AMS-512 Capital Markets and Portfolio Theory

Fitting a Factor Model by MLE and Neural Networks Using Simulated Data.

Robert J. Frey, Research Professor
Stony Brook University, Applied Mathematics and Statistics

Robert.Frey@StonyBrook.edu
http://www.ams.sunysb.edu/~frey

Overview

This notebook is a comparison of factor model fits using MLE and neural networks. The simulated data are 2,000 observations of 20 assets whose returns are driven by three factors. Half of the data is used for model fitting and half for out-of-sample testing. The factor returns and error returns in the generative model are both multivariate Normal. Both the factors and the errors are all mutually uncorrelated.

Given that there is an infinite number of ways a given space can be spanned by factors, we focus on how well the model represents the out-of-sample returns. We do, however, also discuss the differences between the two fits and make some preliminary suggestions as to how they may be reconciled.

Set Up

In[ ]:=
```
<< Local`SaveRecall`
<< Local`FactorFitMLE`
```

Generative Model

Mathematics

$$r_i(t) = \alpha_i + \sum_{j=1}^{m} b_{i,j} f_j(t) + \epsilon(t) \quad i = 1, \ldots, n$$

$$\mathbf{r}(t) = \alpha + \mathbf{B}^T \mathbf{f}(t) + \boldsymbol{\epsilon}(t)$$

$$\mathbf{R} = \mathbf{A} + \mathbf{B}\,\mathbf{F}^T + \mathbf{E}$$

$$\mu = \alpha + \mathbf{B}\,\phi$$

$$\Sigma = \mathbf{B}\,\mathbf{C}\,\mathbf{B}^T + \mathbf{D}$$

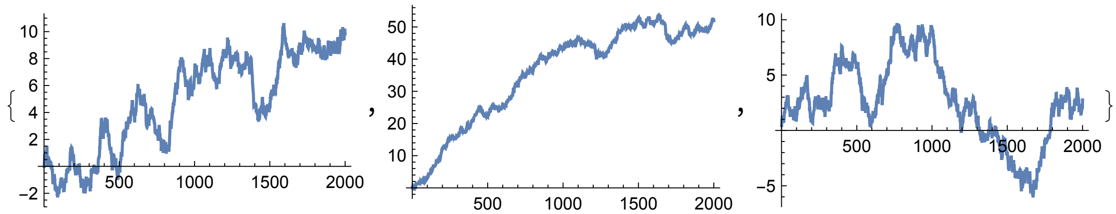
Simulation

In[∘]:=
```
mnFactorReturns = RandomVariate[MultinormalDistribution[
    {0.008, 0.012, 0.010}, DiagonalMatrix[{0.2, 0.3, 0.25}^2]], 2000];
Dimensions@mnFactorReturns
```

Out[∘]=
```
{2000, 3}
```

In[∘]:=
```
ListLinePlot[Accumulate@#] & /@ Transpose[mnFactorReturns]
```

Out[∘]=



In[∘]:=
```
mnFactorLoadings = RandomVariate[UniformDistribution[{-0.25, 0.25}], {20, 3}];
Dimensions@mnFactorLoadings
```

Out[∘]=
```
{20, 3}
```

In[∘]:=
```
mnErrors = RandomVariate[NormalDistribution[0, 0.03], {2000, 20}];
Dimensions@mnErrors
```

Out[∘]=
```
{2000, 20}
```

In[∘]:=
```
vnAlphas = RandomVariate[UniformDistribution[{-0.005, 0.005}], 20];
```

In[∘]:=
```
mnAssetReturns =
  vnAlphas + # & /@ (mnFactorReturns.Transpose[mnFactorLoadings] + mnErrors);
Dimensions@mnAssetReturns
```

Out[∘]=
```
{2000, 20}
```

Training-Test Separation

In[∘]:=
```
mnTraining = mnAssetReturns[[ ;; 1000]];
Dimensions@mnTraining
mnTest = mnAssetReturns[[1001 ;;]];
Dimensions@mnTest
```

Out[∘]=
```
{1000, 20}
```

Out[∘]=
```
{1000, 20}
```

MLE Factor Model

In[•]:=
```
vnMeanMLE = Mean[mnTraining]
```

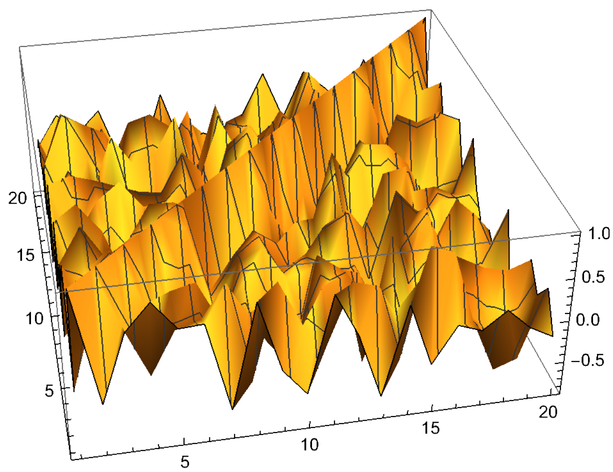
Out[•]=
```
{-0.0091148, 0.00677535, 0.00196628, -0.00737638, -0.00350378,
 0.00197924, 0.00338466, -0.00441579, 0.00625267, 0.00319213,
 -0.00317594, 0.00287809, 0.00918268, -0.00860375, 0.0136834,
 0.00512195, -0.000152026, -0.00955773, -0.00368295, 0.00459397}
```

In[•]:=
```
mnCovMLE = Covariance[mnTraining];
```

In[•]:=
```
mnCorMLE = Correlation[mnTraining];
```

In[•]:=
```
ListPlot3D[mnCorMLE, PlotRange → All]
```

Out[•]=



In[•]:=
```
nObs = Length[mnTraining]
```

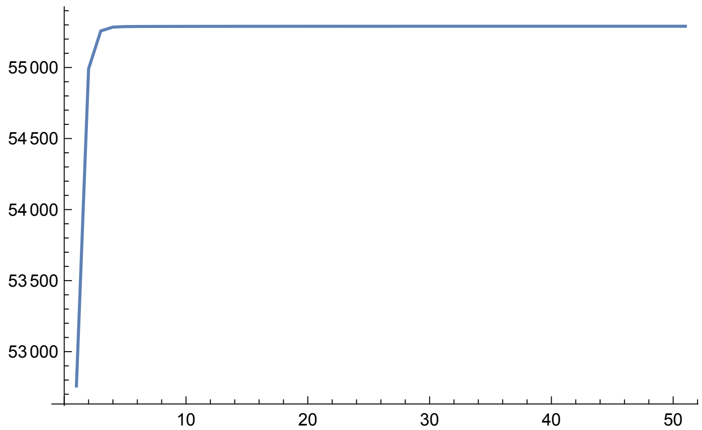
Out[•]=
```
1000
```

In[•]:=
```
iOrder = 3;
```

In[•]:=
```
{{mnFactorLoadingsMLE, mnErrorMatrixMLE}, vnLLHistory, nBIC} =
  xFactorFitMLE[nObs, mnCovMLE, xInitializeFactorModel[mnCovMLE, iOrder]];
```

In[∘]:=
```
ListLinePlot[vnLLHistory, PlotRange → All]
```

Out[∘]=



Reconstructions

In[∘]:=
```
xInverse[mnB_, mnD_] := Module[
    {mnID},
    mnID = DiagonalMatrix[1 / Tr[mnD, List]];
    mnID - mnID.mnB.
      Inverse[IdentityMatrix[Last[Dimensions[mnB]]] + mnBᵀ.mnID.mnB].mnBᵀ.mnID
   ];
```

In[∘]:=
```
vnFactorMeanMLE = Inverse[
    Transpose[mnFactorLoadingsMLE].Inverse[mnErrorMatrixMLE].mnFactorLoadingsMLE].
   Transpose[mnFactorLoadingsMLE].Inverse[mnErrorMatrixMLE].vnMeanMLE
```

Out[∘]=
```
{0.13442, 0.0231493, -0.0727659}
```

In[∘]:=
```
mnFactorReturnsMLE =
   (((mnFactorLoadingsMLEᵀ.xInverse[mnFactorLoadingsMLE, mnErrorMatrixMLE]).
        (# - vnMeanMLE & /@ mnTraining)ᵀ) + vnFactorMeanMLE)ᵀ;
```

In[∘]:=
```
vnPredMeanMLE = mnFactorLoadingsMLE.vnFactorMeanMLE
```

Out[∘]=
```
{-0.0100823, 0.0101782, 0.000535079, -0.00745652, -0.000283632,
 0.000418952, 0.00721491, -0.00552162, 0.00530291, 0.00357713,
 -0.00237757, 0.00359208, 0.00837568, -0.00590503, 0.00906559,
 0.00621896, -0.000492863, -0.00665956, -0.00548896, 0.00196353}
```

In[∘]:=
```
vnAlphasMLE = vnMeanMLE - vnPredMeanMLE
```

Out[∘]=
```
{0.000967465, -0.00340283, 0.0014312, 0.0000801391, -0.00322014,
 0.00156029, -0.00383025, 0.00110583, 0.00094976, -0.000384993,
 -0.000798375, -0.000713985, 0.000807, -0.00269872, 0.00461785,
 -0.00109701, 0.000340837, -0.00289817, 0.00180601, 0.00263044}
```

Autoencoder NN

Topology

In[∘]:=
```
netModel = NetGraph[
  {LinearLayer[3, "Input" → 20], LinearLayer[20], MeanSquaredLossLayer[]},
  {1 → 2 → NetPort["Output"], 2 → NetPort[3, "Input"],
   NetPort["Input"] → NetPort[3, "Target"], 1 → NetPort["Factor"]}
 ]
```

Out[∘]=
```
NetGraph[ uninitialized
  Input port:          vector (size: 20)
  Number of outputs:   3
  Number of layers:    3 ]
```

Initialization

In[∘]:=
```
netFactor = NetInitialize[netModel]
```

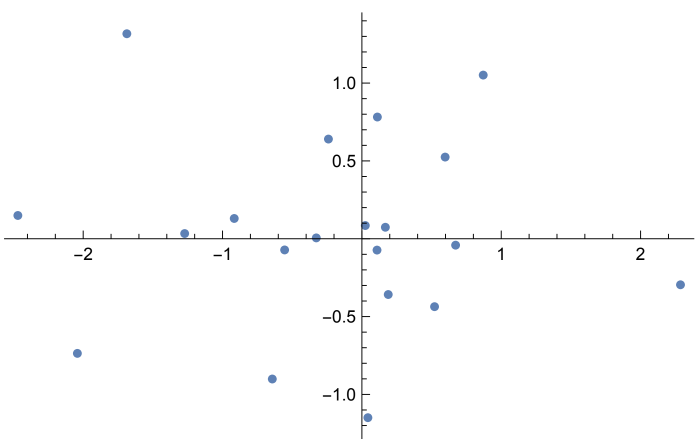
Out[∘]=
```
NetGraph[
  Input port:          vector (size: 20)
  Number of outputs:   3
  Number of layers:    3 ]
```

In[∘]:=
```
x = RandomVariate[NormalDistribution[], {20}]
```

Out[∘]=
```
{0.521437, 0.67249, -2.04173, -0.642919, -0.916029, -0.327242,
 -0.554383, -1.27178, -1.68671, -2.46877, 2.28635, 0.108559, 0.597468,
 0.168467, -0.240104, 0.18907, 0.0244668, 0.111205, 0.0430716, 0.870715}
```

In[∘]:=
```
ListPlot[{x, netFactor[x]["Output"]}^T, PlotRange → All]
```

Out[∘]=

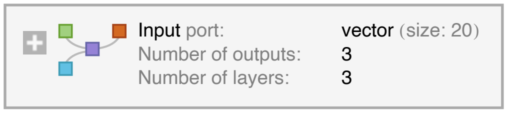


In[∘]:=
```
netFactor[x]["Factor"]
```

Out[∘]=
```
{-1.99053, 0.642039, 1.15202}
```

In[•]:=
```
netFactor[x]["Loss"]
```
Out[•]=
```
1.59568
```

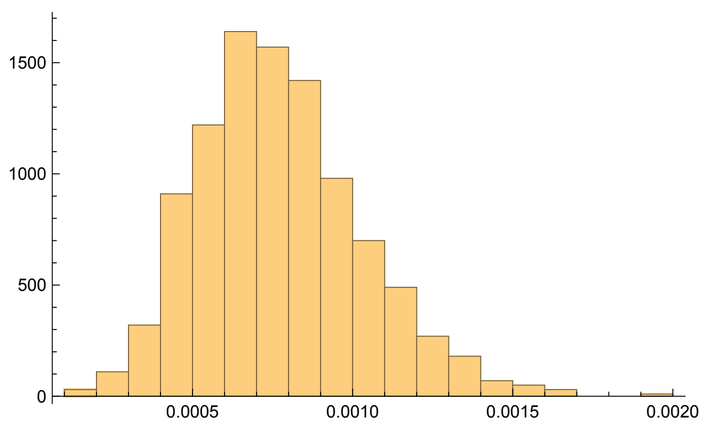
Training

In[•]:=
```
netFactor = NetTrain[netFactor, <|"Input" → mnTraining|>]
```
Out[•]= NetGraph[]

Reconstruction

In[•]:=
```
dsInSample = Dataset[netFactor /@ mnTraining];
```
In[•]:=
```
mnInSample = Normal[dsInSample[All, "Output"]];
vnInSampleLoss = Normal[dsInSample[All, "Loss"]];
mnInSampleFactors = Normal[dsInSample[All, "Factor"]];
```

Comparisons (In Sample)

In[•]:=
```
mnFitMLE =
  Table[vnAlphasMLE[[i]] + mnFactorLoadingsMLE[[i]].mnFactorReturnsMLEᵀ, {i, 1, 20}]ᵀ;
```
In[•]:=
```
Histogram[vnInSampleLoss, Automatic, "PDF"]
```
Out[•]=


Factor Correlations

Note that, as expected, the MLE produces uncorrelated factors while the NN does not.

In[•]:=
```
MatrixForm@Correlation[mnInSampleFactors]
```
Out[•]//MatrixForm=

$$\begin{pmatrix} 1. & -0.595648 & -0.463073 \\ -0.595648 & 1. & 0.690383 \\ -0.463073 & 0.690383 & 1. \end{pmatrix}$$

In[∘]:=
```
MatrixForm@Correlation[mnFactorReturnsMLE]
```

Out[∘]//MatrixForm=

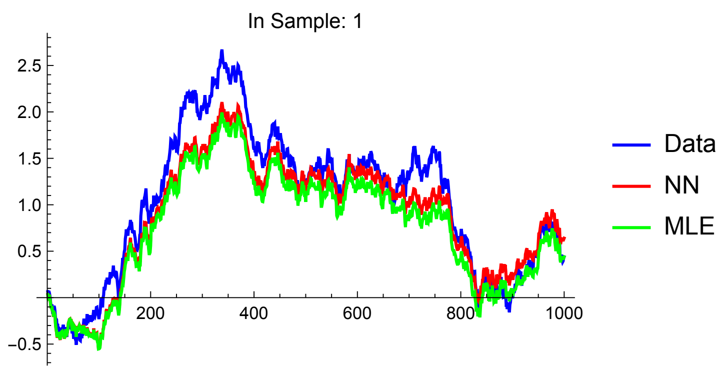
$$\begin{pmatrix} 1. & 0.000232739 & -0.000247916 \\ 0.000232739 & 1. & -0.000714157 \\ -0.000247916 & -0.000714157 & 1. \end{pmatrix}$$

Plots

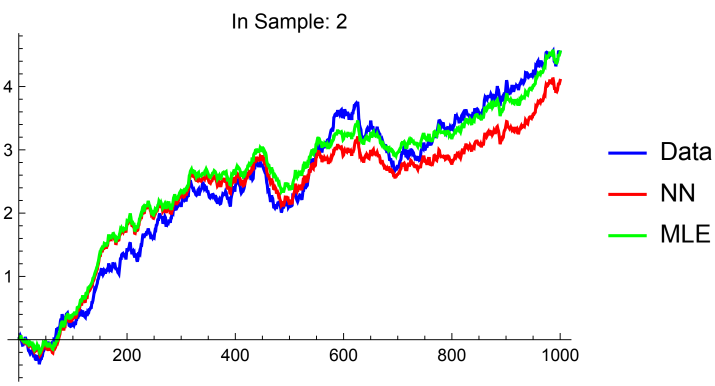
The two models produce essentially the same results when comparing the models’ fits to the original data.

In[∘]:=
```
Table[
 ListLinePlot[
  {Accumulate@mnTraining[[All, i]], Accumulate@mnInSample[[All, i]],
   Accumulate@mnFitMLE[[All, i]]}, PlotRange → All,
  PlotLabel → "In Sample: " <> ToString[i], PlotStyle → {{Blue}, {Red}, {Green}},
  ImageSize → 300, PlotLegends → {"Data", "NN", "MLE"}],
 {i, 1, 20}
]
```

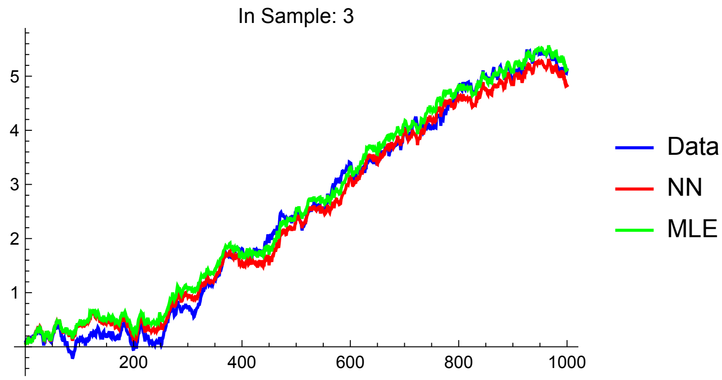
Out[∘]= {



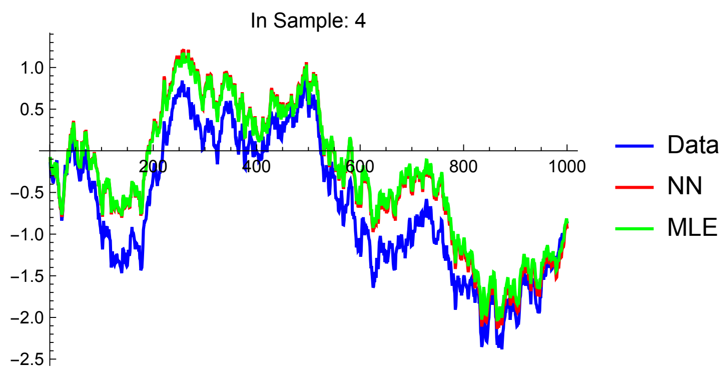
,



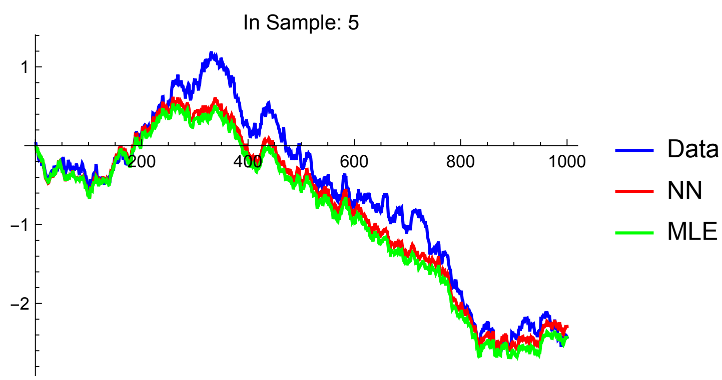
,


In Sample: 3
Data
NN
MLE

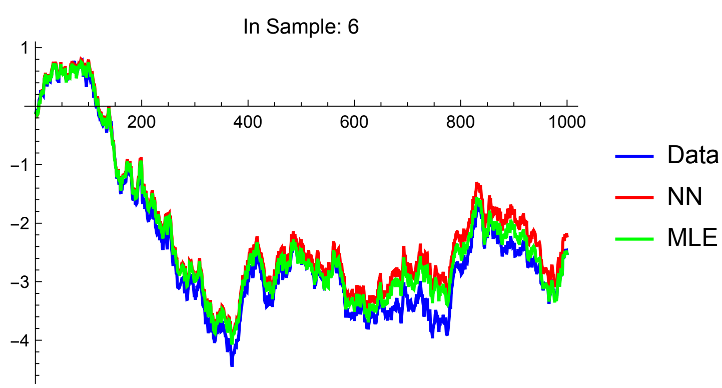
,


In Sample: 4
Data
NN
MLE

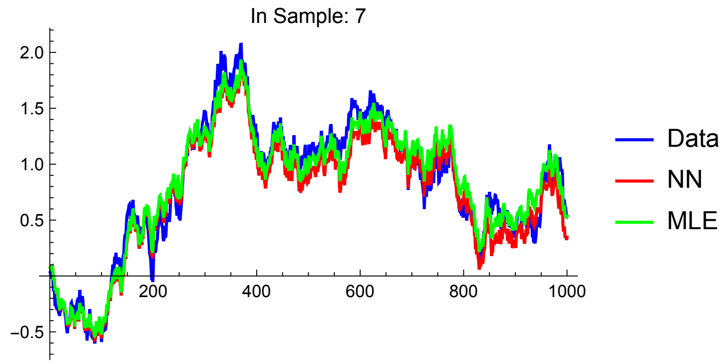
,

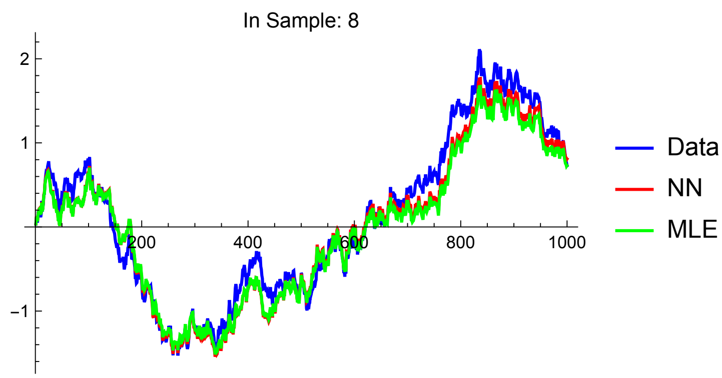

In Sample: 5
Data
NN
MLE

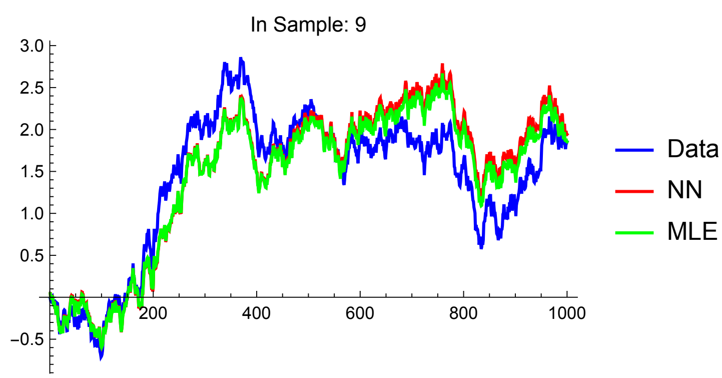
,

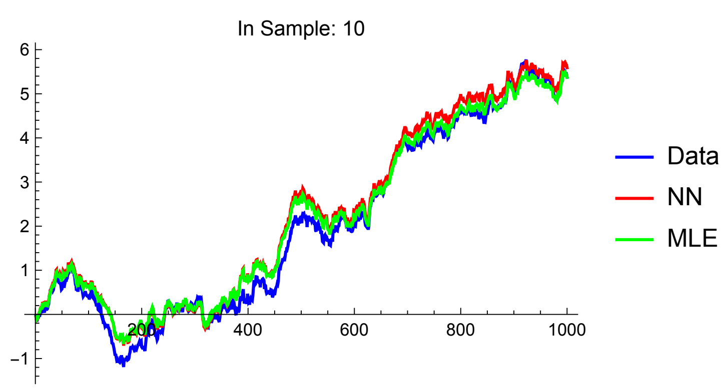

In Sample: 6
Data
NN
MLE

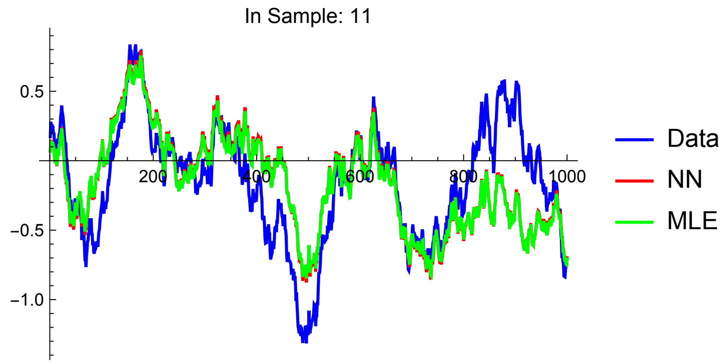
,


In Sample: 7
Data
NN
MLE
,

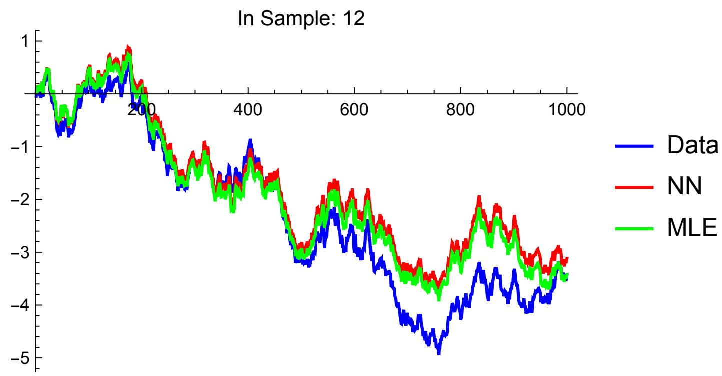

In Sample: 8
Data
NN
MLE
,


In Sample: 9
Data
NN
MLE
,

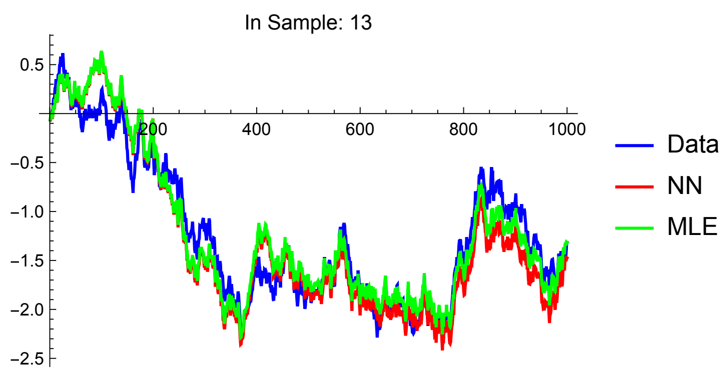

In Sample: 10
Data
NN
MLE
,


In Sample: 11
Data
NN
MLE

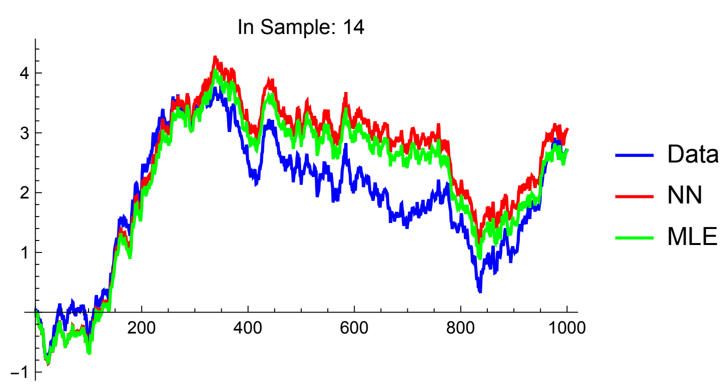
,


In Sample: 12
Data
NN
MLE

,

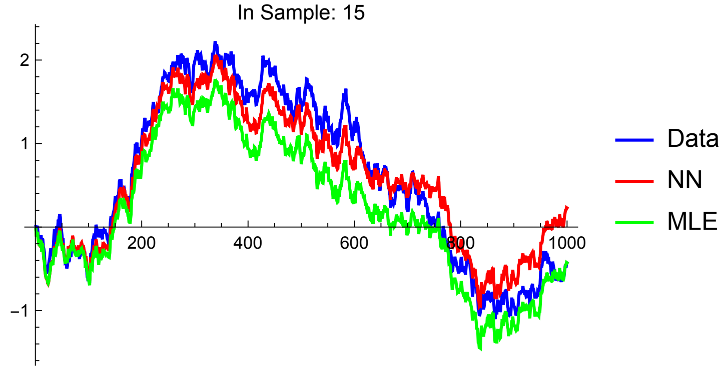

In Sample: 13
Data
NN
MLE

,


In Sample: 14
Data
NN
MLE

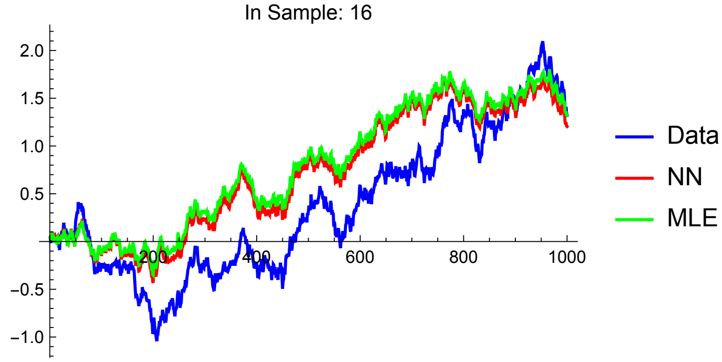
,

In Sample: 15



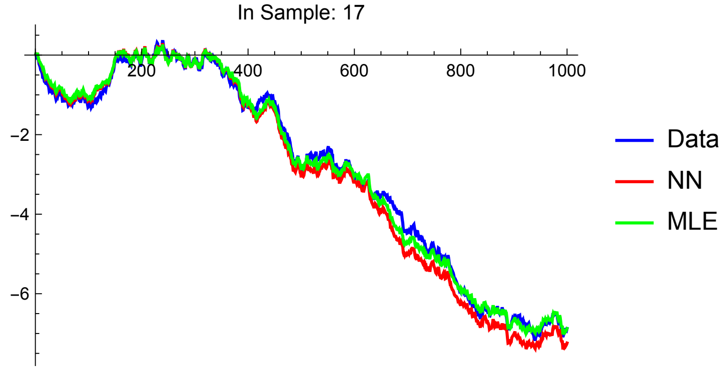
,

In Sample: 16



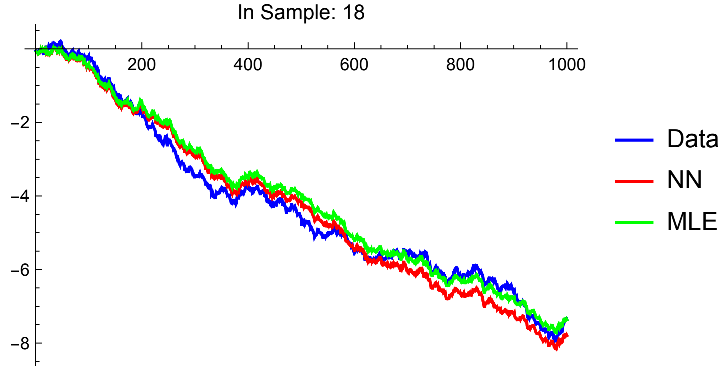
,

In Sample: 17

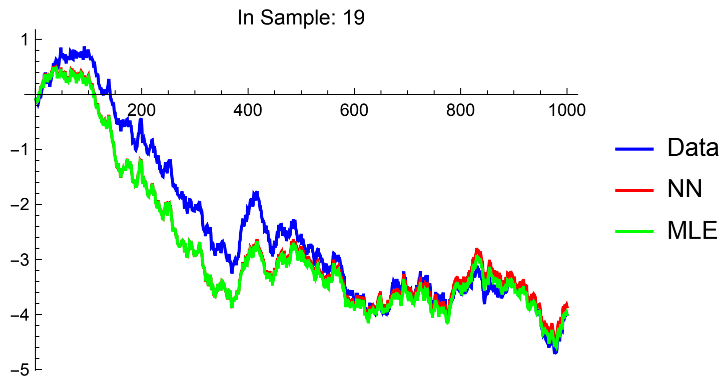


,

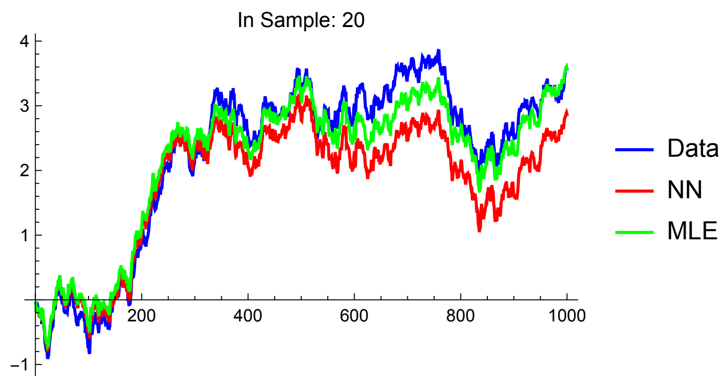
In Sample: 18



,



,



}

Out of Sample

Factor Model MLE

In[•]:=
```
vnOOSMeanMLE = Mean[mnTest]
```

Out[•]=
```
{-0.000876323, 0.00100963, 0.00158499, 0.00281455, -0.00431954,
 0.00402954, -0.00177881, -0.00257824, 0.000249026, 0.00831942,
 -0.00309122, -0.00872581, 0.00237611, 0.00266113, 0.0000938734,
 -0.00040603, -0.00720275, -0.00315773, -0.000754216, 0.00642138}
```

In[•]:=
```
mnOOSFactorReturnsMLE =
  (((mnFactorLoadingsMLEᵀ.xInverse[mnFactorLoadingsMLE, mnErrorMatrixMLE]).
       (# - vnMeanMLE & /@ mnTest)ᵀ) + vnFactorMeanMLE)ᵀ;
```

In[•]:=
```
vnOOSPredMeanMLE = mnFactorLoadingsMLE.vnFactorMeanMLE
```

Out[•]=
```
{0.000957328, -0.00226101, 0.00301583, -0.000221399, -0.000256967,
 -0.00274823, 0.00206702, 0.000546599, 0.00349874, 0.00267843,
 -0.000829141, -0.00447709, -0.0034064, -0.00021385, -0.000645246,
 0.00286982, -0.00190572, -0.00179925, -0.00177932, 0.000756929}
```

```
In[•]:= vnOOSAlphasMLE = vnMeanMLE - vnOOSPredMeanMLE

Out[•]= {-0.000517777, 0.00680804, 0.00210239, -0.000632976, -0.00217255,
         0.000232772, -0.00153119, 0.000183639, -0.00164576, 0.00269598,
         0.000122069, 0.00105406, 0.0020899, 0.00292139, 0.000215715,
         -0.00154895, -0.00496882, -0.00556716, -0.00221856, 0.00280961}
```

```
In[•]:= mnOOSFitMLE = Table[
           vnOOSAlphasMLE⟦i⟧ + mnFactorLoadingsMLE⟦i⟧.mnOOSFactorReturnsMLE^T, {i, 1, 20}]^T;
```

Neural Network

```
In[•]:= dsOutOfSample = Dataset[netFactor /@ mnTest];
```

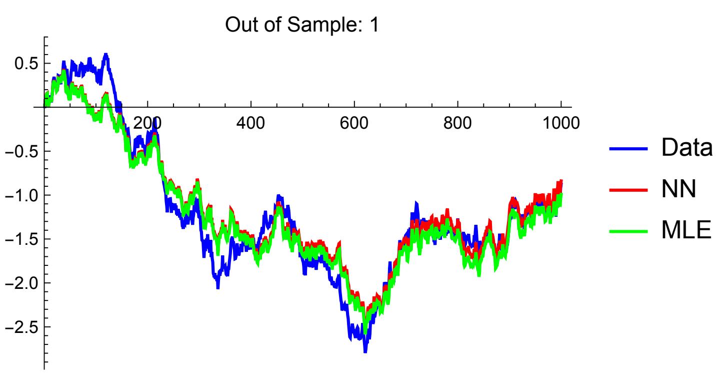
```
In[•]:= mnOutOfSample = Normal[dsOutOfSample[All, "Output"]];
        vnOutOfSampleLoss = Normal[dsOutOfSample[All, "Loss"]];
        mnOutOfSampleFactors = Normal[dsOutOfSample[All, "Factor"]];
```

Comparisons

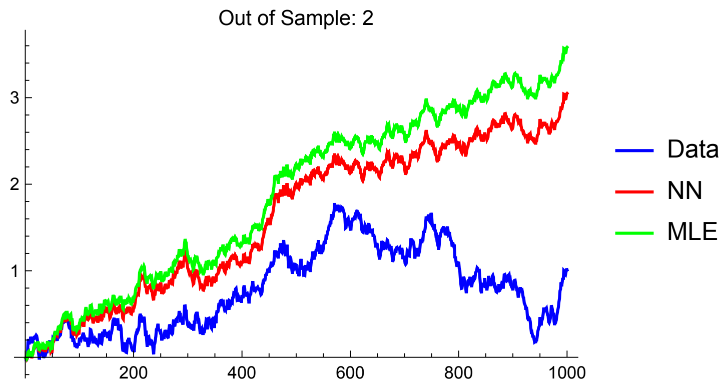
Reconstruction Plots

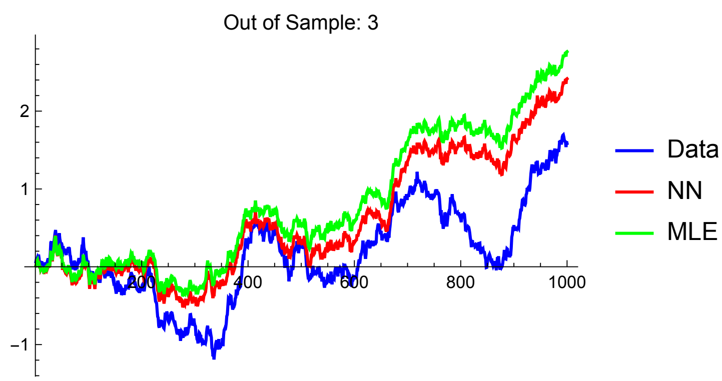
```
In[•]:= Table[
          ListLinePlot[
           {Accumulate@mnTest⟦All, i⟧, Accumulate@mnOutOfSample⟦All, i⟧,
            Accumulate@mnOOSFitMLE⟦All, i⟧}, PlotRange → All,
           PlotLabel → "Out of Sample: " <> ToString[i], PlotStyle → {{Blue}, {Red}, {Green}},
           ImageSize → 300, PlotLegends → {"Data", "NN", "MLE"}],
          {i, 1, 20}
        ]
```

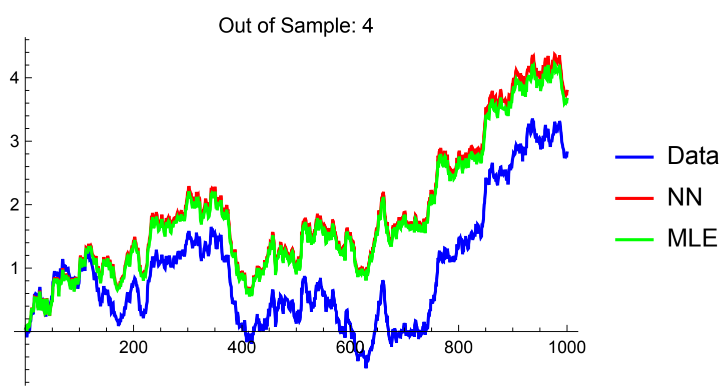
Out[•]= {

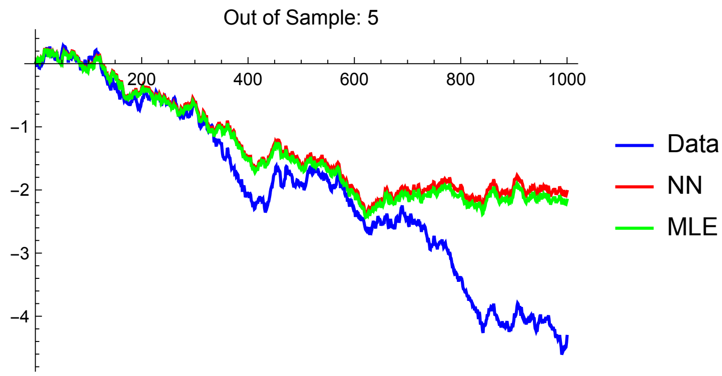


,

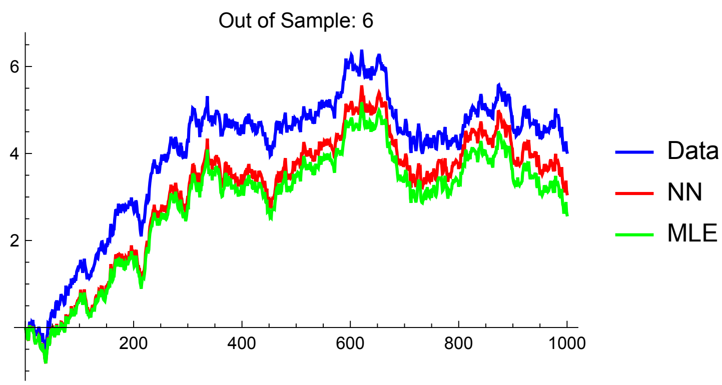

Out of Sample: 2
3
2
1
200
400
600
800
1000
Data
NN
MLE
,


Out of Sample: 3
2
1
-1
200
400
600
800
1000
Data
NN
MLE
,

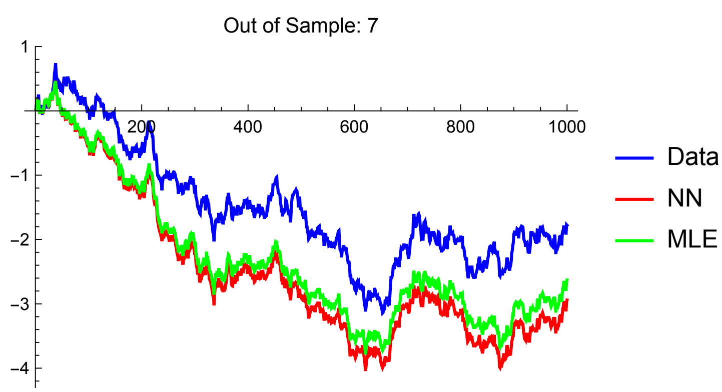

Out of Sample: 4
4
3
2
1
200
400
600
800
1000
Data
NN
MLE
,


Out of Sample: 5
200
400
600
800
1000
−1
−2
−3
−4
Data
NN
MLE

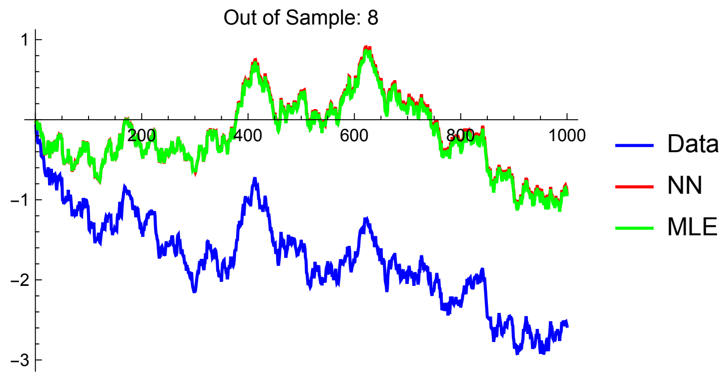
,


Out of Sample: 6
6
4
2
200
400
600
800
1000
Data
NN
MLE

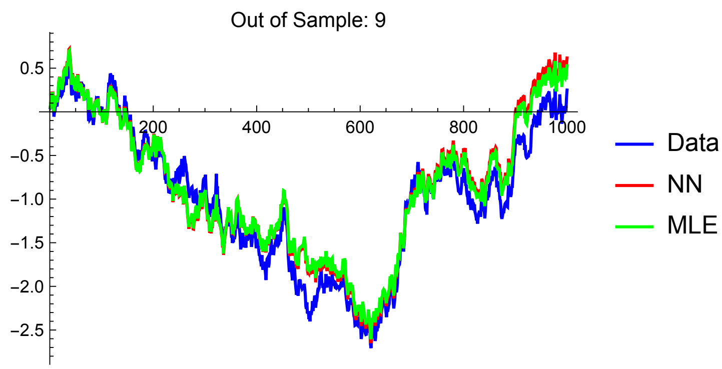
,


Out of Sample: 7
1
200
400
600
800
1000
−1
−2
−3
−4
Data
NN
MLE

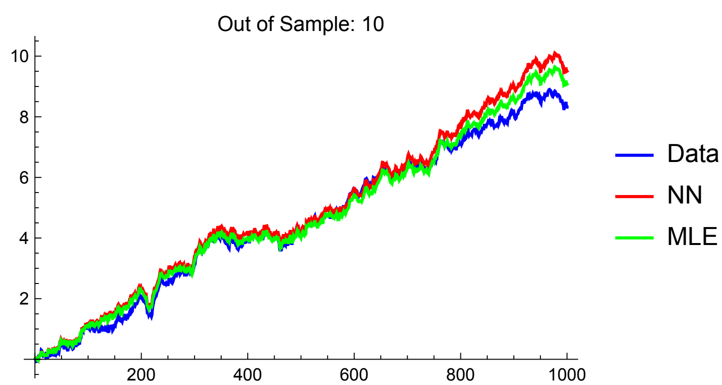
,


Out of Sample: 8
1
−1
−2
−3
200
400
600
800
1000
Data
NN
MLE

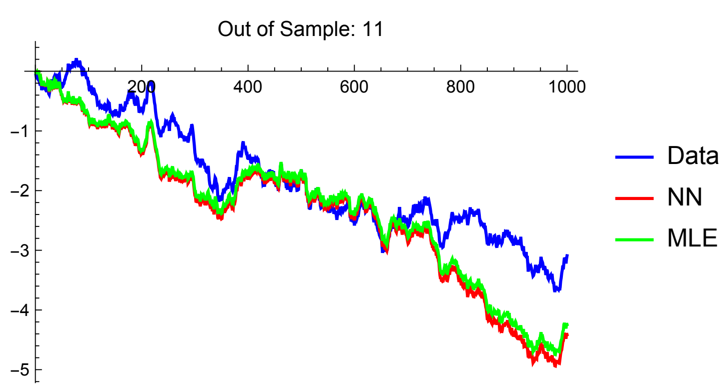
,


Out of Sample: 9
0.5
−0.5
−1.0
−1.5
−2.0
−2.5
200
400
600
800
1000
Data
NN
MLE

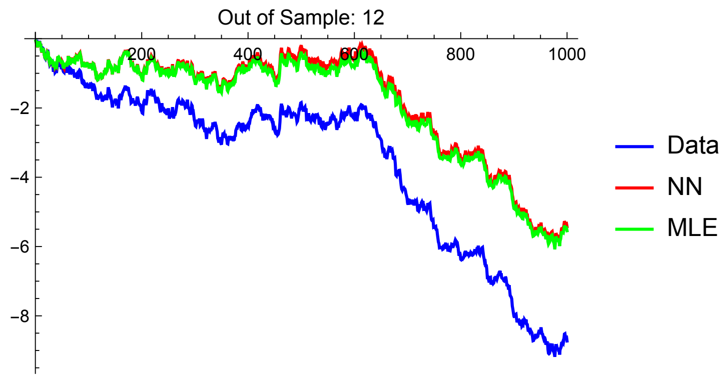
,


Out of Sample: 10
10
8
6
4
2
200
400
600
800
1000
Data
NN
MLE

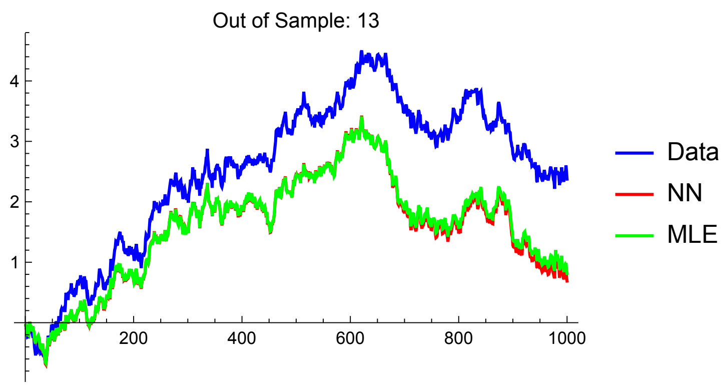
,


Out of Sample: 11
−1
−2
−3
−4
−5
200
400
600
800
1000
Data
NN
MLE

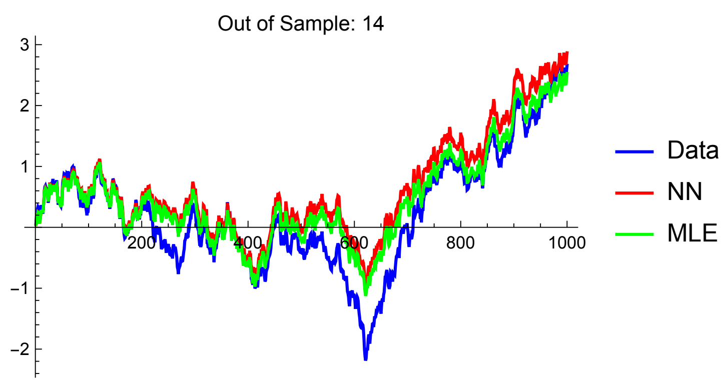
,


Out of Sample: 12
200
400
600
800
1000
-2
-4
-6
-8
Data
NN
MLE

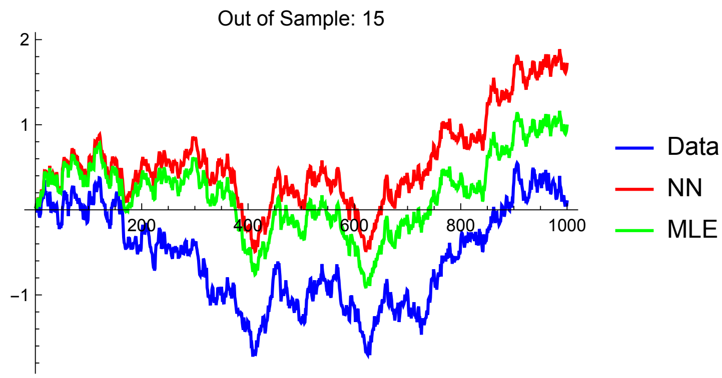
,

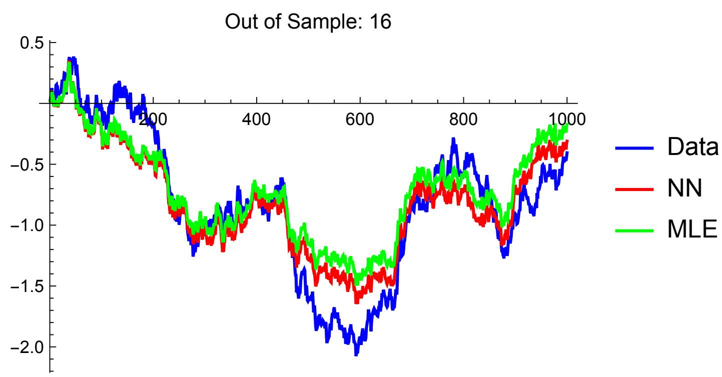

Out of Sample: 13
4
3
2
1
200
400
600
800
1000
Data
NN
MLE

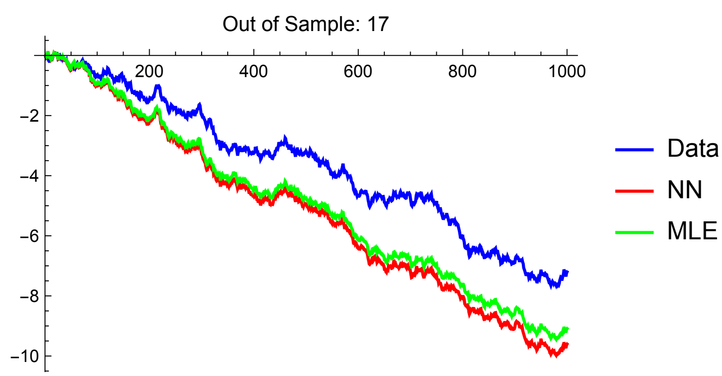
,

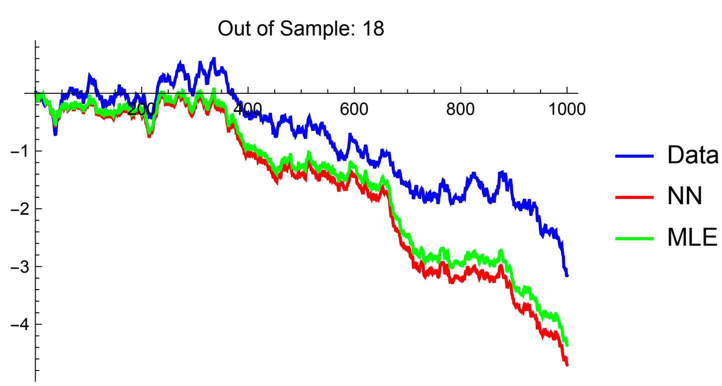

Out of Sample: 14
3
2
1
-1
-2
200
400
600
800
1000
Data
NN
MLE

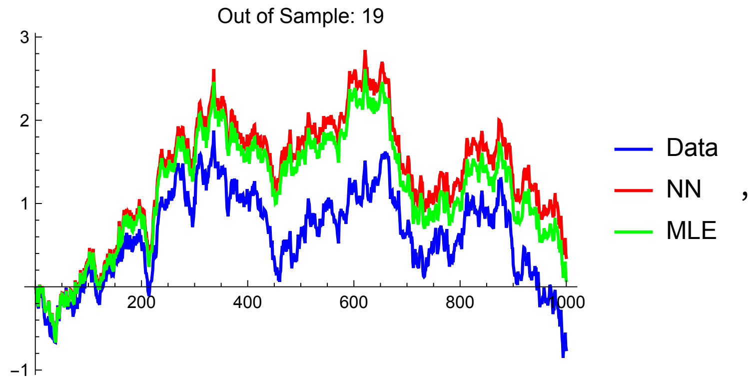
,

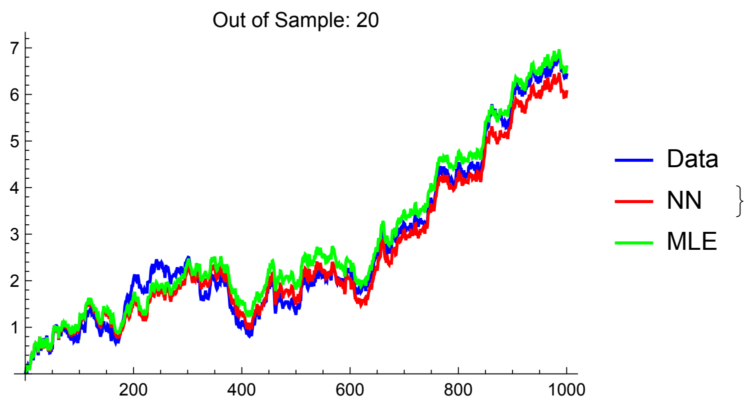

Out of Sample: 15
2
1
−1
200
400
600
800
1000
Data
NN
MLE
,


Out of Sample: 16
0.5
−0.5
−1.0
−1.5
−2.0
200
400
600
800
1000
Data
NN
MLE
,


Out of Sample: 17
200
400
600
800
1000
−2
−4
−6
−8
−10
Data
NN
MLE
,


Out of Sample: 18
200
400
600
800
1000
−1
−2
−3
−4
Data
NN
MLE
,





Covariances and Correlations

In[∘]:=
```
mnOOSCorrelations = Table[
   {Correlation[{mnTest⟦All, i⟧, mnOOSFitMLE⟦All, i⟧}ᵀ]⟦2, 1⟧,
    Correlation[{mnTest⟦All, i⟧, mnOutOfSample⟦All, i⟧}ᵀ]⟦2, 1⟧},
   {i, 1, 20}
  ];
```

In[∘]:=
```
mnOOSCovariances = Table[
   {Covariance[{mnTest⟦All, i⟧, mnOOSFitMLE⟦All, i⟧}ᵀ]⟦2, 1⟧,
    Covariance[{mnTest⟦All, i⟧, mnOutOfSample⟦All, i⟧}ᵀ]⟦2, 1⟧},
   {i, 1, 20}
  ];
```

In[ ]:=
```
Grid[
 {{Style[Column[{"Out of Sample Correlations", "Fit vs Test"}, Center],
    FontSize → 16]},
  {TableForm[mnOOSCorrelations, TableHeadings → {Automatic, {"MLE", "NN"}}]}},
 Frame → All
]
```

Out[ ]=

Out of Sample Correlations
Fit vs Test

	MLE	NN
1	0.871293	0.870852
2	0.778526	0.776619
3	0.853269	0.850838
4	0.946546	0.946115
5	0.763916	0.764487
6	0.968524	0.968515
7	0.905987	0.906933
8	0.893319	0.890657
9	0.924672	0.925998
10	0.927947	0.922149
11	0.868666	0.870177
12	0.956162	0.959279
13	0.923075	0.922958
14	0.946151	0.947495
15	0.893672	0.895092
16	0.818031	0.82183
17	0.91319	0.915448
18	0.848223	0.846556
19	0.940332	0.939887
20	0.94667	0.947178

In[•]:=
```
Grid[
 {{Style[
    Column[{"Out of Sample Covariances", "Fit vs Test"}, Center], FontSize → 16]},
  {TableForm[mnOOSCovariances, TableHeadings → {Automatic, {"MLE", "NN"}}]}},
 Frame → All
]
```

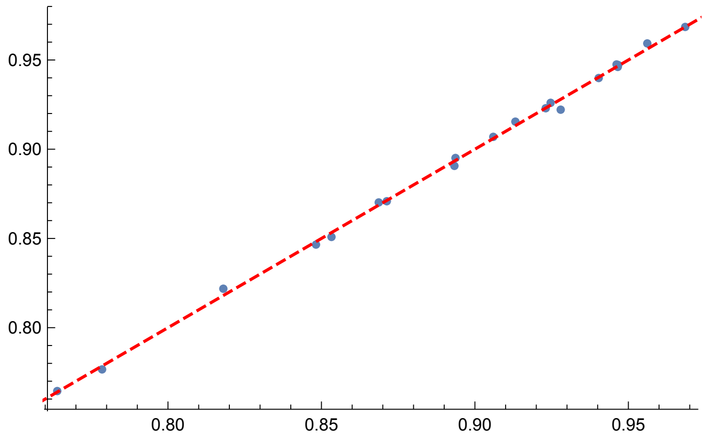
Out[•]=

Out of Sample Covariances
Fit vs Test

	MLE	NN
1	0.00253112	0.0025853
2	0.0012271	0.00130834
3	0.00203706	0.00215099
4	0.00564936	0.00585918
5	0.00122607	0.00126964
6	0.00964287	0.00991296
7	0.00346187	0.00357602
8	0.00294983	0.00302496
9	0.00451996	0.00465466
10	0.0038063	0.0039851
11	0.00240238	0.00251651
12	0.00661996	0.0069195
13	0.00458877	0.00471268
14	0.00634315	0.0065242
15	0.00326805	0.00338973
16	0.00147581	0.00156455
17	0.00341985	0.00361981
18	0.00206794	0.00213564
19	0.00552823	0.00567502
20	0.00587272	0.00608637

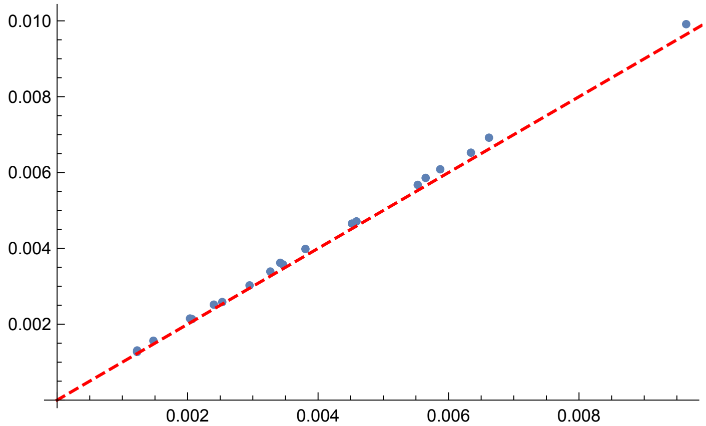
In[•]:=
```
Show[
 ListPlot[mnOOSCorrelations],
 Plot[x, {x, 0, 1}, PlotStyle → {Dashed, Red}]
]
```

Out[•]=



In[∘]:=
```
Show[
 ListPlot[mnOOSCovariances],
 Plot[x, {x, 0, 1}, PlotStyle → {Dashed, Red}]
]
```

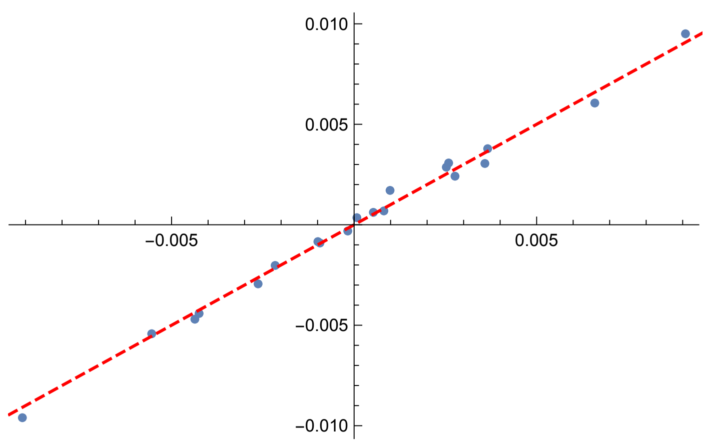
Out[∘]=



Comparison Mean and Standard Deviations

In[∘]:=
```
Show[
 ListPlot[{Mean@mnOOSFitMLE, Mean@mnOutOfSample}\[Transpose]],
 Plot[x, {x, -0.12, 0.12}, PlotStyle → {Dashed, Red}]
]
```

Out[∘]=



In[∘]:=
```
LinearModelFit[{Mean@mnTest, Mean@mnOOSFitMLE}\[Transpose], x, x]
```

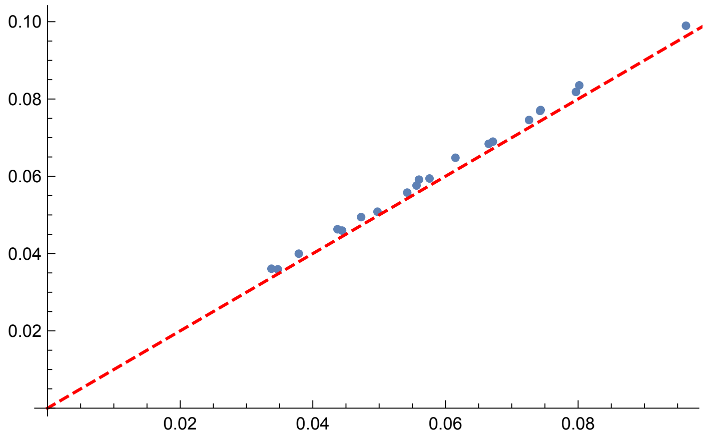
Out[∘]= FittedModel[0.000312706 + 0.957631 x]

In[∘]:=
```
LinearModelFit[{Mean@mnTest, Mean@mnOutOfSample}\[Transpose], x, x]
```

Out[∘]= FittedModel[0.000310573 + 0.97555 x]

In[•]:=
```
Show[
 ListPlot[{StandardDeviation@mnOOSFitMLE, StandardDeviation@mnOutOfSample}ᵀ,
  AxesOrigin → {0, 0}],
 Plot[x, {x, 0, 0.1}, PlotStyle → {Dashed, Red}, PlotRange → All]
]
```

Out[•]=



This first fit compares the standard deviations of the two fits.

In[•]:=
```
LinearModelFit[
 {StandardDeviation@mnOOSFitMLE, StandardDeviation@mnOutOfSample}ᵀ, x, x]
```

Out[•]= FittedModel[$0.00120354 + 1.01688\,x$]

Note that the fit by definition does not include the effect of errors. The next fit uses the actual data to estimate the relationship between the fit and actual standard deviation. The next two analyze how close those results are to the same computation using the out-of-sample model results.

In[•]:=
```
LinearModelFit[
 {StandardDeviation[vnAlphas + # & /@ (mnFactorReturns.Transpose[mnFactorLoadings])],
   StandardDeviation[mnAssetReturns]}ᵀ, x, x]
```

Out[•]= FittedModel[$0.0141635 + 0.888746\,x$]

In[•]:=
```
LinearModelFit[
 {StandardDeviation[vnAlphas + # & /@ (mnFactorReturns.Transpose[mnFactorLoadings])],
   StandardDeviation[mnAssetReturns]}ᵀ, x, x, IncludeConstantBasis → False]
```

Out[•]= FittedModel[$1.10844\,x$]

In[•]:=
```
LinearModelFit[{StandardDeviation@mnOOSFitMLE, StandardDeviation@mnTest}ᵀ, x, x]
```

Out[•]= FittedModel[$0.01471 + 0.897201\,x$]

In[ ]:=
```
LinearModelFit[{StandardDeviation@mnOOSFitMLE, StandardDeviation@mnTest}ᵀ,
  x, x, IncludeConstantBasis → False]
```

Out[ ]=
```
FittedModel[ 1.12723 x ]
```

In[ ]:=
```
LinearModelFit[{StandardDeviation@mnOutOfSample, StandardDeviation@mnTest}ᵀ,
  x, x, IncludeConstantBasis → False]
```

Out[ ]=
```
FittedModel[ 1.0886 x ]
```

Conclusion

The bottom line is that both methods appear to be effective in modeling the returns. The advantages of the MLE approach are that its underlying theory and form are both clearly laid out and understood, including the fact that the factors it generates are mutually uncorrelated. The advantage of the neural network approach is that it is straightforward to incorporate many non-linear effects which may prove useful in improving the fit.