);
end
if (false)
   gensim(net);
end
```

A.2. The shipping costs from the factory to the distributors of automotive plastic accessories

```
T = tonndata(y,true,false);
trainFcn = 'trainlm';
feedbackDelays = 1:3;
hiddenLayerSize = 20;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);
net.input.processFcns = {'removeconstantrows','mapminmax'};
 [x,xi,ai,t] = preparets(net,{},{},T);
net.divideFcn = 'dividerand';
net.divideMode = 'time';
net.divideParam.trainRatio = 85/100;
net.divideParam.valRatio = 5/100;
net.divideParam.testRatio = 10/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate', 'ploterrhist', ...
    'plotregression', 'plotresponse', 'ploterrcorr', 'plotinerrcorr'};
 [net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
view(net)
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,{},{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)
 [x1,xio,aio,t] = preparets(net,{},{},T);
[y1,xfo,afo] = net(x1,xio,aio);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(cell(0,5),xic,aic);
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,{},{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)
if (false)
    genFunction(net,'myNeuralNetworkFunction');
    y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
    genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    x1 = cell2mat(x(1,:));
    xi1 = cell2mat(xi(1,:));
    y = myNeuralNetworkFunction(x1,xi1);
end
if (false)
    gensim(net);
end
```

A.3. The shipping costs from the distributors of automotive plastic accessories to the customers of plastic car accessories

```
T = tonndata(y,true,false);
trainFcn = 'trainlm';
feedbackDelays = 1:3;
hiddenLayerSize = 27;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);
net.input.processFcns = {'removeconstantrows','mapminmax'};
 [x,xi,ai,t] = preparets(net,{},{},T);
net.divideFcn = 'dividerand';
net.divideMode = 'time';
net.divideParam.trainRatio = 85/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 5/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate', 'ploterrhist', ...
   'plotregression', 'plotresponse', 'ploterrcorr', 'plotinerrcorr'};
 [net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
view(net)
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,{},{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)
 [x1,xio,aio,t] = preparets(net,{},{},T);
[y1,xfo,afo] = net(x1,xio,aio);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(cell(0,5),xic,aic);
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,{},{},T);
ys = nets(xs,xis,ais);
```

```
stepAheadPerformance = perform(nets,ts,ys)
if (false)
   genFunction(net,'myNeuralNetworkFunction');
   y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
   genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
   x1 = cell2mat(x(1,:));
   xi1 = cell2mat(xi(1,:));
   y = myNeuralNetworkFunction(x1,xi1);
end
if (false)
   gensim(net);
end
```

A.4. The shipping costs from the customers of plastic car accessories to the recycling center

```
T = tonndata(x4,false,false);
trainFcn = 'trainlm';
feedbackDelays = 1:3;
hiddenLayerSize = 42;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);
net.input.processFcns = {'removeconstantrows','mapminmax'};
 [x,xi,ai,t] = preparets(net,{},{},T);
net.divideFcn = 'dividerand';
net.divideMode = 'time';
net.divideParam.trainRatio = 75/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 10/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate', 'ploterrhist', ...
   'plotregression', 'plotresponse', 'ploterrcorr', 'plotinerrcorr'};
 [net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
view(net)
netc = closeloop(net);
```

```
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,{},{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)
 [x1,xio,aio,t] = preparets(net,{},{},T);
[y1,xfo,afo] = net(x1,xio,aio);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(cell(0,5),xic,aic);
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,{},{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)
if (false)
   genFunction(net,'myNeuralNetworkFunction');
   y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
   genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
   x1 = cell2mat(x(1,:));
   xi1 = cell2mat(xi(1,:));
   y = myNeuralNetworkFunction(x1,xi1);
end
if (false)
   gensim(net);
end
```

A.5. The shipping costs from the recycling center to the factory

```
T = tonndata(x5,false,false);
feedbackDelays = 1:3;
hiddenLayerSize = 38;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);
net.input.processFcns = {'removeconstantrows','mapminmax'};
 [x,xi,ai,t] = preparets(net,{},{},T);
net.divideFcn = 'dividerand';
net.divideMode = 'time';
net.divideParam.trainRatio = 45/100;
net.divideParam.valRatio = 35/100;
net.divideParam.testRatio = 20/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate', 'ploterrhist', ...
```

```
   'plotregression', 'plotresponse', 'ploterrcorr', 'plotinerrcorr'};
 [net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
view(net)
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,{},{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)
 [x1,xio,aio,t] = preparets(net,{},{},T);
[y1,xfo,afo] = net(x1,xio,aio);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(cell(0,5),xic,aic);
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,{},{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)
if (false)
   genFunction(net,'myNeuralNetworkFunction');
   y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
   genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
   x1 = cell2mat(x(1,:));
   xi1 = cell2mat(xi(1,:));
   y = myNeuralNetworkFunction(x1,xi1);
end
if (false)
   gensim(net);
end
```

The changes have no material impact on the conclusion of this article. The original manuscript will be updated [1].

Conflict of interest

The authors declare no conflicts of interest.

References

1. M. Nazari, M. D. Nayeri, K. F. Hafshjani, Developing mathematical models and intelligent sustainable supply chains by uncertain parameters and algorithms, *AIMS Math.*, **9** (2024), 5204–5233. https://doi.org/10.3934/math.2024252