Previsione del tasso di cambio Marco/$

In questo esempio, i dati rappresentano i tassi giornalieri di cambio del marco tedesco rispetto al dollaro USA dall'inizio del 1987 alla fine del 1997.

Operazioni preliminari

- **Carichiamo il pacchetto *Neural Networks* e i dati.**

```
SetDirectory[NotebookDirectory[]];
Needs["NeuralNetworks`"];
<< currency.dat;
```

Viene caricata la serie temporale *ydeu* che contiene i dati sul tasso di cambio del marco tedesco.

- **Controlliamo le dimensioni della serie temporale.**

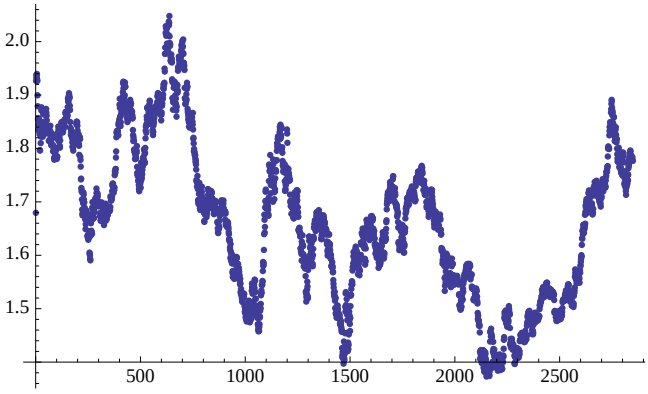
```
Dimensions[ydeu]
```

```
{2849, 4}
```

Sono disponibili i tassi di cambio per circa 2850 giorni. Vi sono quattro valori per ciascun giorno, rappresentanti il tasso di apertura, il tasso massimo, quello minimo e quello di chiusura.

- **Rappresentiamo graficamente il valore massimo del marco rispetto al (numero del) giorno.**

```
ListPlot[Flatten[ydeu[[All, {2}]]]
```



Supponiamo di voler predire il valore di apertura utilizzando i tassi di cambio del giorno precedente. Quindi il valore di apertura sarà definito come l'output del processo, che sarà memorizzato in *y*, e gli altri tre tassi sono definiti come input e saranno memorizzati in *u*.

- **Dividiamo le colonne in input ed output.**

```
u = ydeu[[All, 2 ;; 4]];
y = ydeu[[All, {1}]];
```

Il modello predittivo può essere descritto dalla seguente equazione:

$$\hat{y}(t) = g(\theta, x(t)) \qquad (1)$$

dove $\hat{y}(t)$ è la previsione dell'output $y(t)$, la funzione g rappresenta il modello di rete neurale, θ i parametri del modello ed $x(t)$ è il regressore del modello. Il regressore è dato dalla seguente equazione:

$$x(t) = [y(t-1),\ u_1(t-1),\ u_2(t-1),\ u_3(t-1)]^T \qquad (2)$$

Per un adeguato test del predittore, il set di dati è diviso in dati di training (addestramento) e dati di validazione. Il secondo set di dati verrà utilizzato per validare e comparare i diversi predittori. I comandi seguenti scrivono i dati di training nelle matrici *ue* ed *ye* ed i dati di validazione in *uv* ed *yv*.

▪ Dividiamo i dati in training e validazione.

```
ue = u[[Range[1000]]];
ye = y[[Range[1000]]];
uv = u[[Range[1001, 2000]]];
yv = y[[Range[1001, 2000]]];
```

Il modello neurale appropriato per questo problema è di tipo `NeuralARX`, inizializzato e stimato con il comando `NeuralARXFit`. Verrà prima stimato un modello di previsione lineare, che scegliamo di tipo `FeedForwardNet` senza neuroni nascosti. Per ottenere il regressore nella forma dell'eq. (2), gli indici del regressore devono essere scelti come segue: n_a={1}, n_b={1,1,1}, and n_k={1,1,1}.

Predisposizione della rete neurale

▪ Stimiamo un modello lineare per la previsione del tasso di cambio.

```
{model1, fitrecord} = NeuralARXFit[ue, ye,
   {{1}, {1, 1, 1}, {1, 1, 1}}, FeedForwardNet, {}, CriterionPlot → False];
```

```
NeuralFit::LS :
  The model is linear in the parameters. Only one training iteration is necessary to reach the
      minimum. If no training iteration was performed, it is because
      the fit was completed during the initialization of the network.
```

Un modello lineare è una combinazione lineare dei componenti del regressore e di un parametro di bias; cioè, un modello lineare nell'eq. (1) può essere espresso come

$$\hat{y}(t) = a_1 y(t-1) + b_1 u_1(t-1) + b_2 u_2(t-1) + b_3 u_3(t-1) + b_4 \qquad (3)$$

Se si osservano i parametri del modello "addestrato", si vede che b_3 è vicino ad 1 ed i restanti parametri sono vicini a 0. Questo significa che il tasso di apertura è correlato più da vicino al tasso di chiusura del giorno precedente. Questo sembra abbastanza naturale e consente perciò di avere qualche fiducia nel modello.

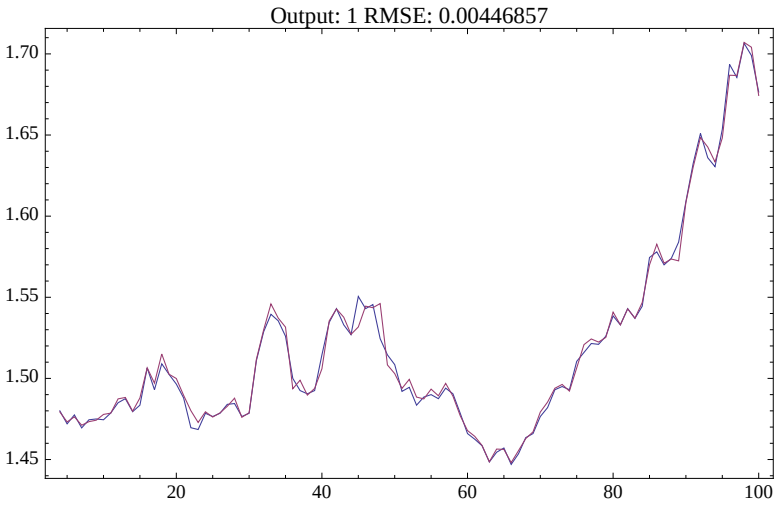
```
model1[[1]][{yy[t - 1], u1[t - 1], u2[t - 1], u3[t - 1]}]
```

```
{0.00367334 - 0.0142261 u1[-1 + t] -
  0.00706428 u2[-1 + t] + 1.02698 u3[-1 + t] - 0.00762763 yy[-1 + t]}
```

Prima di addestrare un modello non lineare, valutiamo la previsione singola (orizzonte di previsione=1) sui dati di validazione. Dato che il mercato cambia nel tempo, la previsione è valutata solo sui 100 giorni successivi i dati di stima.

Valutiamo la previsione singola sui dati di validazione.

```
NetComparePlot[uv, yv, model1, PredictHorizon → 1, ShowRange → {4, 100}]
```

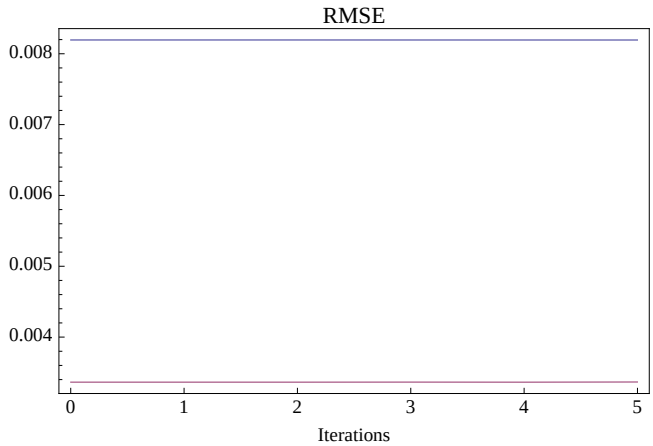


Il grafico mostra il tasso previsto insieme a quello reale. Le due curve sono così vicine che se ne coglie difficilmente la differenza. L'errore quadratico medio (RMSE) della previsione viene anche fornito e può essere utilizzato come misura della qualità del predittore.

E' ora il momento di provare i modelli non lineari per valutare se riescono a prevedere meglio del modello lineare. Viene scelta una rete feedforward con due neuroni nascosti. Il regressore scelto è lo stesso del modello lineare. Un modello lineare viene incluso in parallelo con la rete. Poichè i dati di validazione sono inclusi nell'addestramento successivo, la stima è ottenuta tramite "stopped search".

Addestriamo una rete feedforward con due neuroni sui dati del tasso di cambio.

```
{model2, fitrecord} = NeuralARXFit[ue, ye, {{1}, {1, 1, 1}, {1, 1, 1}},
   FeedForwardNet, {2}, uv, yv, 5, LinearPart → True, Separable → False];
```

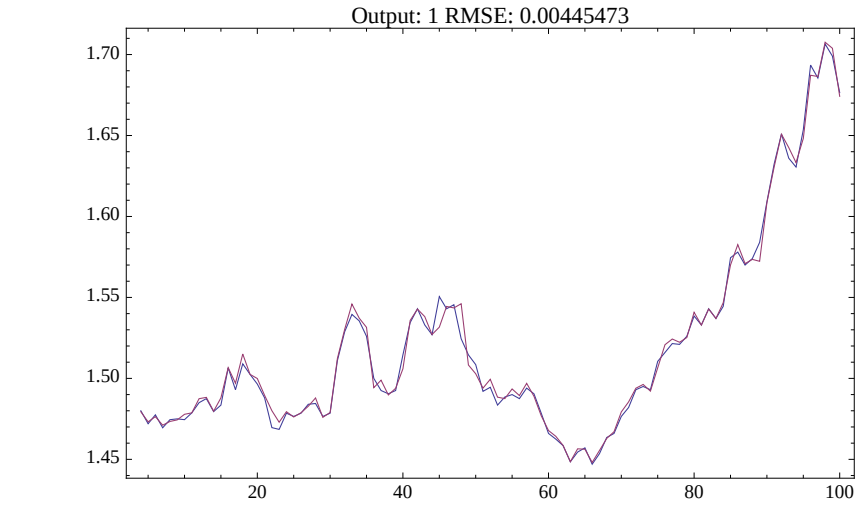


NeuralFit::StoppedSearch :
The net parameters are obtained by stopped search using the supplied validation data. The neural net is given at the 4th training iteration.

Il miglioramento della rete neurale durante l'addestramento è molto piccolo poichè l'inizializzazione della rete utilizza il metodo dei minimi quadrati per adattare i parametri lineari.

Calcoliamo la previsione ad orizzonte singolo sullo stesso intervallo di dati utilizzato per il modello lineare.

```
NetComparePlot[uv, yv, model2, PredictHorizon → 1, ShowRange → {4, 100}]
```

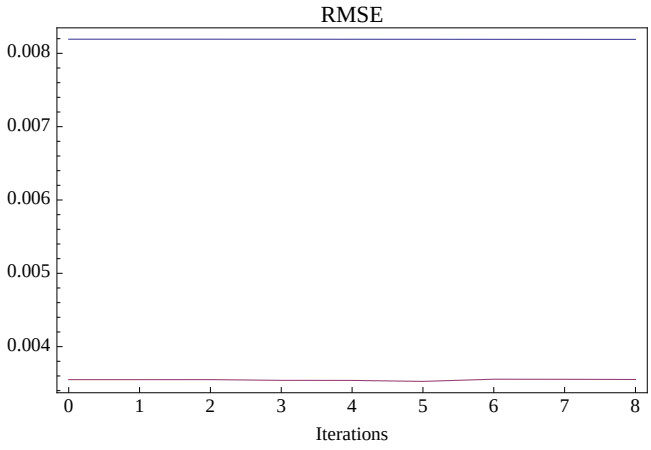


L'RMSE è leggermente inferiore per il modello non lineare rispetto al modello lineare e quindi la previsione è leggermente migliore. E' tipico che il miglioramento sia piccolo, altrimenti sarebbe troppo facile fare soldi!

Consideriamo ora una rete RBF (radial basis function) con due neuroni, mantenendo gli stessi argomenti utilizzati per la rete FF.

- **Inizializziamo ed addestriamo una rete RBF sui dati disponibili.**

```
{model3, fitrecord} =
  NeuralARXFit[ue, ye, {{1}, {1, 1, 1}, {1, 1, 1}}, RBFNet, 2, uv, yv, 8, Separable → False];
```

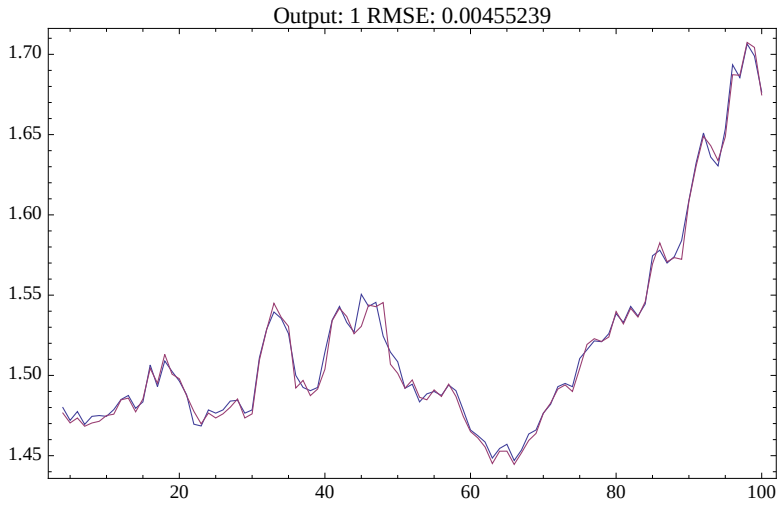


NeuralFit::StoppedSearch :
The net parameters are obtained by stopped search using the supplied validation data. The neural net is given at the 5th training iteration.

Le prestazioni della rete RBF sui dati di validazione sono peggiorate durante l'addestramento, e quindi viene restituita la rete RBF inizializzata.

Valutiamo la previsione ad orizzonte singolo con la rete RBF.

```
NetComparePlot[uv, yv, model3, PredictHorizon → 1, ShowRange → {4, 100}]
```



La rete RBF è leggermente migliore della rete lineare, ma non altrettanto buona quanto la rete FF. Se si esegue nuovamente la valutazione dell'esempio, il risultato potrebbe cambiare leggermente per la casualità dell'inizializzazione. Possiamo ripetere l'esempio modificando diverse opzioni, come:

- il numero di neuroni;
- il regressore, per considerare più valori precedenti;
- escludere alcuni dei segnali di input dal regressore.

Possiamo anche cambiare l'intervallo di dati usato nel training e la validazione.

L'esempio illustra che è possibile predire il tasso di cambio di apertura giornaliero utilizzando i tassi del giorno precedente. La relazione è ovviamente non lineare, dato che i modelli non lineari basati sulle reti FF ed RBF si comportano leggermente meglio del modello lineare.

```
NetInformation[model3]
```

```
NeuralARX model with 3 input signals and 1 output signal. The regressor
  is defined by: na = {1}, nb = {1, 1, 1}, nk = {1, 1, 1}. The mapping from
  regressor to output is defined by a Radial Basis Function network. Created
  2012-2-5 at 16:30. The network has 4 inputs and 1 output.  It consists
  of 2 basis functions of Exp type. The network has a linear submodel.
```

```
model3[[1]]
```

```
RBFNet[{{w1, λ, w2}, χ},
 {AccumulatedIterations → 8, CreationDate → {2012, 2, 5, 16, 30, 30.881247},
  Neuron → Exp, FixedParameters → None, OutputNonlinearity → None, NumberOfInputs → 4}]
```

Test finali

- **Eseguiamo dei test di confronto con i vari modelli di rete :**

```
Clear[modelperf, iMin, iMax];
modelperf[iMin_, iMax_] :=
  Module[{p, t}, p = Table[Abs[ydeu[[i + 1, 1]] - Flatten[{model1[[1]][ydeu[[i]]],
         model2[[1]][ydeu[[i]]], model3[[1]][ydeu[[i]]]}]], {i, iMin, iMax}];
   nr[1] = 0; nr[2] = 0; nr[3] = 0;
   For[i = 1, i ≤ Length[p], i++, nr[Position[p[[i]], Min[p[[i]]][[1, 1]]] ++];
   Print["Nell'intervallo ydeu[[", iMin,
    "-", iMax, "]] abbiamo le seguenti performance dei modelli:"];
   t = iMax - iMin + 1;
   Print["Modello 1: ", N[nr[1] / t] * 100, "% - Modello 2: ",
    N[nr[2] / t] * 100, "% - Modello 3: ", N[nr[3] / t] * 100, "%"]];

iMin = 1; iMax = 1000; modelperf[iMin, iMax]
```

```
Nell'intervallo ydeu[[1-1000]] abbiamo le seguenti performance dei modelli:

Modello 1: 30.9% - Modello 2: 36.9% - Modello 3: 32.2%
```

```
iMin = 1001; iMax = 2000; modelperf[iMin, iMax]
```

```
Nell'intervallo ydeu[[1001-2000]] abbiamo le seguenti performance dei modelli:

Modello 1: 28.1% - Modello 2: 38.8% - Modello 3: 33.1%
```

```
iMin = 2001; iMax = 2848; modelperf[iMin, iMax]
```

```
Nell'intervallo ydeu[[2001-2848]] abbiamo le seguenti performance dei modelli:

Modello 1: 44.4575% - Modello 2: 23.8208% - Modello 3: 31.7217%
```

Dalla comparazione delle performance dei modelli si vede che, mentre la rete RBF (modello 3) si comporta in maniera costante (circa il 32% di successi) in tutto il dataset, il modello lineare (modello 1) migliora notevolmente nella porzione di generalizzazione del dataset (intervallo 2001-2848) a differenza della rete FeedForward (modello 2).

Questo è anche confermato dalla stima effettuata sul dataset complessivo, nel quale naturalmente si livellano le differenze evidenziate in precedenza:

```
iMin = 1; iMax = 2848; modelperf[iMin, iMax]
```

```
Nell'intervallo ydeu[[1-2848]] abbiamo le seguenti performance dei modelli:

Modello 1: 33.9537% - Modello 2: 33.6728% - Modello 3: 32.3736%
```