Need help with LSTMs in Python? Take the FREE Mini-Course.

How to Reshape Input Data for Long Short-Term Memory Networks in Keras

by **Jason Brownlee** on August 30, 2017 in **Long Short-Term Memory Networks**

It can be difficult to understand how to prepare your sequence data for input to an LSTM model.

Often there is confusion around how to define the input layer for the LSTM model.

There is also confusion about how to convert your sequence data that may be a 1D or 2D matrix of numbers to the required 3D format of the LSTM input layer.

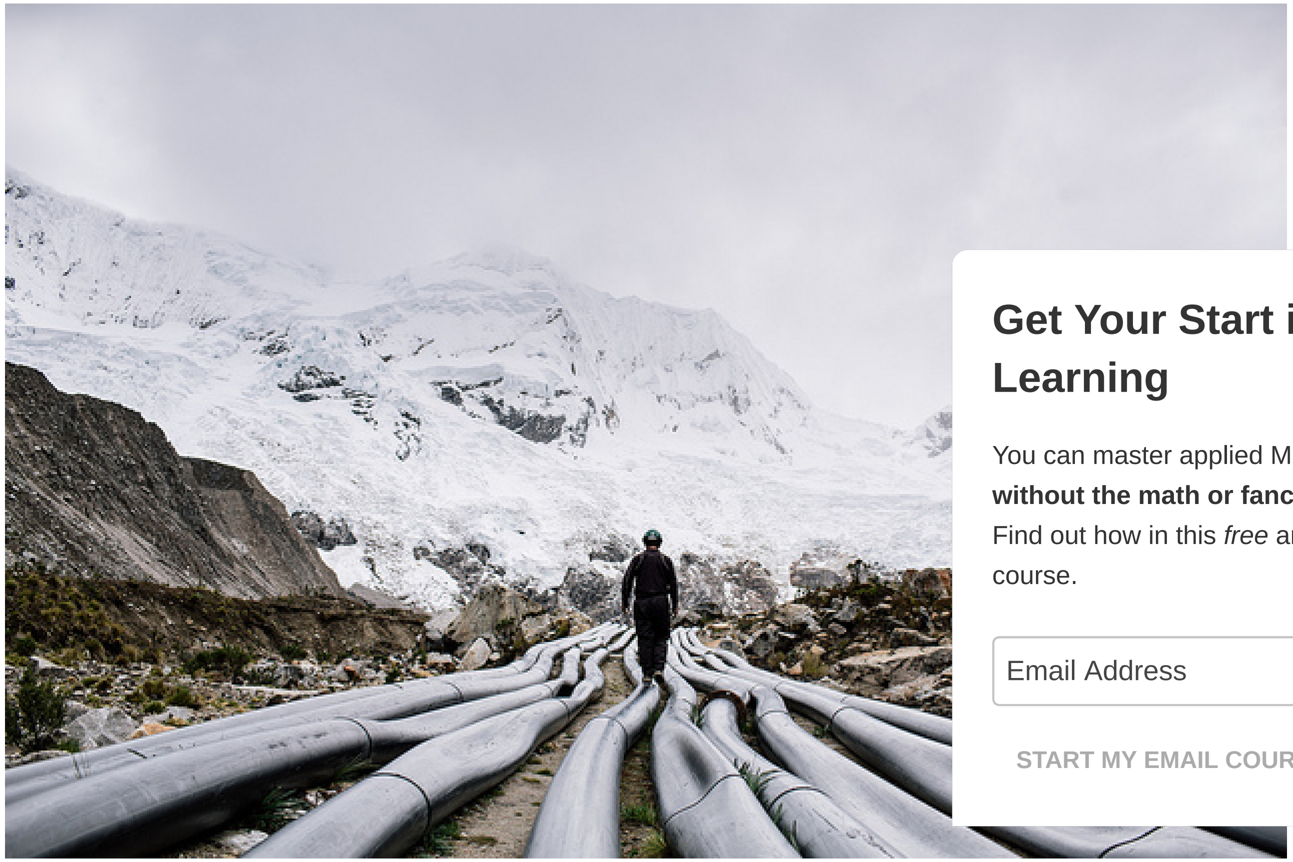
In this tutorial, you will discover how to define the input layer to LSTM models and how to reshape your loaded input data for LSTM models.

After completing this tutorial, you will know:

[redacted]

- How to define an LSTM input layer.
- How to reshape a one-dimensional sequence data for an LSTM model and define the input layer.
- How to reshape multiple parallel series data for an LSTM model and define the input layer.

Let’s get started.



How to Reshape Input for Long Short-Term Memory Networks in Keras
Photo by Global Landscapes Forum, some rights reserved.

Tutorial Overview

This tutorial is divided into 4 parts; they are:

LSTM Input Layer

The LSTM input layer is specified by the “*input_shape*” argument on the first hidden layer of the network.

This can make things confusing for beginners.

For example, below is an example of a network with one hidden LSTM layer and one Dense output layer.

```
model = Sequential()
model.add(LSTM(32))
model.add(Dense(1))
```

In this example, the LSTM() layer must specify the shape of the input.

The input to every LSTM layer must be three-dimensional.

The three dimensions of this input are:

- **Samples**. One sequence is one sample. A batch is comprised of one or more samples.
- **Time Steps**. One time step is one point of observation in the sample.
- **Features**. One feature is one observation at a time step.

This means that the input layer expects a 3D array of data when fitting the model and when making contain a single value, e.g. one sample or one feature.

When defining the input layer of your LSTM network, the network assumes you have 1 or more samples and requires that you specify the number of time steps and the number of features. You can do this by specifying a tuple to the “*input_shape*” argument.

For example, the model below defines an input layer that expects 1 or more samples, 50 time steps, and 2 features.

```
model = Sequential()
```

```
model.add(LSTM(32, input_shape=(50, 2)))
model.add(Dense(1))
```

Now that we know how to define an LSTM input layer and the expectations of 3D inputs, let's look at some examples of how we can prepare our data for the LSTM.

Example of LSTM With Single Input Sample

Consider the case where you have one sequence of multiple time steps and one feature.

For example, this could be a sequence of 10 values:

```
0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
```

We can define this sequence of numbers as a NumPy array.

```
from numpy import array
data = array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
```

We can then use the *reshape()* function on the NumPy array to reshape this one-dimensional array i me steps, and 1 feature at each time step.

The *reshape()* function when called on an array takes one argument which is a tuple defining the ne of numbers; the reshape must evenly reorganize the data in the array.

```
data = data.reshape((1, 10, 1))
```

Once reshaped, we can print the new shape of the array.

```
print(data.shape)
```

Putting all of this together, the complete example is listed below.

```
from numpy import array
data = array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
data = data.reshape((1, 10, 1))
print(data.shape)
```

Running the example prints the new 3D shape of the single sample.

```
(1, 10, 1)
```

This data is now ready to be used as input (*X*) to the LSTM with an input_shape of (10, 1).

```
model = Sequential()
model.add(LSTM(32, input_shape=(10, 1)))
model.add(Dense(1))
```

Example of LSTM with Multiple Input Features

Consider the case where you have multiple parallel series as input for your model.

For example, this could be two parallel series of 10 values:

```
series 1: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
series 2: 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1
```

We can define these data as a matrix of 2 columns with 10 rows:

```
from numpy import array
data = array([
	[0.1, 1.0],
	[0.2, 0.9],
	[0.3, 0.8],
	[0.4, 0.7],
	[0.5, 0.6],
	[0.6, 0.5],
	[0.7, 0.4],
	[0.8, 0.3],
	[0.9, 0.2],
	[1.0, 0.1]])
```

This data can be framed as 1 sample with 10 time steps and 2 features.

It can be reshaped as a 3D array as follows:

```
data = data.reshape(1, 10, 2)
```

Putting all of this together, the complete example is listed below.

```
from numpy import array
```

```
data = array([
    [0.1, 1.0],
    [0.2, 0.9],
    [0.3, 0.8],
    [0.4, 0.7],
    [0.5, 0.6],
    [0.6, 0.5],
    [0.7, 0.4],
    [0.8, 0.3],
    [0.9, 0.2],
    [1.0, 0.1]])
data = data.reshape(1, 10, 2)
print(data.shape)
```

Running the example prints the new 3D shape of the single sample.

```
(1, 10, 2)
```

This data is now ready to be used as input (*X*) to the LSTM with an input_shape of (10, 2).

```
model = Sequential()
model.add(LSTM(32, input_shape=(10, 2)))
model.add(Dense(1))
```

Tips for LSTM Input

This section lists some tips to help you when preparing your input data for LSTMs.

- The LSTM input layer must be 3D.
- The meaning of the 3 input dimensions are: samples, time steps, and features.
- The LSTM input layer is defined by the *input_shape* argument on the first hidden layer.
- The *input_shape* argument takes a tuple of two values that define the number of time steps and features.
- The number of samples is assumed to be 1 or more.
- The *reshape()* function on NumPy arrays can be used to reshape your 1D or 2D data to be 3D.
- The *reshape()* function takes a tuple as an argument that defines the new shape.

Further Reading

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

[redacted]

- Recurrent Layers Keras API
- Numpy reshape() function API
- How to Convert a Time Series to a Supervised Learning Problem in Python
- Time Series Forecasting as Supervised Learning

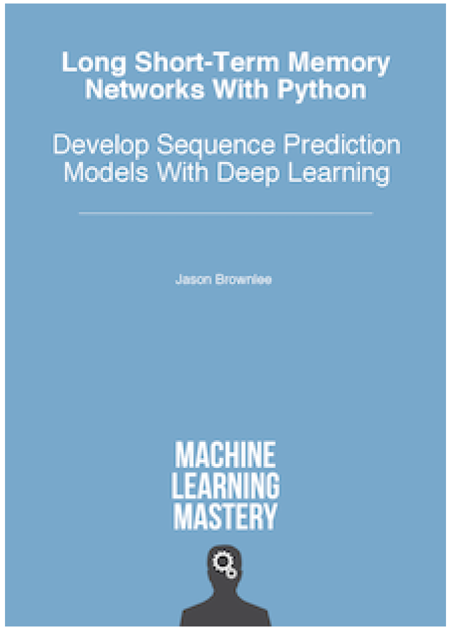
Summary

In this tutorial, you discovered how to define the input layer for LSTMs and how to reshape your sequence data for input to LSTMs.

Specifically, you learned:

- How to define an LSTM input layer.
- How to reshape a one-dimensional sequence data for an LSTM model and define the input laye
- How to reshape multiple parallel series data for an LSTM model and define the input layer.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.


Long Short-Term Memory Networks With Python
Develop Sequence Prediction Models With Deep Learning
Jason Brownlee
MACHINE LEARNING MASTERY

Click to learn mo [redacted]

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional devel
to helping developers get started and get good at applied machine learning. Learn more.
View all posts by Jason Brownlee →

< How to Make Predictions with Long Short-Term Memory Models in Keras

How to Diagnose Overfitting and Underfitting of LSTM Models >

12 Responses to *How to Reshape Input Data for Long Short-Term Memory Networks in Keras*

Steven August 31, 2017 at 2:14 am # REPLY

Great explanation of the dimensions! Just wanted to say this explanation also works for LSTM models in Tensorflow as well.

Jason Brownlee August 31, 2017 at 6:20 am # REPLY

Thanks Steven.

yuan September 1, 2017 at 6:42 pm #

Hi Jason,

Thanks a lot for your explanations .
I have a confusion below:
Assuming that we have multiple parallel series as input for out model.The first step is to define tl e
3D(samples, time steps, and features),is this means that,samples :1 sample ,time steps: row nu
the matrix ? Must it be like this?Looking forward to your reply.Thank you

Jason Brownlee September 2, 2017 at 6:06 am # REPLY

Sorry, I'm not sure I follow your question.

If you have parallel time series, then each series would need the same number of time steps and be represented as a separate feature (e.g. observation at a time).

Does that help?

[redacted]

Oliver August 31, 2017 at 9:23 pm #

REPLY

Hi Jason,

thanks a lot for all the explanations you gave!
I tried to understand the effect of the reshape parameters and the effect in the spyder/variable explorer. But I do not understand the result shown in the data window.
I used the code from a different tutorial:

data = array([
[0.1, 1.0],
[0.2, 0.9],
[0.3, 0.8],
[0.4, 0.7],
[0.5, 0.6],
[0.6, 0.5],
[0.7, 0.4],
[0.8, 0.3],
[0.9, 0.2],
[1.0, 0.1]])
data_re = data.reshape(1, 10, 2)

When checking the result in the variable explorer of spyder I see 3 dimensions of the array but can not c

On axis 0 of data_re I see the complete dataset
On axis 1 of the data_re I get 0.1 and 1.0 in column 1
On axis 2 of the data_re I see the column 1 of axis 0 transposed to row 1

Would you give me a hint how to interpret it?

Regards,
Oliver.

Jason Brownlee September 1, 2017 at 6:46 am # REPLY

There are no named parameters, I am referring to the dimensions by those names because that is how the LSTM model uses the data.

Sorry for the confusion.

Saga September 1, 2017 at 6:46 pm # REPLY

Hi Jason,

Thanks so much for the article (and the whole series in fact!). The documentation in Keras is not very cle

I have been trying to implement a model that receives multiple samples of multivariate timeseries as inpu
"time steps" dimension is different for different samples. I have tried to train a model on each sample ind
going to be extremely prone to overfitting). Another idea was to scale the samples to have the same time
steps for each sample which is not ideal either.

Is there a way to provide the LSTM with samples of dynamic time steps? maybe using a lower-level API

Regards,
Saga

Jason Brownlee September 2, 2017 at 6:07 am #

A way I use often is to pad all sequences to the same length and use a masking layer on the front end to ignore masked time steps.

Shrimanti September 14, 2017 at 2:42 am # REPLY

Hi Jason,

Thanks very much for your tutorials on LSTM. I am trying to predict one time series from 10 different par[redacted]
So, the shape of my X_train is (50000,10) and Y_train is (50000,1). I couldn't figure out how to reshape my dataset and the input shape of LSTM if I want to use let's say 100 time steps or look back as 100.

Thanks.

Jason Brownlee September 15, 2017 at 12:08 pm #

REPLY

This post will help you formulates your series as a supervised learning problem:
https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

Anil Pise October 14, 2017 at 6:38 am #

Respected Sir
I want to use LSTM RNN GRU to check changes in facial expression of the person who is watching a m
boar or interested to continue this movie or at what time he is a boar. Can you please help me how can I

Jason Brownlee October 15, 2017 at 5:16 am #

That sounds like a great problem. I would recommend starting by collecting a ton of training

Then think of using a CNN on the front end of your LSTM.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Leave a Reply

Get Your Start in Machine Learning

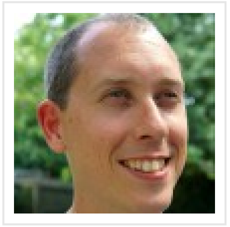
Name (required)

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.

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

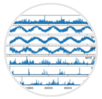
Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

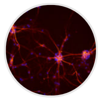
How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017

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

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

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

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Privacy | Contact | About

Get Your Start in Machine Learning