使用深度学习进行时序预测

此示例说明如何使用长期短期记忆 (LSTM) 网络预测时序数据。

要预测序列在将来时间步的值，您可以训练“序列到序列”回归 LSTM 网络，其中响应是将值移位了一个时间步的训练序列。也就是说，在输入序列的每个时间步，LSTM 网络都学习预测下一个时间步的值。

要预测将来多个时间步的值，请使用 `predictAndUpdateState` 函数一次预测一个时间步，并在每次预测时更新网络状态。

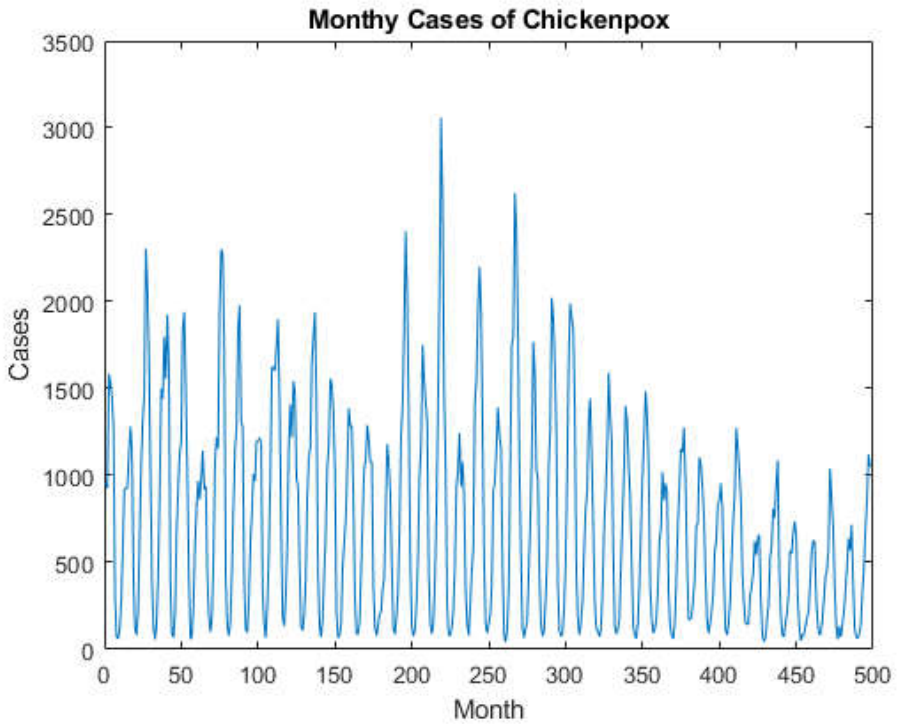
此示例使用数据集 `chickenpox_dataset`。该示例训练一个 LSTM 网络，旨在根据前几个月的水痘病例数来预测未来的水痘病例数。

加载序列数据

加载示例数据。`chickenpox_dataset` 包含一个时序，其时间步对应于月份，值对应于病例数。输出是一个元胞数组，其中每个元素均为单一时间步。将数据重构为行向量。

```
data = chickenpox_dataset;
data = [data{:}];

figure
plot(data)
xlabel("Month")
ylabel("Cases")
title("Monthy Cases of Chickenpox")
```



对训练数据和测试数据进行分区。序列的前 90% 用于训练，后 10% 用于测试。

```
numTimeStepsTrain = floor(0.9*numel(data));

dataTrain = data(1:numTimeStepsTrain+1);
dataTest = data(numTimeStepsTrain+1:end);
```

标准化数据

为了获得较好的拟合并防止训练发散，将训练数据标准化为具有零均值和单位方差。在预测时，您必须使用与训练数据相同的参数来标准化测试数据。

```
mu = mean(dataTrain);
sig = std(dataTrain);

dataTrainStandardized = (dataTrain - mu) / sig;
```

准备预测变量和响应

要预测序列在将来时间步的值，请将响应指定为将值移位了一个时间步的训练序列。也就是说，在输入序列的每个时间步，LSTM 网络都学习预测下一个时间步的值。预测变量是没有最终时间步的训练序列。

```
XTrain = dataTrainStandardized(1:end-1);
YTrain = dataTrainStandardized(2:end);
```

定义 LSTM 网络架构

创建 LSTM 回归网络。指定 LSTM 层有 200 个隐含单元。

```
numFeatures = 1;
numResponses = 1;
numHiddenUnits = 200;

layers = [ ...
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits)
    fullyConnectedLayer(numResponses)
    regressionLayer];
```

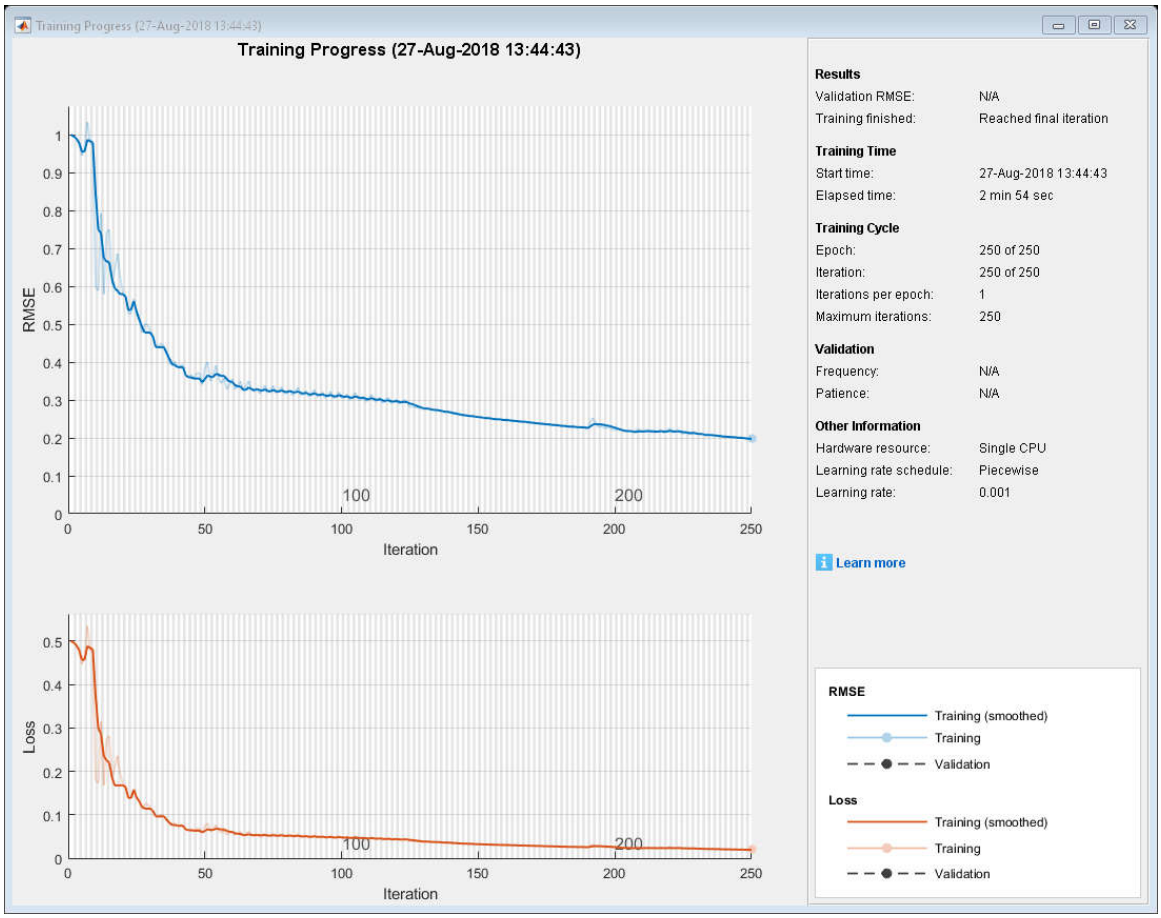
指定训练选项。将求解器设置为 'adam' 并进行 250 轮训练。要防止梯度爆炸，请将梯度阈值设置为 1。指定初始学习率 0.005，在 125 轮训练后通过乘以因子 0.2 来降低学习率。

```
options = trainingOptions('adam', ...
    'MaxEpochs',250, ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.005, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',125, ...
    'LearnRateDropFactor',0.2, ...
    'Verbose',0, ...
    'Plots','training-progress');
```

训练 LSTM 网络

使用 trainNetwork 以指定的训练选项训练 LSTM 网络。

```
net = trainNetwork(XTrain,YTrain,layers,options);
```



预测将来时间步

要预测将来多个时间步的值，请使用 `predictAndUpdateState` 函数一次预测一个时间步，并在每次预测时更新网络状态。对于每次预测，使用前一次预测作为函数的输入。

使用与训练数据相同的参数来标准化测试数据。

```
dataTestStandardized = (dataTest - mu) / sig;
XTest = dataTestStandardized(1:end-1);
```

要初始化网络状态，请先对训练数据 `XTrain` 进行预测。接下来，使用训练响应的最后一个时间步 `YTrain(end)` 进行第一次预测。循环其余预测并将前一次预测输入到 `predictAndUpdateState`。

对于大型数据集合、长序列或大型网络，在 GPU 上进行预测计算通常比在 CPU 上快。其他情况下，在 CPU 上进行预测计算通常更快。对于单时间步预测，请使用 CPU。要使用 CPU 进行预测，请将 `predictAndUpdateState` 的 `'ExecutionEnvironment'` 选项设置为 `'cpu'`。

```
net = predictAndUpdateState(net,XTrain);
[net,YPred] = predictAndUpdateState(net,YTrain(end));

numTimeStepsTest = numel(XTest);
for i = 2:numTimeStepsTest
    [net,YPred(:,i)] = predictAndUpdateState(net,YPred(:,i-1),'ExecutionEnvironment','cpu');
end
```

使用先前计算的参数对预测去标准化。

```
YPred = sig*YPred + mu;
```

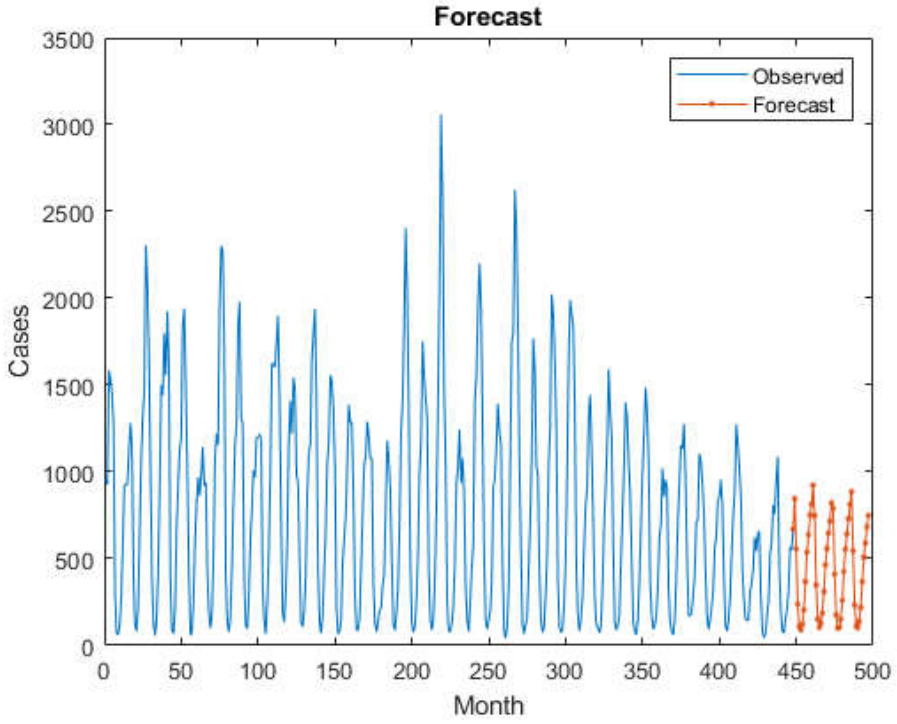
训练进度图会报告根据标准化数据计算出的均方根误差 (RMSE)。根据去标准化的预测值计算 RMSE。

```
YTest = dataTest(2:end);
rmse = sqrt(mean((YPred-YTest).^2))
```

```
rmse = single
    209.5295
```

使用预测值绘制训练时序。

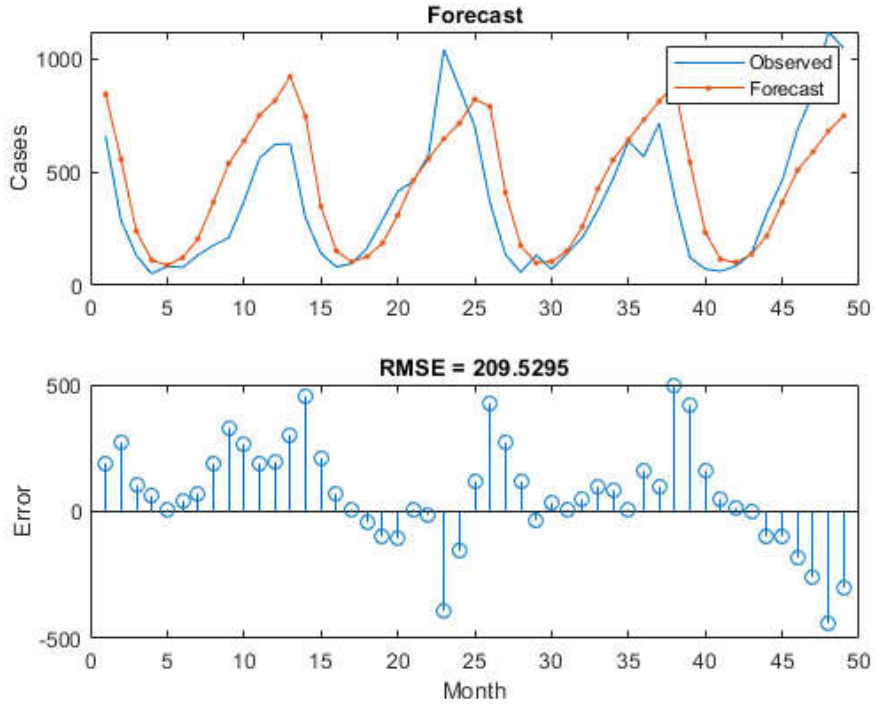
```
figure
plot(dataTrain(1:end-1))
hold on
idx = numTimeStepsTrain:(numTimeStepsTrain+numTimeStepsTest);
plot(idx,[data(numTimeStepsTrain) YPred],'.-')
hold off
xlabel("Month")
ylabel("Cases")
title("Forecast")
legend(["Observed" "Forecast"])
```



将预测值与测试数据进行比较。

```
figure
subplot(2,1,1)
plot(YTest)
hold on
plot(YPred,'.-')
hold off
legend(["Observed" "Forecast"])
ylabel("Cases")
title("Forecast")

subplot(2,1,2)
stem(YPred - YTest)
xlabel("Month")
ylabel("Error")
title("RMSE = " + rmse)
```



使用观测值更新网络状态

如果您可以访问预测之间的时间步的实际值，则可以使用观测值而不是预测值更新网络状态。

首先，初始化网络状态。要对新序列进行预测，请使用 `resetState` 重置网络状态。重置网络状态可防止先前的预测影响对新数据的预测。重置网络状态，然后通过对训练数据进行预测来初始化网络状态。

```
net = resetState(net);
net = predictAndUpdateState(net,XTrain);
```

对每个时间步进行预测。对于每次预测，使用前一时间步的观测值预测下一个时间步。将 `predictAndUpdateState` 的 `'ExecutionEnvironment'` 选项设置为 `'cpu'`。

```
YPred = [];
numTimeStepsTest = numel(XTest);
for i = 1:numTimeStepsTest
    [net,YPred(:,i)] = predictAndUpdateState(net,XTest(:,i),'ExecutionEnvironment','cpu');
end
```

使用先前计算的参数对预测去标准化。

```
YPred = sig*YPred + mu;
```

计算均方根误差 (RMSE)。

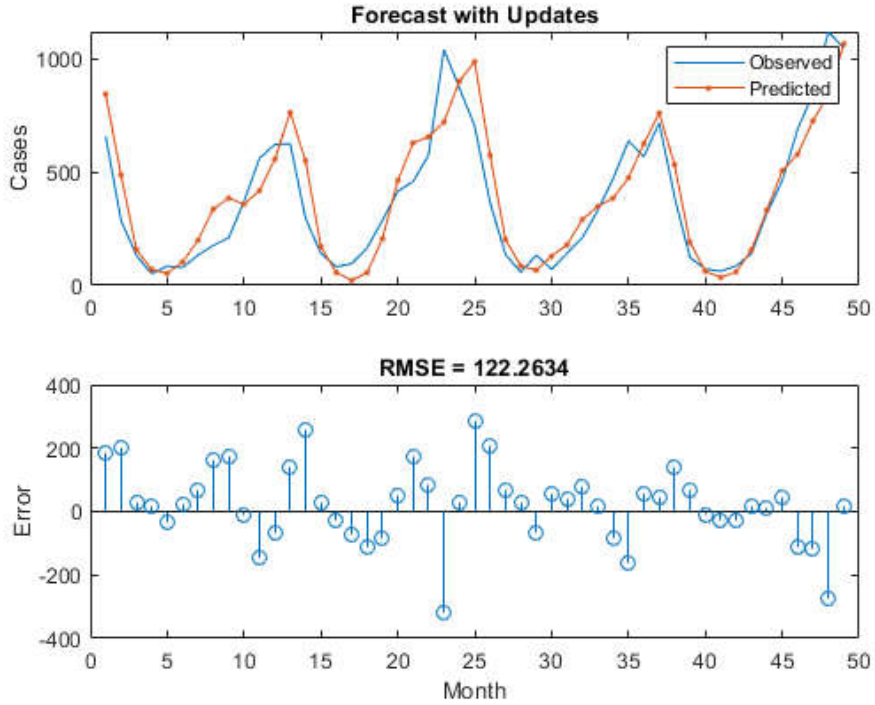
```
rmse = sqrt(mean((YPred-YTest).^2))
```

```
rmse = 122.2634
```

将预测值与测试数据进行比较。

```
figure
subplot(2,1,1)
plot(YTest)
hold on
plot(YPred,'.-')
hold off
legend(["Observed" "Predicted"])
ylabel("Cases")
title("Forecast with Updates")

subplot(2,1,2)
stem(YPred - YTest)
xlabel("Month")
ylabel("Error")
title("RMSE = " + rmse)
```



这里，当使用观测值而不是预测值更新网络状态时，预测更准确。

另请参阅

`lstmLayer` | `sequenceInputLayer` | `trainNetwork` | `trainingOptions`

相关主题

- 使用深度学习生成文本
- 使用深度学习进行序列分类
- 使用深度学习进行“序列到序列”分类
- 使用深度学习进行“序列到序列”回归
- Long Short-Term Memory Networks
- Deep Learning in MATLAB

How useful was this information?

☆ ☆ ☆ ☆ ☆