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

# Data Preparation for Variable Length Input Sequences

by **Jason Brownlee** on June 19, 2017 in **Long Short-Term Memory Networks**

Deep learning libraries assume a vectorized representation of your data.

In the case of variable length sequence prediction problems, this requires that your data be transformed such that each sequence has the same length.

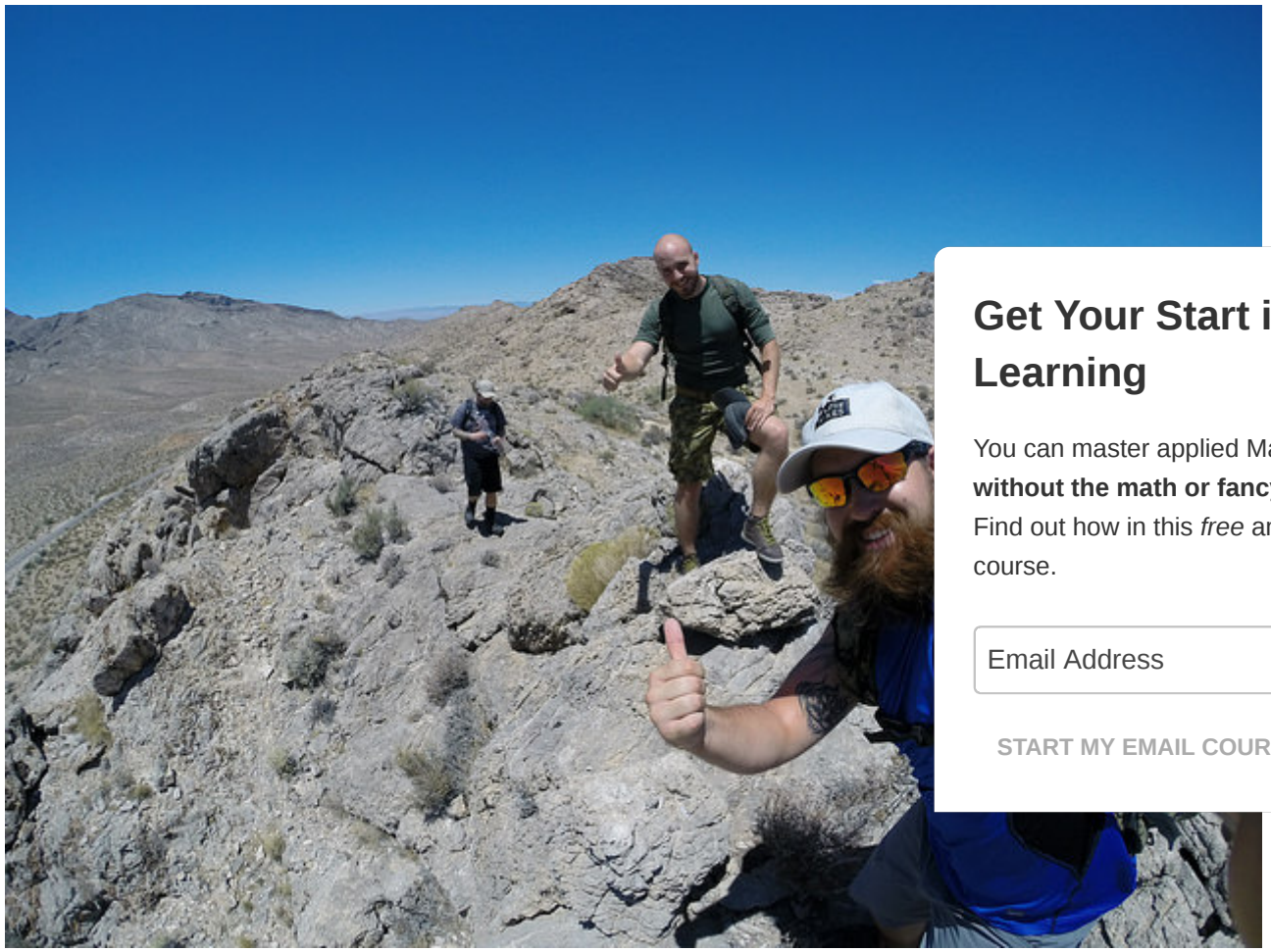
This vectorization allows code to efficiently perform the matrix operations in batch for your chosen deep learning algorithms.

In this tutorial, you will discover techniques that you can use to prepare your variable length sequence data for sequence prediction problems in Python with Keras.

After completing this tutorial, you will know:

- How to pad variable length sequences with dummy values.
- How to pad variable length sequences to a new longer desired length.
- How to truncate variable length sequences to a shorter desired length.

Let's get started.



Data Preparation for Variable-Length Input Sequences for Sequence Prediction
Photo by Adam Bautz, some rights reserved.



## Overview

This section is divided into 3 parts; they are:

1. Contrived Sequence Problem
2. Sequence Padding
3. Sequence Truncation

## Environment

This tutorial assumes you have a Python SciPy environment installed. You can use either Python 2 or 3 with this example.

This tutorial assumes you have Keras (v2.0.4+) installed with either the TensorFlow (v1.1.0+) or Theano (v0.9+) backend.

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

If you need help setting up your Python environment, see this post:

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



# Contrived Sequence Problem

We can contrive a simple sequence problem for the purposes of this tutorial.

The problem is defined as sequences of integers. There are three sequences with a length between

```
1 1, 2, 3, 4
2 1, 2, 3
3 1
```

These can be defined as a list of lists in Python as follows (with spacing for readability):

```
1 sequences = [
2 [1, 2, 3, 4],
3 [1, 2, 3],
4 [1]
5 ]
```

We will use these sequences as the basis for exploring sequence padding in this tutorial.

# Sequence Padding

The pad_sequences() function in the Keras deep learning library can be used to pad variable length

The default padding value is 0.0, which is suitable for most applications, although this can be change argument. For example:

```
pad_sequences(..., value=99)
```

The padding to be applied to the beginning or the end of the sequence, called pre- or post-sequence argument, as follows.

## Pre-Sequence Padding

Pre-sequence padding is the default (padding='pre')

The example below demonstrates pre-padding 3-input sequences with 0 values.

```
from keras.preprocessing.sequence import pad_sequences
# define sequences
sequences = [
    [1, 2, 3, 4],
       [1, 2, 3],
            [1]
```

```
    ]
# pad sequence
padded = pad_sequences(sequences)
print(padded)
```

Running the example prints the 3 sequences pre-pended with zero values.

```
[[1 2 3 4]
[0 1 2 3]
[0 0 0 1]
```

## Post-Sequence Padding

Padding can also be applied to the end of the sequences, which may be more appropriate for some problem domains.

Post-sequence padding can be specified by setting the “padding” argument to “post”.

```
from keras.preprocessing.sequence import pad_sequences
# define sequences
sequences = [
    [1, 2, 3, 4],
       [1, 2, 3],
             [1]
    ]
# pad sequence
padded = pad_sequences(sequences, padding='post')
print(padded)
```

Running the example prints the same sequences with zero-values appended.

```
[[1 2 3 4]
[1 2 3 0]
[1 0 0 0]]
```



## Pad Sequences To Length

The pad_sequences() function can also be used to pad sequences to a preferred length that may be longer than any observed sequences.

This can be done by specifying the “maxlen” argument to the desired length. Padding will then be performed on all sequences to achieve the desired length, as follows.

Get Your Start in Machine Learning

```
from keras.preprocessing.sequence import pad_sequences
# define sequences
sequences = [
    [1, 2, 3, 4],
       [1, 2, 3],
            [1]
    ]
# pad sequence
padded = pad_sequences(sequences, maxlen=5)
print(padded)
```

Running the example pads each sequence to the desired length of 5 timesteps, even though the maximum length of an observed sequence is only 4 timesteps.

```
[[0 1 2 3 4]
[0 0 1 2 3]
[0 0 0 0 1]]
```

## Sequence Truncation

The length of sequences can also be trimmed to a desired length.

The desired length for sequences can be specified as a number of timesteps with the “maxlen” argu

There are two ways that sequences can be truncated: by removing timesteps from the beginning or

### Pre-Sequence Truncation

The default truncation method is to remove timesteps from the beginning of sequences. This is calle

The example below truncates sequences to a desired length of 2.

```
from keras.preprocessing.sequence import pad_sequences
# define sequences
sequences = [
    [1, 2, 3, 4],
       [1, 2, 3],
            [1]
    ]
# truncate sequence
truncated= pad_sequences(sequences, maxlen=2)
```

```
10 print(truncated)
```

Running the example removes the first two timesteps from the first sequence, the first timestep from the second sequence, and pads the final sequence.

```
1 [[3 4]
2 [2 3]
3 [0 1]]
```

## Post-Sequence Truncation

Sequences can also be trimmed by removing timesteps from the end of the sequences.

This approach may be more desirable for some problem domains.

Post-sequence truncation can be configured by changing the “truncating” argument from the default

```
1  from keras.preprocessing.sequence import pad_sequences
2  # define sequences
3  sequences = [
4      [1, 2, 3, 4],
5         [1, 2, 3],
6               [1]
7      ]
8  # truncate sequence
9  truncated= pad_sequences(sequences, maxlen=2, truncating='post')
10 print(truncated)
```

Running the example removes the last two timesteps from the first sequence, the last timestep from
sequence.

```
1 [[1 2]
2 [1 2]
3 [0 1]]
```



# Summary

In this tutorial, you discovered how to prepare variable length sequence data for use with sequence prediction problems in Python.

Specifically, you learned:

- How to pad variable length sequences with dummy values.
- How to pad out variable length sequences to a new desired length.
- How to truncate variable length sequences to a new desired length.

Do you have any questions about preparing variable length sequences?
Ask your questions in the comments and I will do my best to answer.

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 →

‹ How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras

How to Handle Missing Timesteps in Sequence Prediction Problems with Python ›

## 10 Responses to *Data Preparation for Variable Length Input Sequences*

**Yuya** June 19, 2017 at 6:46 am #

Are there alternative methods which doesn't make of use padding as a way to handle sequence

**Jason Brownlee** June 19, 2017 at 8:49 am #

Yes, some ideas:

– You could truncate sequences.
– You could concatenate sequences.
– You could write your own inefficient implementation of RNNs.

**Tom** June 19, 2017 at 10:54 am #

REPLY

If you know the target length, you can try interpolation or warping, right? But they are more costly than just adding zeros.

**Jason Brownlee** June 20, 2017 at 6:33 am #

Nice tip Tom!

REPLY

**Sudipta Kar** June 23, 2017 at 7:57 am #

REPLY

I am interested how does the padding affect the performance of models? Specifically when the dataset contains both long texts and very short texts. Lets assume, median length is 30 sentences, max length is 120 and minimum length is 10. how pre and post sequence padding affects the performance?

**Rahul** June 24, 2017 at 3:06 am #

I am not wrong you usually batch together inputs of similar length, so in this case you would lengths ranging from 30-120

**Jason Brownlee** June 24, 2017 at 7:52 am #

Padding and Masking is the best, little to no impact. I would recommend testing though.

**Rahul** June 24, 2017 at 3:06 am #

REPLY

If* I

**Patrick** September 13, 2017 at 2:11 pm #

REPLY

Thank you so much for writing these tutorials. They are remarkably effective in explaining concepts that other websites often gloss over. I have a question about padding outputs in sequence-to-sequence classification problems. Let's say X has the shape (100, 50, 10), and y has the shape (100, 50, 3). X consists of 100 time series, 50 time steps per time series, and 10 features per time step. The y has three possible one-hot encoded classes per tilmestep. The samples of X have variable length so the shorter samples are pre-padded with 0 For the y labels corresponding to the pre-padded X time steps, should they be [0, 0, 0]? Or should a new fourth label, [0, 0, 0, 1], be created for the pre-padded time steps thus changing the shape of y to (100, 100, 4).

**Jason Brownlee** September 15, 2017 at 11:56 am #

REPLY

Thanks Patrick.

Why do you need to pad the output if each series is classified as one of 3 labels?



## Leave a Reply

Name (required)

Email (will not be published) (required)

Website

Get Your Start in Machine Learning

SUBMIT COMMENT

### Welcome to Machine Learning Mastery

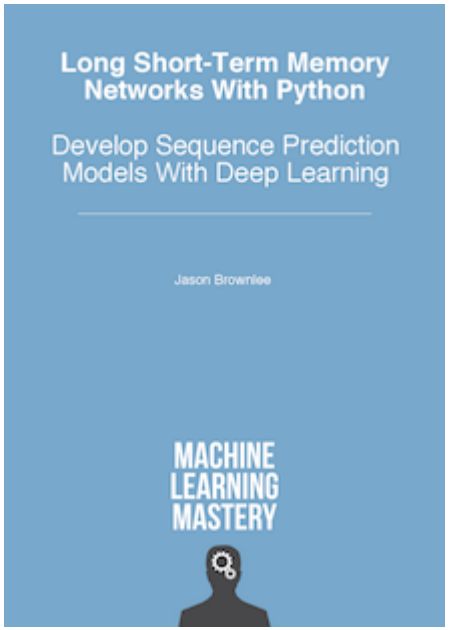
Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More

### Deep Learning for Sequence Prediction

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

Get Started With LSTMs in Python Today!



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

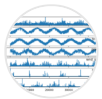

**Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras**
JULY 26, 2016

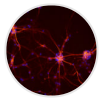

**How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda**
MARCH 13, 2017


**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