Multivariate Time Series Forecasting with LSTMs in Keras

by **Jason Brownlee** on August 14, 2017 in **Long Short-Term Memory Networks**

Neural networks like Long Short-Term Memory (LSTM) recurrent neural networks are able to almost seamlessly model problems with multiple input variables.

This is a great benefit in time series forecasting, where classical linear methods can be difficult to adapt to multivariate or multiple input forecasting problems.

In this tutorial, you will discover how you can develop an LSTM model for multivariate time series forecasting in the Keras deep learning library.

After completing this tutorial, you will know:

[redacted]

- How to transform a raw dataset into something we can use for time series forecasting.
- How to prepare data and fit an LSTM for a multivariate time series forecasting problem.
- How to make a forecast and rescale the result back into the original units.

Let’s get started.

- **Updated Aug/2017**: Fixed a bug where yhat was compared to obs at the previous time step when calculating the final RMSE. Thanks, Songbin Xu and David Righart.

Tutorial Overview

This tutorial is divided into 3 parts; they are:

1. Air Pollution Forecasting
2. Basic Data Preparation
3. Multivariate LSTM Forecast Model

Python Environment

This tutorial assumes you have a Python SciPy environment installed. You can use either Python 2 c

You must have Keras (2.0 or higher) installed with either the TensorFlow or Theano backend.

The tutorial also assumes you have scikit-learn, Pandas, NumPy and Matplotlib installed.

If you need help with your environment, see this post:

- How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda

1. Air Pollution Forecasting

In this tutorial, we are going to use the Air Quality dataset.

This is a dataset that reports on the weather and the level of pollution each hour for five years at the US embassy in Beijing, China.

The data includes the date-time, the pollution called PM2.5 concentration, and the weather informati nd direction, wind speed and the cumulative number of hours of snow and rain. The complete feature lis

1. **No**: row number
2. **year**: year of data in this row
3. **month**: month of data in this row
4. **day**: day of data in this row
5. **hour**: hour of data in this row
6. **pm2.5**: PM2.5 concentration
7. **DEWP**: Dew Point
8. **TEMP**: Temperature
9. **PRES**: Pressure
10. **cbwd**: Combined wind direction
11. **Iws**: Cumulated wind speed
12. **Is**: Cumulated hours of snow
13. **Ir**: Cumulated hours of rain

We can use this data and frame a forecasting problem where, given the weather conditions and pollution for prior hours, we forecast the pollution at the next hour.

[redacted]

This dataset can be used to frame other forecasting problems.
Do you have good ideas? Let me know in the comments below.

You can download the dataset from the UCI Machine Learning Repository.

- Beijing PM2.5 Data Set

Download the dataset and place it in your current working directory with the filename “*raw.csv*“.

2. Basic Data Preparation

The data is not ready to use. We must prepare it first.

Below are the first few rows of the raw dataset.

```
No,year,month,day,hour,pm2.5,DEWP,TEMP,PRES,cbwd,Iws,Is,Ir
1,2010,1,1,0,NA,-21,-11,1021,NW,1.79,0,0
2,2010,1,1,1,NA,-21,-12,1020,NW,4.92,0,0
3,2010,1,1,2,NA,-21,-11,1019,NW,6.71,0,0
4,2010,1,1,3,NA,-21,-14,1019,NW,9.84,0,0
5,2010,1,1,4,NA,-20,-12,1018,NW,12.97,0,0
```

The first step is to consolidate the date-time information into a single date-time so that we can use it

A quick check reveals NA values for pm2.5 for the first 24 hours. We will, therefore, need to remove d “NA” values later in the dataset; we can mark them with 0 values for now.

The script below loads the raw dataset and parses the date-time information as the Pandas DataFra clearer names are specified for each column. Finally, the NA values are replaced with “0” values and the first 24 hours are removed.

The “No” column is dropped and then clearer names are specified for each column. Finally, the NA values are replaced with “0” values and the first 24 hours are removed.

```
from pandas import read_csv
from datetime import datetime
# load data
def parse(x):
    return datetime.strptime(x, '%Y %m %d %H')
```

```
dataset = read_csv('raw.csv',  parse_dates = [['year', 'month', 'day', 'hour']], index_[redacted]
dataset.drop('No', axis=1, inplace=True)
# manually specify column names
dataset.columns = ['pollution', 'dew', 'temp', 'press', 'wnd_dir', 'wnd_spd', 'snow', 'rain']
dataset.index.name = 'date'
# mark all NA values with 0
dataset['pollution'].fillna(0, inplace=True)
# drop the first 24 hours
dataset = dataset[24:]
# summarize first 5 rows
print(dataset.head(5))
# save to file
dataset.to_csv('pollution.csv')
```

Running the example prints the first 5 rows of the transformed dataset and saves the dataset to “*pollution.csv*“.

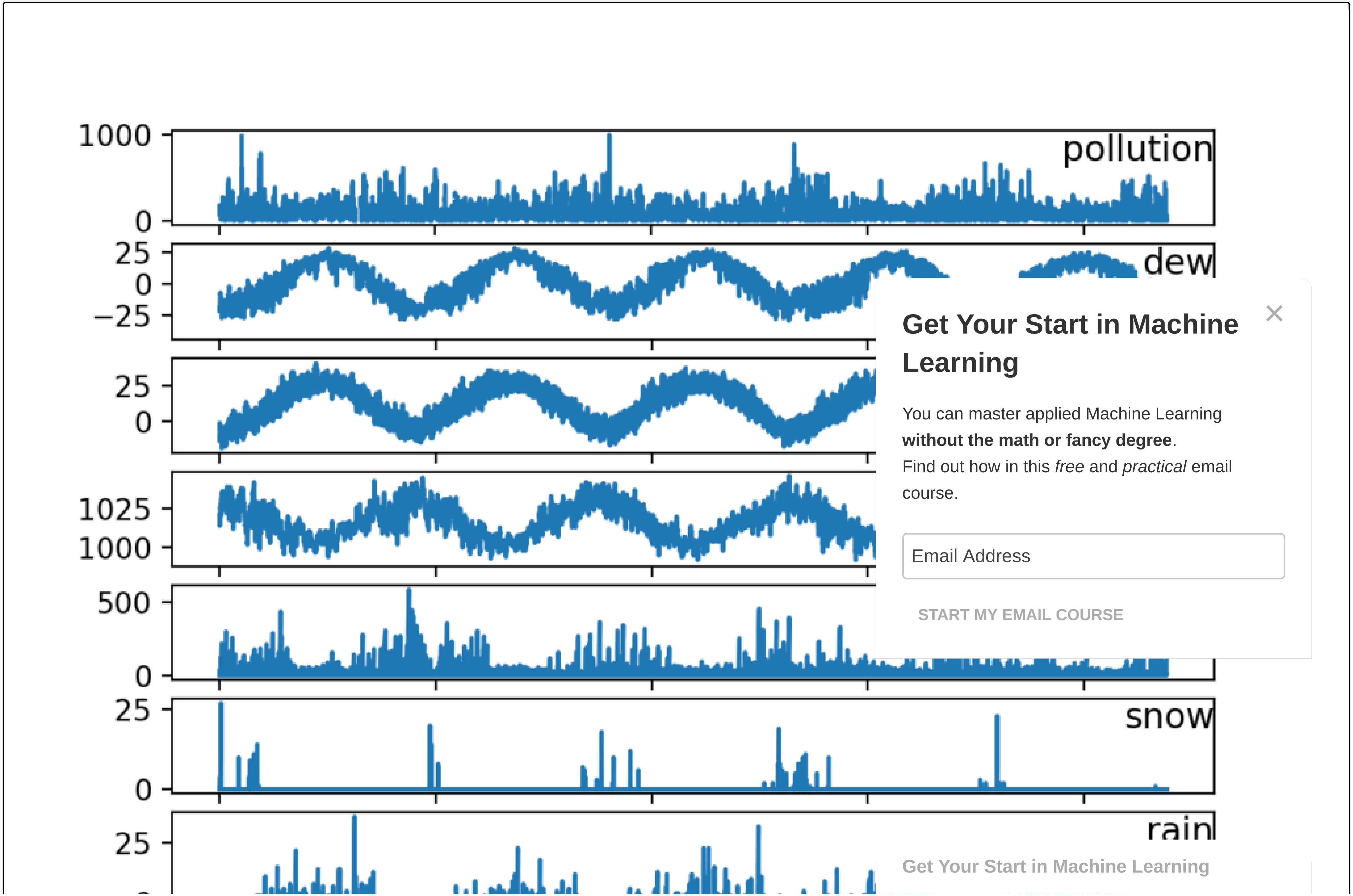
```
                     pollution  dew  temp   press wnd_dir  wnd_spd  snow  rain
date
2010-01-02 00:00:00      129.0  -16  -4.0  1020.0      SE     1.79     0     0
2010-01-02 01:00:00      148.0  -15  -4.0  1020.0      SE     2.68     0     0
2010-01-02 02:00:00      159.0  -11  -5.0  1021.0      SE     3.57     0     0
2010-01-02 03:00:00      181.0   -7  -5.0  1022.0      SE     5.36     1     0
2010-01-02 04:00:00      138.0   -7  -5.0  1022.0      SE     6.25     2     0
```

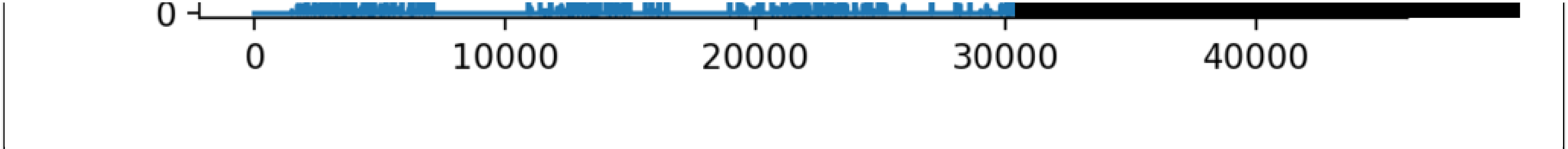
Now that we have the data in an easy-to-use form, we can create a quick plot of each series and see

The code below loads the new “*pollution.csv*” file and plots each series as a separate subplot, excep

```
from pandas import read_csv
from matplotlib import pyplot
# load dataset
dataset = read_csv('pollution.csv', header=0, index_col=0)
values = dataset.values
# specify columns to plot
groups = [0, 1, 2, 3, 5, 6, 7]
i = 1
# plot each column
pyplot.figure()
for group in groups:
    pyplot.subplot(len(groups), 1, i)
    pyplot.plot(values[:, group])
    pyplot.title(dataset.columns[group], y=0.5, loc='right')
    i += 1
pyplot.show()
```

Running the example creates a plot with 7 subplots showing the 5 years of data for each variable. [redacted]





Line Plots of Air Pollution Time Series

3. Multivariate LSTM Forecast Model

In this section, we will fit an LSTM to the problem.

LSTM Data Preparation

The first step is to prepare the pollution dataset for the LSTM.

This involves framing the dataset as a supervised learning problem and normalizing the input variabl

We will frame the supervised learning problem as predicting the pollution at the current hour (t) giver ns at the prior time step.

This formulation is straightforward and just for this demonstration. Some alternate formulations you c

- Predict the pollution for the next hour based on the weather conditions and pollution over the las
- Predict the pollution for the next hour as above and given the "expected" weather conditions for

We can transform the dataset using the *series_to_supervised()* function developed in the blog post:

- How to Convert a Time Series to a Supervised Learning Problem in Python

First, the "*pollution.csv*" dataset is loaded. The wind speed feature is label encoded (integer encoded). This could further be one-hot encoded in the future if you are interested in exploring it.

Next, all features are normalized, then the dataset is transformed into a supervised learning problem. The weather variables for the hour to be predicted (t) are then removed.

[redacted]

The complete code listing is provided below.

```
# convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
    # put it all together
    agg = concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg

# load dataset
dataset = read_csv('pollution.csv', header=0, index_col=0)
values = dataset.values
# integer encode direction
encoder = LabelEncoder()
values[:,4] = encoder.fit_transform(values[:,4])
# ensure all data is float
values = values.astype('float32')
# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)
# frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
# drop columns we don't want to predict
reframed.drop(reframed.columns[[9,10,11,12,13,14,15]], axis=1, inplace=True)
print(reframed.head())
```

Running the example prints the first 5 rows of the transformed dataset. We can see the 8 input variables (input series) and the 1 output variable (pollution level at the current hour).

[redacted]

```
    var1(t-1)  var2(t-1)  var3(t-1)  var4(t-1)  var5(t-1)  var6(t-1)  \
1    0.129779   0.352941   0.245902   0.527273   0.666667   0.002290
2    0.148893   0.367647   0.245902   0.527273   0.666667   0.003811
3    0.159960   0.426471   0.229508   0.545454   0.666667   0.005332
4    0.182093   0.485294   0.229508   0.563637   0.666667   0.008391
5    0.138833   0.485294   0.229508   0.563637   0.666667   0.009912

    var7(t-1)  var8(t-1)   var1(t)
1    0.000000        0.0  0.148893
2    0.000000        0.0  0.159960
3    0.000000        0.0  0.182093
4    0.037037        0.0  0.138833
5    0.074074        0.0  0.109658
```

This data preparation is simple and there is more we could explore. Some ideas you could look at include:

- One-hot encoding wind speed.
- Making all series stationary with differencing and seasonal adjustment.
- Providing more than 1 hour of input time steps.

This last point is perhaps the most important given the use of Backpropagation through time by LSTI

Define and Fit Model

In this section, we will fit an LSTM on the multivariate input data.

First, we must split the prepared dataset into train and test sets. To speed up the training of the mod on the first year of data, then evaluate it on the remaining 4 years of data. If you have time, consider ex

The example below splits the dataset into train and test sets, then splits the train and test sets into i e reshaped into the 3D format expected by LSTMs, namely [samples, timesteps, features].

```
# split into train and test sets
values = reframed.values
n_train_hours = 365 * 24
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]
# split into input and outputs
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]
# reshape input to be 3D [samples, timesteps, features]
```

```
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

[redacted]

Running this example prints the shape of the train and test input and output sets with about 9K hours of data for training and about 35K hours for testing.

```
(8760, 1, 8) (8760,) (35039, 1, 8) (35039,)
```

Now we can define and fit our LSTM model.

We will define the LSTM with 50 neurons in the first hidden layer and 1 neuron in the output layer for predicting pollution. The input shape will be 1 time step with 8 features.

We will use the Mean Absolute Error (MAE) loss function and the efficient Adam version of stochastic gradient descent.

The model will be fit for 50 training epochs with a batch size of 72. Remember that the internal state batch, so an internal state that is a function of a number of days may be helpful (try testing this).

Finally, we keep track of both the training and test loss during training by setting the *validation_data* both the training and test loss are plotted.

```
# design network
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
# fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=(test_X,
# plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

Evaluate Model

After the model is fit, we can forecast for the entire test dataset.

We combine the forecast with the test dataset and invert the scaling. We also invert scaling on the te

With forecasts and actual values in their original scale, we can then calculate an error score for the [redacted] Squared Error (RMSE) that gives error in the same units as the variable itself.

```
# make a prediction
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
# calculate RMSE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
```

Complete Example

The complete example is listed below.

NOTE: This example assumes you have prepared the data correctly, e.g. converted the downloaded first part of this tutorial.

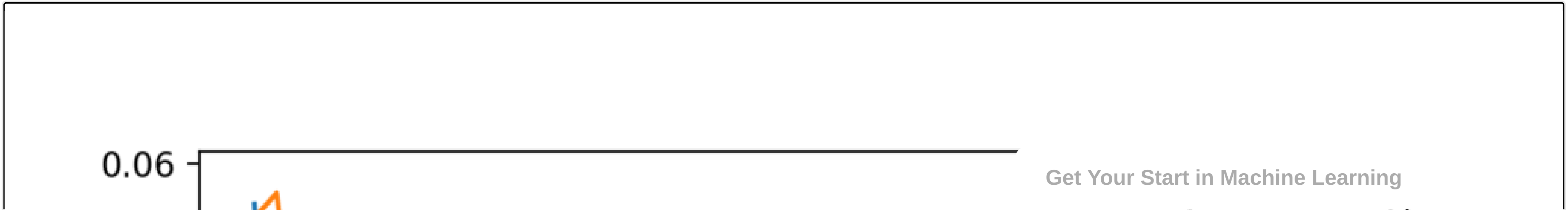
```
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

# convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
    # put it all together
    agg = concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg

# load dataset
dataset = read_csv('pollution.csv', header=0, index_col=0)
values = dataset.values
# integer encode direction
encoder = LabelEncoder()
values[:,4] = encoder.fit_transform(values[:,4])
# ensure all data is float
values = values.astype('float32')
# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)
# frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
# drop columns we don't want to predict
reframed.drop(reframed.columns[[9,10,11,12,13,14,15]], axis=1, inplace=True)
print(reframed.head())

# split into train and test sets
values = reframed.values
n_train_hours = 365 * 24
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]
# split into input and outputs
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]
# reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
```

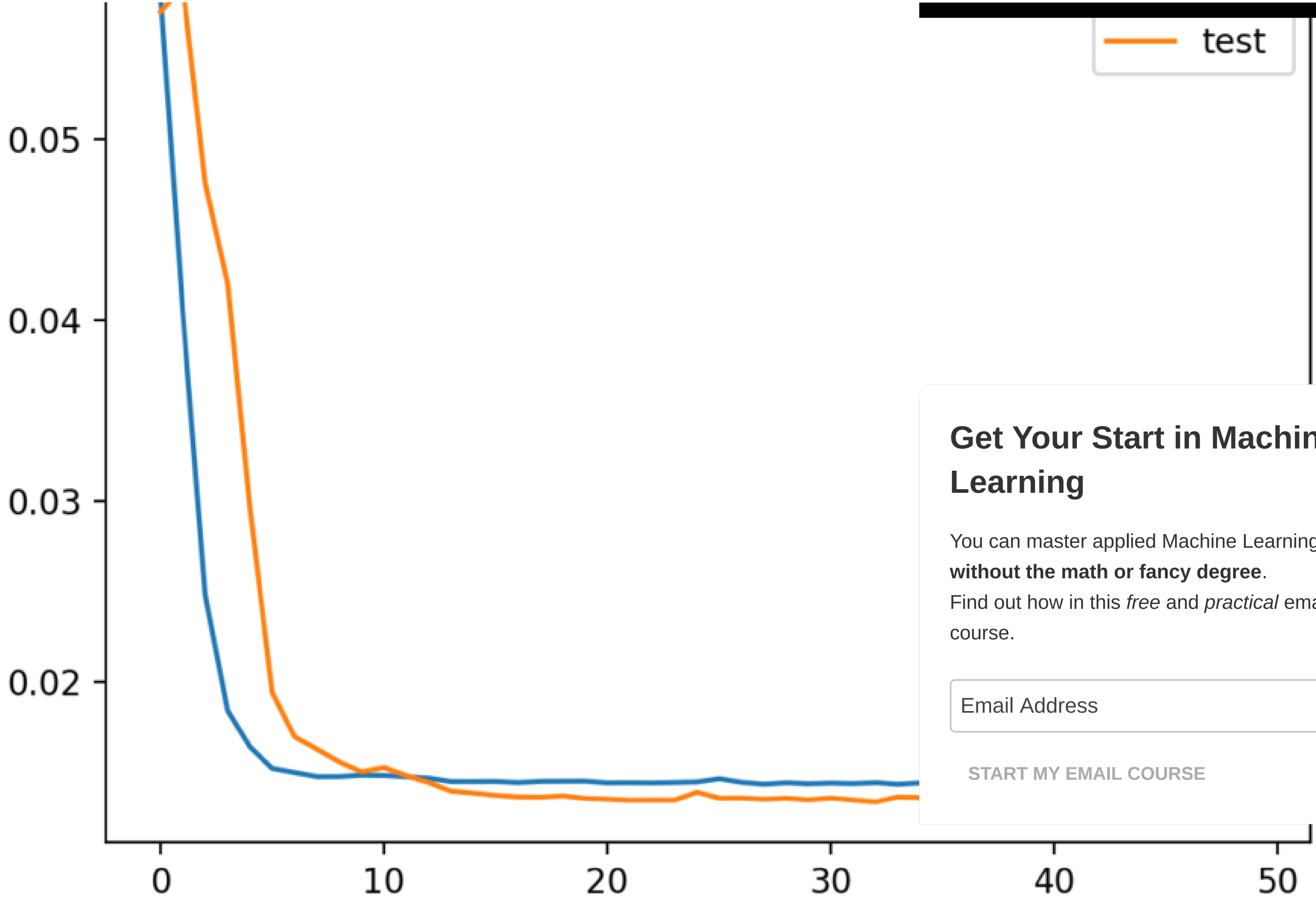
[redacted]

```
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)

# design network
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
# fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=(test_X, test_y), verbose=2, shuffle=False)
# plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()

# make a prediction
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
# calculate RMSE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
```

Running the example first creates a plot showing the train and test loss during training.

Interestingly, we can see that test loss drops below training loss. The model may be overfitting the tr training may shed more light on this.





Line Plot of Train and Test Loss from the Multivariate LSTM During Training

The Train and test loss are printed at the end of each training epoch. At the end of the run, the final l

We can see that the model achieves a respectable RMSE of 26.496, which is lower than an RMSE [redacted]

```
...
Epoch 46/50
0s - loss: 0.0143 - val_loss: 0.0133
Epoch 47/50
0s - loss: 0.0143 - val_loss: 0.0133
Epoch 48/50
0s - loss: 0.0144 - val_loss: 0.0133
Epoch 49/50
0s - loss: 0.0143 - val_loss: 0.0133
Epoch 50/50
0s - loss: 0.0144 - val_loss: 0.0133
Test RMSE: 26.496
```

This model is not tuned. Can you do better?
Let me know your problem framing, model configuration, and RMSE in the comments below.

Further Reading

This section provides more resources on the topic if you are looking go deeper.

- Beijing PM2.5 Data Set on the UCI Machine Learning Repository
- The 5 Step Life-Cycle for Long Short-Term Memory Models in Keras
- Time Series Forecasting with the Long Short-Term Memory Network in Python
- Multi-step Time Series Forecasting with Long Short-Term Memory Networks in Python

Summary

In this tutorial, you discovered how to fit an LSTM to a multivariate time series forecasting problem.

Specifically, you learned:

- How to transform a raw dataset into something we can use for time series forecasting.
- How to prepare data and fit an LSTM for a multivariate time series forecasting problem.
- How to make a forecast and rescale the result back into the original units.

Do you have any questions?

[redacted]

Ask your questions in the comments below and I will do my best to answer.

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional developer and a machine learning practitioner. He is dedicated to helping developers get started and get good at applied machine learning. Learn more.

View all posts by Jason Brownlee →

[redacted]

< Get the Most out of LSTMs on Your Sequence Prediction Problem

Mini-Course on Long Short-Term Memory Recurrent Neural Networks with Keras >

188 Responses to *Multivariate Time Series Forecasting with LSTMs in Keras*

zorg August 14, 2017 at 7:08 pm # REPLY

except wind *dir*, which is categorical.

Jason Brownlee August 15, 2017 at 6:33 am #

Thanks, fixed!

Francois AKOA August 15, 2017 at 7:16 am #

Great post Jason. Thank you so much for making this material available for the community..

Jason Brownlee August 15, 2017 at 4:54 pm # REPLY

Thanks Francois, I'm glad it helped!

yao August 15, 2017 at 2:02 pm #

hi, jason. There were some problems under my environment which were keras2.0.4and tensorf[redacted]

And Bug was that “TypeError: Expected int32, got list containing Tensors of type ‘_Message’ instead.”

The sentence that “model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))” was located.

Could you please help me with that?

Regards,

yao

Jason Brownlee August 15, 2017 at 4:54 pm #

REPLY

I would recommend this tutorial for setting up your environment:
http://machinelearningmastery.com/setup-python-environment-machine-learning-deep-learning-anac

yao August 16, 2017 at 7:18 pm #

Thx a lot, doctor, it works! fabulous! 🙂

Jason Brownlee August 17, 2017 at 6:40 am #

I’m glad to hear that.

Shirley Yang August 18, 2017 at 12:00 pm #

Dr.Jason, I update TensorFlow then it works!
Sorry to bother you.

[redacted]

Thank you very much !
Best wishes !

Jason Brownlee August 18, 2017 at 4:40 pm #

I'm glad to hear that!

Shirley Yang August 17, 2017 at 8:54 pm #

REPLY

I met the same problem .

Did you uninstall all the programs previously installed or just set up the environment again?

Thx a lot!

Shirley Yang August 18, 2017 at 11:43 am #

Hi Jason,I set up my environment as the your tutorial.

scipy: 0.19.0
numoy: 1.12.1
matplotlib: 2.0.2
pandas: 0.20.1
statsmodels: 0.8.0
sklearn: 0.18.1

theano: 0.9.0.dev-c697eeab84e5b8a74908da654b66ec9eca4f1291
tensorflow: 0.12.1
Using TensorFlow backend.
keras: 2.0.5

[redacted]

But the bug still existed.Is the version of tensorFlow too odd?How could I do?
Thanks!

Jason Brownlee August 18, 2017 at 4:39 pm #

REPLY

It might be, I am running v1.2.1.

Perhaps try running Keras off Theano instead (e.g. change the backend in the ~/.keras.jason config)

Songbin Xu August 15, 2017 at 10:42 pm #

It seems that inv_y = scaler.inverse_transform(test_X)[:,0] is not the actual, should inv_yhat be I
think this inv_y here means pollution(t-1). Is this prediction equals to only making a time shifting from the
models just take pollution(t) as the prediction of pollution(t+1))?

Jason Brownlee August 16, 2017 at 6:35 am #

Sorry, I'm not sure I follow. Can you please restate your question, perhaps with an example

Songbin Xu August 16, 2017 at 7:36 pm #

REPLY

Sorry for the confusing expression. In fact, the series_to_supervised() function would create a DataFrame whose columns are: [var1(t-1), var2(t-1), …, var1(t)] where 'var1' represents 'pollution', therefore, the first dimension in test_X (that is, test_X[:,0]) would be 'pollution(t-1)'. However, in the code you calculate the rmse between inv_yhat and test_X[:,0], even though the rmse is low, it could only shows that the model's prediction for t+1 is close to what it has known at t.
I am asking this question because I've ran through the codes and saw the models prediction pollution(t+1) looks just like pollution(t). I've also tried to

[redacted]

use t-1, t-2 and so on for training, but still changed nothing.
Do you think the model tends to learn to just take the pollution value at current moment as the prediction for the next moment?

thanks 🙂

Jason Brownlee August 17, 2017 at 6:42 am #

REPLY ↩

If we predict t for t+1 that is called persistence, and we show in the tutorial that the LSTM does a lot better than persistence.

Perhaps I don't understand your question? Can you give me an example of what you are asking?

Songbin Xu August 17, 2017 at 10:53 am #

Hmm, it's difficult to explain without a graph.

In a word, and also it's an example, I want to ask two questions:

1. In the "make a prediction" part of your codes, why it computes rmse between predicte
real t+1?

2. After the "make a prediction" part of your codes run, it turns out that rmse between pr
LSTM is making persistence?

Jason Brownlee August 17, 2017 at 4:52 pm #

RMSE is calculated for y and yhat for the same time periods (well, that was the intent), why do you think they are not?

Is there a bug?

David Righart August 18, 2017 at 5:30 am #

I think Songbin Xu is right. By executing the statement at line 90: inv_y = inv_y[redacted] polution(t-1) and inv_yhat is the predicted polution(t).

On line 50 the second parameter the function series_to_supervised can be changed to 3 or 5, so more days of history are used. If you do so, an error occurs in the scaler.inverse_transform (line 89).

No worries, great tutorial and I learned a lot so far!

Jason Brownlee August 18, 2017 at 6:54 am #

I see now, you guys are 100% correct. Thank you!

I have updated the calculation of RMSE and the final score reported in the post.

Note, I ran a ton of experiments on AWS with many different lag values > 1 and none ac
an LSTM model with no BPTT). I see this as a bad sign for the use of LSTMs for autoreg

Simone August 16, 2017 at 1:11 am #

Hi Jason, great post!

Is it necessary remove seasonality (by seasonal differentiation) when we are using LSTM?

REPLY ↩

Jason Brownlee August 16, 2017 at 6:37 am #

No, but results are often better.

REPLY ↩

Slavenya August 16, 2017 at 5:18 am #

[redacted]

Good article, thank.

Two questions:
What changes will be required if your data is sporadic? Meaning sometimes it could be 5 hours without the report.

And how do you add more timesteps into your model? Obviously you have to reshape it properly but you also have to calculate it properly.

Jason Brownlee August 16, 2017 at 6:41 am # REPLY

You could fill in the missing data by imputing or ignore the gaps using masking.

What do you mean by "add more timesteps"?

Slavenya August 16, 2017 at 7:00 pm #

But what should I do if all data is stochastic time sequence?

For example predicting time till the next event – when events frequency is stochastically distribu

Jason Brownlee August 17, 2017 at 6:39 am #

Good question, this sounds like survival analysis to me, perhaps see if it applies:
https://en.wikipedia.org/wiki/Survival_analysis

Jack Dan August 16, 2017 at 5:48 am # REPLY

Dr.Jason,

[redacted]

Thank you for an awesome post.
(I was practicing on load forecast using MLP and SVR (You also suggested on a comment in your other LSTM tutorials). I also tried with LSTM and it did almost perform like SVR. However, in LSTM, I did not consider time lags because I have predicted future predictor variables that I was feeding as test set. I will try this method with time lags to cross validate the models)

Jason Brownlee August 16, 2017 at 6:42 am # REPLY

Nice Jack, let me know how you go.

Adam August 16, 2017 at 1:03 pm #

Hi Jason,

Can I use 'look back'(Using t-2 , t-1 steps data to predict t step air pollution) in this case?
If it's available,that my input data shape will be [samples , look back , features] isn't it?

Jason Brownlee August 16, 2017 at 5:00 pm #

You can Adam, see the series_to_supervised() function and its usage in the tutorial.

Adam August 18, 2017 at 6:07 pm # REPLY

Hi Jason,

If I used n_in=5 in series_to_supervised() function,in your tutorial the input shape will be [samples, 1 , features*5].Can I reshape it to [samples, 5 , features]?If I can, what is the difference between these two shape?

REPLY ↩

Jason Brownlee August 19, 2017 at 6:09 am #

The second dimension is time steps (e.g. BPTT) and the third dimension are the features (e.g. observations at each time step). You can use features as time steps, but it would not really make sense and I expect performance to be poor.

Here's how to build a model multiple time steps for multiple features:

```
# specify number of hours
n_hours = 2
reframed = series_to_supervised(scaled, n_hours, 1)
...

# no longer just drop those columns
# reframed.drop(reframed.columns[[9,10,11,12,13,14,15]], axis=1, inplace=True)
# print(reframed.head())
...

# be more careful about choosing columns for input and output
n_features = 7
n_obs = n_hours * n_features
train_X, train_y = train[:, 0:n_obs], train[:, -n_features]
test_X, test_y = test[:, 0:n_obs], test[:, -n_features]
# reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], n_hours, n_features))
test_X = test_X.reshape((test_X.shape[0], n_hours, n_features))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
...
```

And that's it. I just tested and it looks good. The RMSE calculation will blow up, but you guys

George Khoury August 19, 2017 at 11:55 pm #

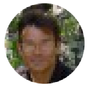
Jason, great post, very clear, and very useful!! I'm about 90% with you and think a few folks may be stuck on this final point if they try to implement multi-feature, multi-hour-lookback LSTM.

Seems like by making adjustments above, I'm able to make a prediction, but the scaling inversion doesn't want to cooperate. The reshape step now that we have multiple features and multiple timesteps has a mismatch in the shape, and even if I make the shape work, the concatenation and inversion still don't work. Could you share what else you changed in this section to make it work? I'm not so concerned about the RMSE as much as that I can extract useful predictions. Thank you for any insi

[redacted]

```
# make a prediction
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
```

…



Lg September 2, 2017 at 12:40 am #

Hi Jason,

Great and useful article.

I am somewhat puzzled by the number of features you specify to forecast the pollution r

Do not we have 8 features for each time-step and not 7?

After generating data to supervise with the function series_to_supervised(scaled,24, 1), n
is 25 * 8.

To invert the scaling for forecast I made few modifications. I used scaled.shape[1] below
don't know if the values concatenated to yhat and test_y really matter, as long as they h
the right shape.

```
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], n_obs))
```

```
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:scaled.shape[1]]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
```

```
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
```

[redacted]

inv_y = concatenate((test_y, test_X[:, 1:scaled.shape[1]]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]

The model has 4 layers with dropout.
After 200 epochs I have got
loss: 0.0169 – val_loss: 0.0162
And a rmse = 29.173

Regards.

Jason Brownlee September 2, 2017 at 6:13 am #

We have 7 features because we drop one in section “2. Basic Data Preparation

lg September 2, 2017 at 5:59 pm #

Hi Jason,

It’s really weird to me :(, as I used your code to prepare the data (pollution.csv) and I hav

[date, pollution, dew, temp, press, wnd_dir, wnd_spd, snow, rain]

😯

Jason Brownlee September 3, 2017 at 5:40 am #

Date and wind direction are dropped during data preparation, perhaps you accidentally skipped a step or are reviewing a different file from the output file?

[redacted]

Lg September 3, 2017 at 6:22 pm #

Hi Jason,

So that’s fine, in my case I have 8 features.

When reading the file, the field ‘date’ becomes the index of the dataframe and the field ‘wnd_dir’ is later label encoded, as you do above in “The complete example” lines 42-43.

It is now much clearer for me. I am not puzzled anymore. 😉

Thanks a lot for all the information contained in your articles and your e-books.

They are really very informative.

🙂

Jason Brownlee September 4, 2017 at 4:26 am #

I’m glad to hear that!

Cloud September 20, 2017 at 8:06 pm #

Hi Jason,
I think the output is column var1(t), that means:
train_X, train_y = train[:, 0:n_obs], train[:, -(n_features+1)]
am I right?
In case the “pollution” is in the last column, it is easy to get train[:, -1]
am i right?
I just want to verify that I understand your post.
Thank you, Jason

[redacted]

Arun August 18, 2017 at 12:45 am #

Hi Jason, I get the following error from line # 82 of your 'Complete Example' code.

ValueError: Error when checking : expected lstm_1_input to have 3 dimensions, but got array with shape (34895, 8)

I think LSTM() is looking for (sequences, timesteps, dimensions). In your code, line # 70, I believe 50 is timesteps while input_shape (1,8) represents the dimensions. May be it's missing 'sequences' ?

Appreciate your response.

Jason Brownlee August 18, 2017 at 6:25 am #

REPLY ↩

Ensure that you first prepare the data (e.g. convert "raw.csv" to "pollution.csv").

Neal Valiant August 18, 2017 at 2:35 am #

Hi Jason, I am wondering what the issue that I'm getting is caused by, maybe a different type o the history into the model, When i check the History.history.keys() I only get back 'loss' as my only key.

Jason Brownlee August 18, 2017 at 6:27 am #

You must specify the metrics to collect when you compile the model.

For example, in classification:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Aman Garg August 18, 2017 at 4:18 pm #

[redacted]

Hello Jason,

Thank you for such a nice tutorial.

Since you have published a similar topic and few other related topics in one of your paid books (LSTM networks), should the reader also expect some different topics covered in it?

I'm an ardent fan of your blogs since it covers most of the learning material and therefore, it makes me wonder that will be different in your book?

Jason Brownlee August 18, 2017 at 4:42 pm #

REPLY ↩

Thanks Arman.

The book does not cover time series, instead it focuses on teaching you how to implement a suite of a
for your problems.

Some ideas were tested on the blog first, most are only in the book.

You can see the full table of contents here:
http://machinelearningmastery.com/lstms-with-python/

The book provides all the content in one place, code as well, more access to me, updates as I fix bu
support my site so I can keep doing this.

Songbin Xu August 18, 2017 at 6:54 pm #

Thank you for accepting my opinions, such a pleasure!

Running the codes u modified, still something puzzles me here,

1. Have u drawn the waveforms of inv_y and inv_yhat in the same plot? I think they looks quite like persistence.

2. Curiously, I computed the rmse between pollution(t) and pollution(t-1) in test_X, it's 4.629, much lower than your final score 26.496, does it mean LSTM performs even worse than persistence?

3. I've tried to remove var1 at t-1, t-2, … , and I've also tried to use lag values>1, and also assign differe [redacted] none of them improved, they performed even worse.

Do you have any other ideas to avoid the whole model to learn persistence?

Looking forward to your advices 🙂

Jason Brownlee August 19, 2017 at 6:14 am #

REPLY ↩

Thank you for pointing out the fault!

The final line plot shows loss on the transformed train and test sets.

Yes, LSTMs are no good at autoregression, yet I keep getting asked to develop examples (tens of e
http://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecas

Consider developing a baseline with an MLP, you'll find it tough to beat it with an LSTM!

Varuna Jayasiri August 19, 2017 at 2:51 pm #

Why are you only training with a single timestep (or sequence length)? Shouldn't you use more
in https://github.com/fchollet/keras/blob/master/examples/lstm_text_generation.py they use 40 (maxlen)

Jason Brownlee August 20, 2017 at 6:05 am #

REPLY ↩

Yes, it is just an example to help you get started. I do recommend using multiple time steps in order to get the full BPTT.

Long.Ye August 23, 2017 at 11:06 am #

REPLY ↩

[redacted]

Hi Jason and Varuna,

When the timesteps = 1 as you mentioned, does it mean the value of t-1 time was used to predict the value of t time? Is moving window a method to use multiple time steps? Is there any other way? Has Keras any functions of moving window?

Thank you very much.

Jason Brownlee August 23, 2017 at 4:23 pm # REPLY

Keras treats the “time steps” of a sequence as the window, kind of. It is the closest match I can think of.

lymlin August 20, 2017 at 4:28 pm #

Hi Jason,
I met some problem when learning your codes.

dataset = read_csv(‘D:\Geany\scriptslym\raw.csv’, parse_dates = [[‘year’, ‘month’, ‘day’, ‘hour’]],index_c
Traceback (most recent call last):
File “”, line 1, in
dataset = read_csv(‘D:\Geany\scriptslym\raw.csv’, parse_dates = [[‘year’, ‘month’, ‘day’, ‘hour’]],index_c
NameError: name ‘parse’ is not defined
>>>

Jason Brownlee August 21, 2017 at 6:04 am # REPLY

It looks like you have specified a function “parse” but not defined it.

guntama August 21, 2017 at 11:30 am #

[redacted]

Hi Jason,
Can I use “keras.layers.normalization.BatchNormalization” as a substitute for “sklearn.preprocessing.MinMaxScaler”?

Jason Brownlee August 21, 2017 at 4:22 pm # REPLY

No, they do very different things.

Naveen Koneti August 21, 2017 at 10:56 pm # REPLY

Hi Jason, Its a very Informative article. Thanks. I have a question regarding forecasting in time s
columns while learning after variable transformations and the same has been done for the test data too.
during prediction. For instance, If I want to predict the pollution for a future date, Should I know the other
date which I’m not aware off? Another question is, Suppose we have same data about multiple regions(l
is not negligible), How can we model so that the input argument while prediction is the region name alon

Jason Brownlee August 22, 2017 at 6:43 am #

It depends on how you define your model.

The model defined above uses the variables from the prior time step as inputs to predict the next po

In your case, maybe you want to build a separate model per region, perhaps a model that improves performance by combining models across regions.
You must experiment to see what works best for your data.

Naveen Koneti August 24, 2017 at 4:12 pm # REPLY

Thanks! I missed the trick of converting the time-series to supervised learning problem.
guess. We just have to submit the input parameters of the previous time stamp for the specific re

encoding on the region variable too during data preprocessing.

[redacted]

LY September 7, 2017 at 8:12 pm #

REPLY

Thank you for your excellent blog, Jason. I've really learnt a lot from your nice work recently. After this post, I've already known how to transform data into data that formates LSTM and how to construct a LSTM model.

Like the question aksed by Naveen Koneti, I have the same puzzle.
Recently I've worked on some clinical data. The data is not like the one we used in this demo. It is consist of hunderds of patients, each patient has several vital sign records. If it is about one individual's records through many years, I can process the data as what you told us. I wonder how I can conquer this kind of data. Could you give me some advice, or tell me where I can find any solutions about it?
If I didn't state my question clearly and you're interested it, pls let me know.
Thanks in advance.

PS. the data set in my situation is like this
[ID date feature1 feature2 feautre3]
[patient1 date1 value11 value12 value13]
[patient1 date2 value21 value22 value23]
[patient2 date1 value31 value32 value33]
[patient2 date2……………………………………..]
[patient3 ………………………………………………]

Jason Brownlee September 9, 2017 at 11:43 am #

You could model one patient at a time, or groups or all of them. Try different approaches and see what works best.

I cannot tell you what would work best – I have no idea – you must discover it.

See this post:
http://machinelearningmastery.com/a-data-driven-approach-to-machine-learning/

Chris August 21, 2017 at 11:23 pm #

[redacted]

REPLY

Hi,
again a nice post for the use of lstm's!

I had the following idea when reading.

I would like to build a network, in which each feature has its own LSTM neuron/layer, so that the input is not fully connected.
My idea is adding a lstm layer for each feature and merge it with the merge layer and feed these results to the output neurons.

Is there a better way to do this? Or would you recommend to avoid this because the features are poorly abstracted? On the other hand, this might also be interesting.

Thank you!

Jason Brownlee August 22, 2017 at 6:44 am #

Try it and see if it can out-perform a model that learns all features together.

Also, contrast to an MLP with a window – that often does better than LSTMs on autoregression prob

Tryfon August 22, 2017 at 5:20 am #

Hi Jason,

I have two questions:

1) I have a question/ notice regarding the scaling of the Y variable (pollution). The way you implement the rescaling between [0-1] you consider the entire length of the array (all of the 43799 observations -after the dropna-).

Is it rightto rescale it that way? By doing so we are incorporating information of the furture (test set) to the past (train set) because the scaler is "exposed" to both of them and therefore we introduce bias.

If you agree with my point what could be a fix?

2) Also the activation function of the output (Y variable) is sigmoid, that's why we rescale it within the [0,

[redacted]

Thanks for sharing the article!

Jason Brownlee August 22, 2017 at 6:49 am #

REPLY ↰

No, ideally you would develop a scaling procedure on the training data and use it on test and when making predictions on new data.

I tried to keep the tutorial simple by scaling all data together.

The activation on the output layer is 'linear', the default. This must be the case because we are predicting a real-value.

WCH August 22, 2017 at 5:25 pm #

Thank you very much for your tutorial.

I have one question,

but I failed to read the NW in pollution. csv.(cbwd column)

values = values.astype('float32')
ValueError: could not convert string to float: NW

How do you fix it?

WCH August 22, 2017 at 5:30 pm #

REPLY ↰

sorry, I saw the text above and solved it.

Jason Brownlee August 23, 2017 at 6:42 am #

REPLY ↰

Glad to hear it!

[redacted]

Dmitry August 22, 2017 at 5:58 pm #

REPLY

Hi Jason!
I assume there is little mistake when you calculate RMSE on test data.
You must write this code before calculate RMSE:

```
inv_y = inv_y[:-1]
inv_yhat = inv_yhat[1:]
```

Thus, RMSE equals 10.6 (on the same data, in my case), that is much less than 26.5 in your case.

Jason Brownlee August 23, 2017 at 6:44 am #

Sorry, I don’t understand your comment and snippet of code, can you spell out the bug you

jan August 22, 2017 at 11:01 pm #

Hi Jason,

great post! I was waiting for meteo problems to infiltrate the machinelearningmastery world.

Could you write something about the changed scenareo where, given the weather conditions and pollution for some time, we can predict the pollution for another time or place with given weather conditions?

For example: We have the weather conditions and pollution given for Beijing in 2016, and we have the weather conditions given for Chengde (city close to Bejing) also in 2016. Now we want to know how was the pollution in Chengde in 2016.

Would be great to learn about that!

[redacted]

Jason Brownlee August 23, 2017 at 6:52 am # REPLY

Great suggestion, I like it. An approach would be to train the model to generalize across geographical domains based only on weather conditions.

I have tried not to use too many weather examples – I came from 6 years of work in severe weather, it's too close to home 🙂

Simone August 23, 2017 at 9:43 am # REPLY

Hi Jason,
I have read many of your posts about LSTM. I have not completely clear the difference between the para
means when the memory is reset (right?), but this shouldn't have the same value of time_steps that, if I
system makes a prediction?

Jason Brownlee August 23, 2017 at 4:22 pm #

Great question!

Batch size is the number of samples (e.g. sequences) to that are used to estimate the gradient befor
at the end of each batch after the weights are updated.

One sample is comprised of 1 or more time steps that are stepped over during backpropagation thro
features (e.g. observations recorded at that time).

Time steps and batch size and generally not related.

You can split up a sequence to have one-time step per sequence. In that case you will not get the benefit of learning across time (e.g. bptt), but you can reset state at the end of the time steps for one sequence. This an odd config though and really only good to showing off the LSTMs memory capability.

Does that help?

Simone August 24, 2017 at 6:26 am #

REPLY

Thanks, now it’s more clear!

Pedro August 23, 2017 at 8:58 pm #

REPLY

Hi,I ger this error at this step, could you help me please?

```
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))

________________________________________

TypeError Traceback (most recent call last)
in ()
—-> 1 model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))

C:\Anaconda3\lib\site-packages\keras\models.py in add(self, layer)
431 # and create the node connecting the current layer
432 # to the input layer we just created.
–> 433 layer(x)
434
435 if len(layer.inbound_nodes) != 1:

C:\Anaconda3\lib\site-packages\keras\layers\recurrent.py in __call__(self, inputs, initial_state, **kwargs)
241 # modify the input spec to include the state.
242 if initial_state is None:
–> 243 return super(Recurrent, self).__call__(inputs, **kwargs)
244
245 if not isinstance(initial_state, (list, tuple)):

C:\Anaconda3\lib\site-packages\keras\engine\topology.py in __call__(self, inputs, **kwargs)
556 ‘layer.build(batch_input_shape)‘)
557 if len(input_shapes) == 1:
–> 558 self.build(input_shapes[0])
```

```
559 else:
560 self.build(input_shapes)

C:\Anaconda3\lib\site-packages\keras\layers\recurrent.py in build(self, input_shape)
1010 initializer=bias_initializer,
1011 regularizer=self.bias_regularizer,
-> 1012 constraint=self.bias_constraint)
1013 else:
1014 self.bias = None

C:\Anaconda3\lib\site-packages\keras\legacy\interfaces.py in wrapper(*args, **kwargs)
86 warnings.warn(‘Update your ' + object_name +
87 ' call to the Keras 2 API: ‘ + signature, stacklevel=2)
—> 88 return func(*args, **kwargs)
89 wrapper._legacy_support_signature = inspect.getargspec(func)
90 return wrapper

C:\Anaconda3\lib\site-packages\keras\engine\topology.py in add_weight(self, name, shape, dtype, initial
389 if dtype is None:
390 dtype = K.floatx()
–> 391 weight = K.variable(initializer(shape), dtype=dtype, name=name)
392 if regularizer is not None:
393 self.add_loss(regularizer(weight))

C:\Anaconda3\lib\site-packages\keras\layers\recurrent.py in bias_initializer(shape, *args, **kwargs)
1002 self.bias_initializer((self.units,), *args, **kwargs),
1003 initializers.Ones()((self.units,), *args, **kwargs),
-> 1004 self.bias_initializer((self.units * 2,), *args, **kwargs),
1005 ])
1006 else:

C:\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py in concatenate(tensors, axis)
1679 return tf.sparse_concat(axis, tensors)
1680 else:
-> 1681 return tf.concat([to_dense(x) for x in tensors], axis)
```

```
1682
1683

C:\Anaconda3\lib\site-packages\tensorflow\python\ops\array_ops.py in concat(concat_dim, values, name)
998 ops.convert_to_tensor(concat_dim,
999 name=”concat_dim”,
-> 1000 dtype=dtypes.int32).get_shape(
1001 ).assert_is_compatible_with(tensor_shape.scalar())
1002 return identity(values[0], name=scope)

C:\Anaconda3\lib\site-packages\tensorflow\python\framework\ops.py in convert_to_tensor(value, dtype, name, as_ref, preferred_dtype)
667
668 if ret is None:
–> 669 ret = conversion_func(value, dtype=dtype, name=name, as_ref=as_ref)
670
671 if ret is NotImplemented:

C:\Anaconda3\lib\site-packages\tensorflow\python\framework\constant_op.py in _constant_tensor_conv
174 as_ref=False):
175 _ = as_ref
–> 176 return constant(v, dtype=dtype, name=name)
177
178

C:\Anaconda3\lib\site-packages\tensorflow\python\framework\constant_op.py in constant(value, dtype, s
163 tensor_value = attr_value_pb2.AttrValue()
164 tensor_value.tensor.CopyFrom(
–> 165 tensor_util.make_tensor_proto(value, dtype=dtype, shape=shape, verify_shape=verify_shape))
166 dtype_value = attr_value_pb2.AttrValue(type=tensor_value.tensor.dtype)
167 const_tensor = g.create_op(

C:\Anaconda3\lib\site-packages\tensorflow\python\framework\tensor_util.py in make_tensor_proto(values, dtype, shape, verify_shape)
365 nparray = np.empty(shape, dtype=np_dt)
366 else:
–> 367 _AssertCompatible(values, dtype)
```

[redacted]

```
368 nparray = np.array(values, dtype=np_dt)
369 # check to them.

C:\Anaconda3\lib\site-packages\tensorflow\python\framework\tensor_util.py in _AssertCompatible(values, dtype)
300 else:
301 raise TypeError(“Expected %s, got %s of type ‘%s’ instead.” %
–> 302 (dtype.name, repr(mismatch), type(mismatch).__name__))
303
304

TypeError: Expected int32, got list containing Tensors of type ‘_Message’ instead.
```

Jason Brownlee August 24, 2017 at 6:36 am #

Perhaps check that your environment is setup correctly:
http://machinelearningmastery.com/setup-python-environment-machine-learning-deep-learning-anac

Also, ensure that you have copied all of the code.

Neal Valiant August 24, 2017 at 2:49 am #

Hi Jason,
I was curious if you can point me in the right direction for converting data back to the actual values inste

Jason Brownlee August 24, 2017 at 6:48 am #

REPLY

Yes, you can invert the scaling.

This tutorial demonstrates how to do that Neal.

Neal Valiant August 25, 2017 at 7:34 am #

REPLY ↰

Hi Jason, I did have an issue converting back to actual values, but was able to get past it using the drop columns on the reframed data which got me past it.

When looking at my predicted values vs actual values, I'm noticing that my first column has a prediction and a true value, but for every other variable, I only see what I can assume is a prediction? does this make a prediction on every column, or just one particular one.

Im sorry for asking a question such as this, I just think I'm confusing myself looking at my results.

Jason Brownlee August 25, 2017 at 3:56 pm #

REPLY ↰

The code in the tutorial only predicts pollution.

Jack Dan August 24, 2017 at 3:24 am #

Dr. Jason,
I have been trying with my own dataset and I am getting an error "ValueError: operands could not be bro
when I try to do inv_yhat = scaler.inverse_transform(inv_yhat) as you have in line 86 in your script. I still
yhat.shape as (168,1) and test_X.shape as (168,38). When I do this, inv_yhat = np.concatenate((yhat, te
still can not figure why inverse_transform gives that error.

Jason Brownlee August 24, 2017 at 6:50 am #

REPLY ↰

The shape of the data must be the same when inverting the scale as when it was originally scaled.

This means, if you scaled with the entire test dataset (all columns), then you need to tack the yhat onto the test dataset for the inverse. We jump through these exact hoops at the end of the example when calculating RMSE.

Jay Regalia August 24, 2017 at 7:29 am # REPLY

This seems to be the same issue I am having at the moment also. i concatenate my inv_yhat with my test_X like you said, but the shape of inv_yhat after is still not taking into account the 2nd numbers(in posts case (41,).

Jack Dan August 26, 2017 at 6:00 am # REPLY

Ask a question in stackoverflow and post the link, I should be able to help. I spent lots of time on this and have a decent idea now.

Jack Dan August 24, 2017 at 7:39 am #

Yes, you're right! I did that and it worked, nice! Thank you for your comment!

Jason Brownlee August 24, 2017 at 4:24 pm #

Glad to hear that Jack.

John Regilina August 24, 2017 at 8:38 am #

I am having the same problem, but cannot solve the issue. everytime i try to concatenante them together, there is not change to my inv_yhat variable. i still am unable to understand this issue if you can expand a bit more that would be amazing

Jack Dan August 26, 2017 at 6:08 am # REPLY

[redacted]

@John Regilina,
Check the shape of data after you scale the data and then check the scale again after you do the concatenation. Remember, when your yhat shape will be (rowlength,1) and after concatenation inv_yhat should be the same shape after you scaled the data. Look at Dr.Jason's answer to my comment/question. Hope that will help. (Thanks to Dr.Jason saved a lot of my time)

Shan September 19, 2017 at 1:59 pm # REPLY

I am also stuck with same thing. How did you fix it?

Lizzie August 24, 2017 at 4:23 am #

Hi Jason, In dataset.drop('No', axis =1, inplace = True), what is the purpose of 'axis' and 'inplac

Jason Brownlee August 24, 2017 at 6:50 am #

Great question.

We specify to remove the column with axis=1 and to do it on the array in memory with inplace rather
removed.

Lizzie August 24, 2017 at 4:44 am # REPLY

Fabulous tutorials Jason!

Jason Brownlee August 24, 2017 at 6:51 am #

Thanks Lizzie.

Jaskaran August 24, 2017 at 5:19 am #

REPLY

Can you show how the multi variate forecast looks like?
Looks like you missed it in the article.

Jason Brownlee August 24, 2017 at 6:56 am #

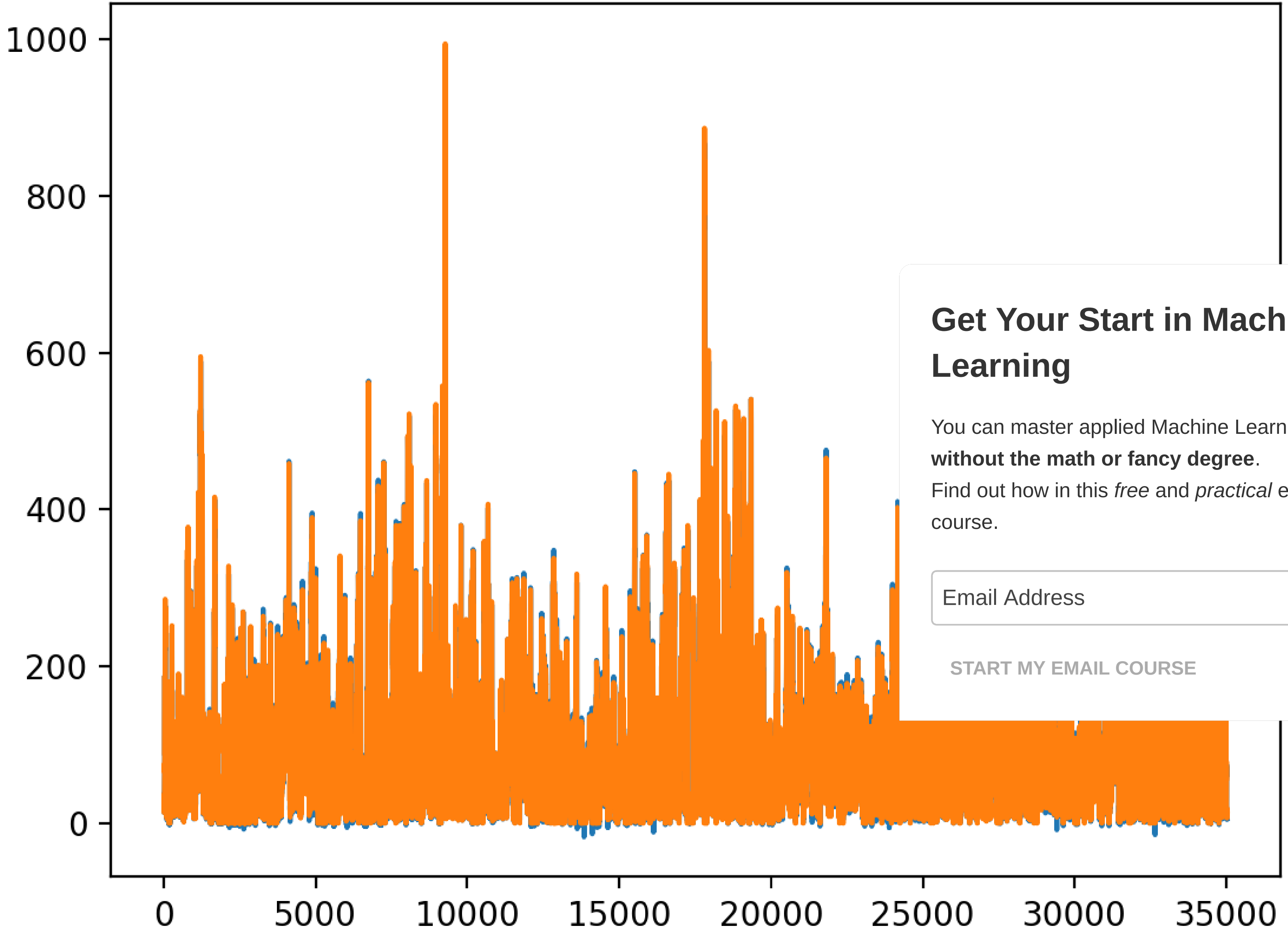
REPLY

Sure,

You can plot all predictions as follows:

```
pyplot.plot(inv_yhat)
pyplot.plot(inv_y)
pyplot.show()
```

You get:


1000
800
600
400
200
0
0
5000
10000
15000
20000
25000
30000
35000

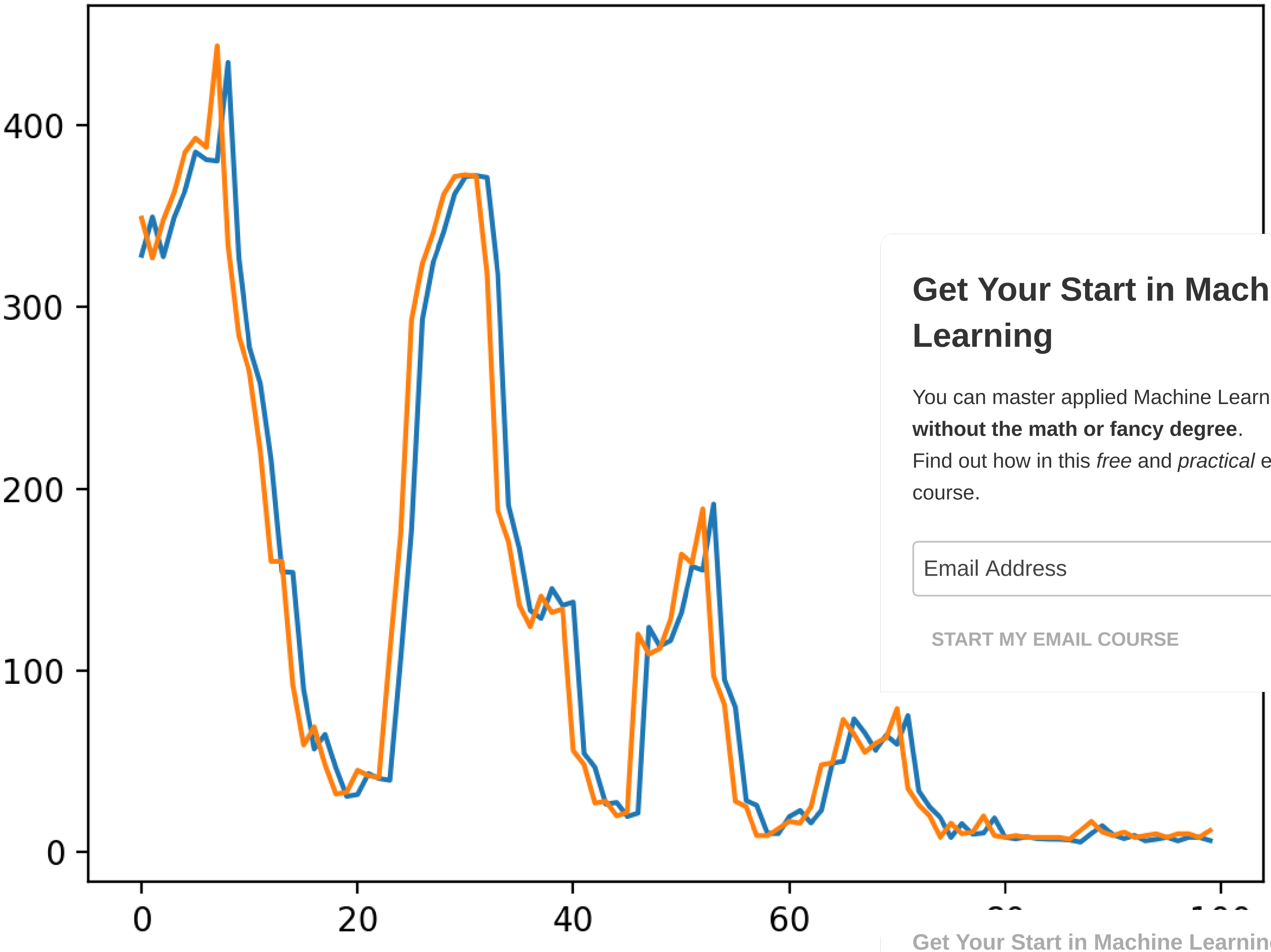
It’s a mess, you can plot the last 100 time steps as follows:

[redacted]

```
pyplot.plot(inv_yhat[-100:])
pyplot.plot(inv_y[-100:])
pyplot.show()
```

You get:

[redacted]



[redacted]

The predictions look like persistence.

BEN BECKER August 29, 2017 at 1:33 pm #

REPLY

Jason, what am I missing, looking at your plot of the most recent 100 time steps, it looks like the predicted value is always 1 time period after the actual? If on step 90 the actual is 17, but the predicted value shows 17 for step 91, we are one time period off, that is if we shifted the predicted values back a day, it would overlap with the actual which doesn't really buy us much since the next hour prediction seems to really align with the prior actual. Am I missing something looking at this chart?

Jason Brownlee August 29, 2017 at 5:16 pm #

This is what a persistence forecast looks like, that value(t) = value(t-1).

BECKER August 29, 2017 at 9:22 pm #

So how would you get the true predicted value(t)? I am thinking of the last reco e value for the next hour.

Jason Brownlee August 30, 2017 at 6:15 am #

Sorry, I don't follow. Perhaps you can restate your question?

Anna October 2, 2017 at 4:38 pm #

Hello Jason Brownlee

[redacted]

Thank you for your great posts. I run the model above for my data and it works perfectly, how ever when I draw the real data (blue one – inv_y) and the prediction (the orange one – inv_yhat), the result shows the prediction is delay after 1 step. it should be predicted one step before as your graph. your model is the same with the matlab tool: https://nl.mathworks.com/videos/maglev-modeling-with-neural-time-series-tool-68797.html

And after running the model, I applyed realtime this model for my problem to compute the inv_yhat in every step. I got the result is really bad, since I have never had the real inv_y. I took the prediction to feed the input (instead of real data inv_y)

My problem is: I received some signals as inputs, then I labeled offline to have output (real data inv_y or the first column in train_X)

Do you have the model that trains without the real data in the first column?????? thank you

Jason Brownlee October 3, 2017 at 5:40 am #

Your model may have low skill and be simply predicting the input as the output

You may need to continue to develop your model, I list some ideas for lifting model skill I
http://machinelearningmastery.com/improve-deep-learning-performance/

gammarayburst August 24, 2017 at 11:32 pm #

Wind dir is label encoded not wind speed!!!

Jason Brownlee August 25, 2017 at 6:43 am #

REPLY

Yes.

Filipe August 27, 2017 at 4:16 am #

REPLY

First of all, thanks. All of this material on the blog is super interesting, and helpful and making me learn a lot.

Of course… I have a question.

I'm surprised by the use of LSTMs here. The property of them being "stateful" I guess is being used. But is there "sequence" information flowing?

So when I used LSTMs in Keras for text classification tasks (sentence, outcome), each "sentence" is a sequence. Each observation is a sequence. It's an ordered array of the words in the sentence (and it's outcome).
In this example, I could not see a sense in which var1(t-1) is linked to var1(t-2). Aren't they being treated as independent Xs in a regression problem? (predicting var8(t))

Jason Brownlee August 27, 2017 at 5:53 am #

Correct, we are not providing a sequence of observations and therefore not getting good Bl

Based on my tests, I have found LSTMs to be poor at autoregression, and in this case, as I added m performance degraded.

I would strongly encourage you to use an MLP baseline that any MLP would have to out-perform.

See this post for more on the limitations of LSTM for time series:
https://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-foreca

STYLIANOS IORDANIS August 27, 2017 at 5:23 am #

REPLY

Awesome article, as always.
Btw, what is your view on using an autoencoder/ restricted Boltzmann layer compressing features/ features before feeding an LSTM network ? For example, if one has a financial timeseries to forecast, e.g. a classifier trying to predict increase or decrease in a look ahead time window, via numerous technical indicators and/or other candidate exogenous leading indicators…..
Could you write an article based on that idea?

[redacted]

Jason Brownlee August 27, 2017 at 5:53 am # REPLY

I have seen better results from large MLPs, nevertheless, try it and see how you go.

STYLIANOS IORDANIS August 27, 2017 at 7:25 am # REPLY

autoencoder/ restricted Boltzmann layers also deal with multicollinearity issues… do MLPs also deal with multicollinearity if you have multicollinearity in the features, right?

Jason Brownlee August 28, 2017 at 6:46 am #

MLPs are more robust to multicollinearity than linear models.

Hee Un August 29, 2017 at 12:28 am #

Hi, I am always amazed at your article. Thank you.
I have a question.
Is this LSTM code now weighted for each features?
Nowdays, I'm predicting precipitation, that is the trend is correct, but the amount is not right.
What's wrong with that?:(

Jason Brownlee August 29, 2017 at 5:06 pm # REPLY

Thanks!

Sorry, I'm not sure I understand the question, perhaps you could rephrase it?

I can say that I would expect better skill if the data was further prepared – e.g. made stationary.

[redacted]

Vipul August 30, 2017 at 7:53 pm #

REPLY ↩

Hi Jason,

Thanks for wonderful explanation!
Could you please help me to understand dimensionality reduction concept. Should PCA or statistical approach be used before feeding the data to LSTM OR LSTM will learn correlation with the inputs provided on its own? how to approach regression problem in LSTM when we have large set of features?

Your reply is greatly appreciated!

Jason Brownlee August 31, 2017 at 6:18 am #

Generally, if you make the problem simpler using data preparation, the LSTM or any model

Nader August 31, 2017 at 2:42 am #

How can I predict a single input ?
for example :

[0.036, 0.338, 0.197, 0.836, 0.333, 0.128, 0.00000001, 0.0000001]

how do i reshape and do a model.predict () ?

Thank you

Jason Brownlee August 31, 2017 at 6:23 am #

REPLY ↩

Perhaps this post will make it clearer:
https://machinelearningmastery.com/make-predictions-long-short-term-memory-models-keras/

[redacted]

Nader August 31, 2017 at 12:48 pm #

REPLY

Thank you, Jason.
I applied:

my_x = np.array([0.036, 0.338, 0.197, 0.836, 0.333, 0.128, 0.00000001, 0.0000001])
print(my_x.shape) # (8,)
my_x = my_x.reshape((1, 1, 8))
my_pred = model.predict(my_x)
print(my_pred)

The answer is the "scaled" answer which is 0.03436

I tried applying the scaler.inverse_transform(my_pred) to GET the actual number

But I get the following error:

on-broadcastable output operand with shape (1,1) doesn't match the broadcast shape (1,8)

Thank you

Jason Brownlee September 1, 2017 at 6:40 am #

Yes, the transform requires data in the same form as when you "fit" it.

David September 23, 2017 at 3:27 pm #

Then what if I use multi-time step prediction? (use several lags for prediction)
The y_hat and X_test can not have the same dimension.

[redacted]

Jason Brownlee September 24, 2017 at 5:13 am #

If the size of X or y must vary, you can use padding.

Fejwin August 31, 2017 at 3:52 am #

REPLY ↩

Hi Jason,
Thanks for the tutorial!
Maybe I missed something, but it seems that you provided the model with all of remaining data as 'testd
pointless, since we should be interested in predicting unknown data in the future, instead of data that the
sense to try the model to predict a first timestep into the future that neither the training nor the test data h
training data, but no test data, and afterwards ask it to predict first time step after training data?) How wo

Jason Brownlee August 31, 2017 at 6:25 am #

The model is fit on the training data, then makes a prediction for each step in the test data.
prior to making each prediction.

Normally we would use walk-forward validation:
https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

I did use walk forward validation on other LSTM examples (use the blog search) but it confuses readers more than helps it seems.

David September 24, 2017 at 1:01 pm #

REPLY ↩

Can I use part of trainX to predict testY ? (lags needed to predict testY is in trainX) Not sure if it is a logical way to do it

[redacted]

Jason Brownlee September 25, 2017 at 5:36 am # REPLY

Yes.

hadi September 1, 2017 at 12:08 pm # REPLY

Dear Jason Brownlee,

I have a little different question, Actually I have a sequence of characters as input and I want to project it into a multidimensional space.
I mean I want to project each sequence of chars (let say word) to an vector of 100 real numbers along my corpus, so my input is a sequence of chars (any
char-emedding is welcome) and my output is a vector for each sequence (which is a word) and Im really
I would appreciate if you give any clue help or sample code to define my model.

Thanks a lot in advance.

Jason Brownlee September 1, 2017 at 3:26 pm #

Keras provides an Embedding layer that you can use directly:
https://keras.io/layers/embeddings/

Sai k September 2, 2017 at 12:12 am # REPLY

Hi Jason,

Thanks for the wonderful tutorial!
Could you please explain how to deal the problem when situation is “Predict the pollution for the complete month (assume month has 30 days. t+1…t+30) and given the “expected” weather features for that month…assuming we have been provided historic data of pollution and weather data on daily basis”

How should the data be prepared and how it should be feed into LSTM?

As I new to LSTM model, I have problem understanding the data preparation and feeding to LSTM.

[redacted]

Thanks in advance for your response

Jason Brownlee September 2, 2017 at 6:11 am #

REPLY

Predicting for a month is called multi-step forecasting.

Here is a post on the general approach:
https://machinelearningmastery.com/multi-step-time-series-forecasting/

Here is an example of doing multi-step forecasting with an LSTM:
https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-netw

Adrian September 5, 2017 at 5:29 am #

Hi Jason,

Thanks for sharing. I added accuracy info to model while training using ' metrics=['accuracy'] '.

So model.compile(loss='mae', optimizer='adam') becomes :

model.compile(loss='mae', optimizer='adam', metrics=['accuracy'])

This adds acc & val_acc to output. After 100 epochs the acc value appears quite low : (0.0761) :
Epoch 100/100
1s – loss: 0.0143 – acc: 0.0761 – val_loss: 0.0132 – val_acc: 0.0393

The accuracy of the model appears very low ? Is this expected ?

Further info on acc & val_acc values : https://github.com/tflearn/tflearn/issues/357 "acc is the accuracy of a batch of training data and val_acc is the accuracy of a batch of testing data."

Jason Brownlee September 7, 2017 at 12:38 pm # REPLY

This is a regression problem. Accuracy does not make sense.

Eric H September 5, 2017 at 6:33 am # REPLY

Hi Jason, I've recently discovered your site and have been so pleased with your information – thank you. I've been trying to model data which is much like the air quality data described here, but every few time steps there will be a change in the number of features present.
Example: in my data a time step = 1 day and a sequence can be 800 – 1200 days long. Normally the data consists of features
– pm2.5: PM2.5 concentration
– DEWP: Dew Point
– TEMP: Temperature
– PRES: Pressure
– cbwd: Combined wind direction
– Iws: Cumulated wind speed
– Is: Cumulated hours of snow
– Ir: Cumulated hours of rain

But then every (random-ish amount of time) there will be an additional number of features for a day and

I've no idea on how to handle variable feature length. I've seen and played with plenty of variable seque
sequenceS and features. I'd love your input!
Thanks!
-Eric

Jason Brownlee September 7, 2017 at 12:40 pm # REPLY

You will need to normalize the number of features to be consistent for all time.

Eric Hiller September 10, 2017 at 5:21 am # REPLY

Is it possible to use (what in TensorFlow – land is called) SparseFeatures or SparseTensors to represent sparse datasets, or is there a fundamental issue with handling sparse datasets within RNNs?

Jason Brownlee September 11, 2017 at 12:04 pm # REPLY

Good question, I'm not sure off the cuff. Keras may support sparse numpy arrays – try it and see?

Ali Haidar September 8, 2017 at 1:56 am #

Hi Jason,

Thanks for the amazing articles. They are really helpful.

Lets say I want to forecast with lead 2. I mean by that forecasting values at time t using t-2 values, witho
reframed after running function series_to_supervised right ? To remove all columns with values t-1?
reframed.drop(reframed.columns[…])

Thanks

Jason Brownlee September 9, 2017 at 11:46 am # REPLY

Yep, looks good.

Inna September 11, 2017 at 7:53 pm # REPLY

[redacted]

Hello!
Thanks for articles.

I have a question related with time series. Is it possible to forecast all variables? For example, I have 'pollution', 'dew', 'temp', 'press', 'wnd_dir', 'wnd_spd', 'snow', 'rain' and want to predict all of them for the next hour. We know about trends and common rules (because of data amount: few years), so we can do forecasting. Where can I find more info about it?

Jason Brownlee September 13, 2017 at 12:22 pm # REPLY

Yes, this example can be modified to predict each variable.

appreciator September 12, 2017 at 10:59 am #

Thank you Jason for the great tutorial! I'm adapting it for different data, and i'm trying to use >1
the series-to-supervised: Since the first loops ends at 0 and the last loops starts at 0, won't there be two

Jason Brownlee September 13, 2017 at 12:26 pm #

No, try it with the data and see.

Eric September 12, 2017 at 11:49 am # REPLY

Hi Jason,

Thanks for the tutorial. I had just one question though.
I've seen tutorial using multivariate time series to train a lot of dataset (all have correlation between each other) at the same time and were able to predict for each dataset used.

For sake of argument let's say than one of the dataset is broke, the sensor that get the information to fee[redacted] the column of data only have 0 instead of whatever value). Do you think that we could use the other spot to continue to predict the broken one? (there is correlation between them and there would be a lot of non broken data from before the bug)

Best regards,

Jason Brownlee September 13, 2017 at 12:27 pm # REPLY ↩

Yes, you could try it and see. Or impute the missing data and see if that is better.

Eric September 14, 2017 at 2:22 pm #

Thank you Jason,

I shall try that as soon as possible.I guess that the overall accuracy will lower for every set predi every spot data set and predict each of them (with possibility to predict a broken one)) so one sp accuracy no?

Best regards,

Jason Brownlee September 15, 2017 at 12:10 pm #

It will.

Shan September 13, 2017 at 3:46 am # REPLY ↩

Is there any time parser like date parser? I am working with data which is in milliseconds.

Jason Brownlee September 13, 2017 at 12:33 pm # REPLY

It can handle parsing dates and times I believe.

kumar September 13, 2017 at 10:00 pm # REPLY

i got this error when i tried to run the program

pyplot.plot(history.history['val_loss'], label='test')
KeyError: 'val_loss'

Jason Brownlee September 15, 2017 at 12:05 pm #

Ensure you copy all of the code.

Simon September 15, 2017 at 9:55 pm #

Hi Jason,

Wouldn't it be better to scale the data after you run the series_to_supervised function? As it stands now, dimensions don't line up anymore.

Jason Brownlee September 16, 2017 at 8:41 am # REPLY

It would, but the scaling would be column-wise and incorrect.

[redacted]

REPLY

Simon September 17, 2017 at 11:26 am #

Could you expand more on this and how the code might be modified to incorporate multi-step? I'm also playing around with turning this into a classification problem, would it still work if the feature we are trying to predict is a classifier?

Jason Brownlee September 18, 2017 at 5:42 am #

REPLY

I give the code to do this in another comment.

For classification, you will need to change the number of neurons in the output layer, the activation function in the output layer and the loss function.

Agrippa Sulla September 16, 2017 at 5:18 am #

I have a little question. I've successfully built my own LSTM multivariate NN using your code as
UK using past export growth and GDP. It perform decently but the financial crisis kinda messes things u

Now I want to add data to this model, but I can't go further back than 1980 for the time-series (not for no
growth rate of all the UK's major trading partners. Should I be worried about adding another 20 input ne
about the risks of using data that is low in rows (e.g. years) but high in columns (e.g. inputs).

I hope my question makes sense.

Cheers

Jason Brownlee September 16, 2017 at 8:46 am #

REPLY

I don't have posts on the topic of more columns than rows. It does require careful handling.

As a start, I would recommend developing a strong test harness, then try adding data and see how it impacts the model skill. Experiment

Ed September 16, 2017 at 6:00 am #

REPLY

Jason
Thanks a lot for your tutorial!
Is there a feature importance plot for cases like this?
sometimes is very important to know it

Jason Brownlee September 16, 2017 at 8:47 am #

REPLY

Good question. I'm not sure about feature importance plots for LSTMs. I would expect that n it could be calculated for LSTMs, but this is not something I have looked into sorry.

Ed September 17, 2017 at 2:49 am #

Thanks a lot, Jason!

Jason Brownlee September 17, 2017 at 5:29 am #

No problem.

Kuldeep September 20, 2017 at 12:53 am #

REPLY

Hi Jason,

Great post as always!

I have a question regarding scaling. My problem is quite different as I have to apply series to supervised [redacted] and then combine the data… my question is, can I apply scaling at the end? Should scaling be applied column wise or on complete matrix/array?

Jason Brownlee September 20, 2017 at 5:58 am # REPLY

The key is being able to scale the data consistently. The place in the pipeline is less important.

Nejra September 21, 2017 at 1:25 am # REPLY

Hi Jason thank you very much for your tutorials!
I'm trying to develop an LSTM for time prediction having as input 3 features (2 measurements and a thir t
(value to predict) is not a single value but a vector of 6 values. So, at every time step my network should
1. Since my inputs are not correlated between them, their order in the input array will not influence my pr
2. How can I shape my output in order to estimate all the 6 values of the vector for each time step?
Thanks for any kind of help!

Jason Brownlee September 21, 2017 at 5:51 am #

This post will help you understand how to prepare data for multi-step forecasting:
https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-netv

Mitchel Myers September 22, 2017 at 5:34 am # REPLY

I replicated the example described on this page, and saved my test_y and yhat vectors to csv so that I could manually check how my prediction compared with the true values. However, when I did this, I discovered that every yhat value in my array is the exact same value (~34). I was expecting a unique yhat value for each input vector. Do you have any suggestions to help fix this?

[redacted]

Mitchel Myers September 23, 2017 at 3:25 am #

REPLY

Follow up on this — when this error arose, I was using my own data set that I want to perform time series forecasting on. When I duplicated the guide exactly as described above, the issue goes away. Do you have any idea why this issue comes up (where every predicted yhat value is the exact same) when I use a different data set?

Jason Brownlee September 23, 2017 at 5:44 am #

REPLY

Perhaps the model needs to be tuned to your specific dataset?

zwj September 25, 2017 at 1:10 pm #

Hi Jason thank you very much for your tutorials! I try to delete the columns ['dew', 'temp', 'press
data, and I also get the almost same test RMSE. It is 26.461. It seems to show that the 8 weather condit
is below.

```
# fit an LSTM network to training data
def fit_lstm(train, test, batch_size, neurons):
# split into input and outputs
train_X, train_y = train[:, 0:1], train[:, -1]
test_X, test_y = test [:, 0:1], test [:, -1]

train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)

# design network
model = Sequential()
model.add(LSTM(neurons, input_shape=(train_X.shape[1], train_X.shape[2])))
```

[redacted]

```
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')

# fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=batch_size, validation_data=(test_X, test_y), verbose=2, shuffle=False)
#history = model.fit(train_X, train_y, epochs=50, batch_size=72, verbose=2, shuffle=False)

return model

# make a prediction
def make_forecasts(model, test_X):
test_X = test_X[:, 0:1]
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
forecasts = model.predict(test_X)

return forecasts
```

Jason Brownlee September 25, 2017 at 3:26 pm #

Nice one!

The real motivation for me writing this post was to help the 100s of people asking how to develop a r

Mitchel September 27, 2017 at 1:39 am #

Can you explain why the train_X and test_X data sets are reshaped to this?

```
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
```

Jason Brownlee September 27, 2017 at 5:44 am #

The shape is: samples, time steps, features.

[redacted]

Lino September 28, 2017 at 12:59 pm #

REPLY ↩

Hi Jason

Great post.
Suppose i want to predict the next 24h using previous one year dataset. How can we do it?
Thanks

Jason Brownlee September 28, 2017 at 4:45 pm #

I give an example in another comment.

Also, generally, see this post on multi-step forecasting with LSTMs:
https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-netw

Nels September 29, 2017 at 5:56 am #

I think I'm missing something fundamental in my understanding of LSTM/s and BPTT. I've read
understand RNN's and LSTM in particular much better because of them, so thank you for that!

My question that I hope you can shed some light on is what is the difference between passing the past information, i.e. var(t-n)…var(t-1) in the input vector for a single sample, and passing multiple sequences, of length n as a single sample?

To help clarify, using temsteps of length N, I have a configuration that looks like this:

Input to LSTM is [samples, timesteps, features].
Each sample/observation consists of a vector of timestamps (of size N+1) where each of these vector's values corresponds to the input feature's values I.e.

Observations for each time t, with features f and r
[

time t
[
[f(t-N) r(t-N)]
[f(t-N+1) r(t-N+1)]
[f(t-N+2) r(t-N+2)]
. .
. .
. .
[f(t) r(t)]
]
]
And for each observation/sequence the target is Y(t).

Or, as many of your examples do, you can include the the past information in the form of a windowed inp

Input is [samples, 1, features]. So for every observation, we include previous time values as features

Observations for each time t, with features f and r
[
time t
[
[f(t-N), r(t-N), f(t-N+1), r(t-N+1), f(t-N+2), r(t-N+2), f(t), r(t)]
]
]
And again, for each observation, the target is Y(t).

I understand that having sequences longer than 1 allows BPTT to work over the length of those sequenc
in these two methods.

I have tried the described two options, and I find the the latter is performing better based on preliminary tests. I can use a window size of 3 and a sequence length of 1 and get good results, but if I use the first approach and a window size of 12, the model actually fails to learn within the same amount of time.

Hence, I wonder if I don’t have a fundamental misconception. If you have some time, I would like to hear your explanation on this difference and how the LSTM responds in terms of “memory” based on these two different types of input setup. (I have read a lot of articles, blogs, git hub issues, and stack overflow posts trying to wrap my head around this, but I haven’t found anything that address this directly.)

Thanks!

[redacted]

Jason Brownlee September 30, 2017 at 7:31 am # REPLY

Generally, the multiple steps for one sequence are required for BPTT:
https://machinelearningmastery.com/gentle-introduction-backpropagation-time/

Without the history, the training will not have sufficient context to estimate the error gradient and your model will learn a function mapping rather than a sequence prediction problem.

Does that help?

Paul September 29, 2017 at 12:28 pm #

With this line…

# drop columns we don't want to predict
reframed.drop(reframed.columns[[9,10,11,12,13,14,15]], axis=1, inplace=True)

I don't understand the numbers used here, doesn't the data not even have that many columns? There a

I'm adapting this code for my own use and have very different features but I'm not sure I'm getting that li

Thanks for the great post!

Paul September 29, 2017 at 1:29 pm # REPLY

Nevermind! I figured it out.

Jason Brownlee September 30, 2017 at 7:34 am # REPLY

Glad to hear it Paul.

Jason Brownlee September 30, 2017 at 7:33 am # REPLY

It does have that many columns after we reshape it to be a supervised learning problem.

Wenhan Wang September 30, 2017 at 2:05 pm # REPLY

This is awesome!
Helping me a lot in my real work!

Jason Brownlee October 1, 2017 at 9:07 am #

Thanks, I'm glad to hear that.

Vilmara Sanchez October 4, 2017 at 3:54 pm #

Hi Dr. Jason, I am working on a project for sleep stage classification where the number of times s different than the number of timesteps in the output series (sleep stage scores).

The issue here is that the input and output time series are not equal in terms of timesteps as the examples you have shown in your problems.

I have tried to frame the problem in different ways without getting results that make sense. Could you please provide guidance on how to approach this problem?.

Thanks,

Vilmara

[redacted] REPLY ↩

Jason Brownlee October 5, 2017 at 5:21 am #

Generally, I would recommend an encoder-decoder model:
https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

Devakar Verma October 6, 2017 at 6:06 pm #

REPLY ↩

Hi Jason,
If we want to predict multiple features as output and having multiple feature as input. How can we solve this problem. For example input variables are temperature and humidity and want to predict both temperature and humidity, can we solve this with single LSTM model.

Thanks for your anticipated response.

Jason Brownlee October 7, 2017 at 5:50 am #

Yes you can. Change the multivariate input model to output more than one value in the outp

Brent October 7, 2017 at 5:55 am #

Hi Jason,

Thank you for taking the time to write such an excellent post and follow up with questions. The mechanics of the data conversion & training work great.

However, my first reaction is that the LSTM doesn’t seem to have learned anything more than to copy the previous value. As BECKER states:

> it looks like the predicted value is always 1 time period after the actual?

These are the same results as in your Shampoo example: the predicted value appears to be equal to the previous value (possibly with some constant offset).

Have you found a different network architecture that performs better than a DNN without LSTM layers?

Jason Brownlee October 7, 2017 at 5:58 am # REPLY

Agreed, LSTMs do not seem to be very good for autoregression. I would generally recommend using an MLP with a window for time series forecasting instead.

See this post:
https://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecasting/

sathvik October 9, 2017 at 1:34 pm # REPLY

Thank you so much Jason for the wonderful article, learnt a lot… I wanted to have a compariso l networks and I was looking for some post/article on multivariate time series model using ARIMA. I would

Thank you

Jason Brownlee October 9, 2017 at 4:46 pm #

You will need to look into using SARIMAX, sorry I do not have an example at this stage.

Leave a Reply

[redacted]

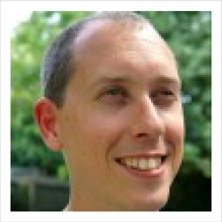
Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery



Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More

Deep Learning for Sequence Prediction

Cut through the math and research papers.
Discover 4 Models, 6 Architectures, and 14 Tutorials.

[redacted]

POPULAR

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras

JULY 21, 2016

Your First Machine Learning Project in Python Step-By-Step

JUNE 10, 2016

Develop Your First Neural Network in Python With Keras Step-By-Step

MAY 24, 2016

Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras

JULY 26, 2016

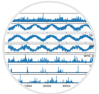
How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda

MARCH 13, 2017

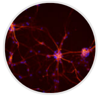
Multi-Class Classification Tutorial with the Keras Deep Learning Library
JUNE 2, 2016

Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016



Multivariate Time Series Forecasting with LSTMs in Keras
AUGUST 14, 2017



How to Implement the Backpropagation Algorithm From Scratch In Python
NOVEMBER 7, 2016

Get Your Start in Machine Learning