

[Start Here](#)[Blog](#)[Books](#)[About](#)[Contact](#)

Need help with LSTMs in Python? [Take the FREE Mini-Course.](#)

5 Examples of Simple Sequence Prediction Problems for Learning LSTM Recurrent Neural Networks

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



Sequence prediction is different from traditional classification and regression problems.

It requires that you take the order of observations into account and that you use models like Long Short-Term Memory (LSTM) recurrent neural networks that have memory and that can learn any temporal dependence between observations.

It is critical to apply LSTMs to learn how to use them on sequence prediction problems, and for that, you need a suite of well-defined problems that allow you to focus on different problem types and framings. It is critical so that you can build up your intuition for how sequence prediction problems are different and how sophisticated models like LSTMs can be used to address them.

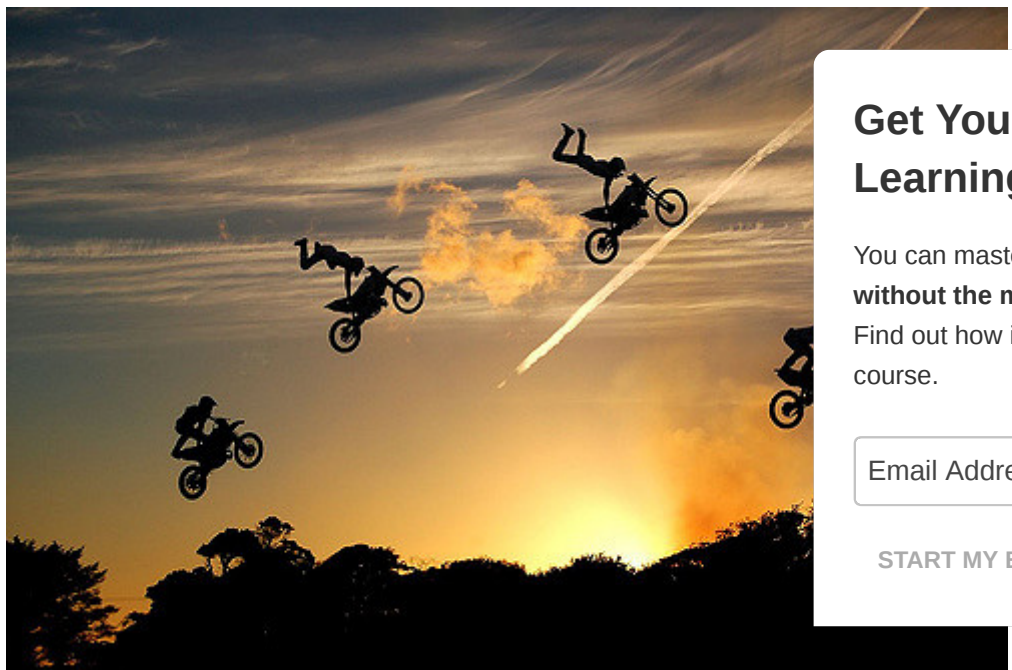
[Get Your Start in Machine Learning](#)

In this tutorial, you will discover a suite of 5 narrowly defined and scalable sequence prediction problems that you can use to apply and learn more about LSTM recurrent neural networks.

After completing this tutorial, you will know:

- Simple memorization tasks to test the learned memory capability of LSTMs.
- Simple echo tasks to test the learned temporal dependence capability of LSTMs.
- Simple arithmetic tasks to test the interpretation capability of LSTMs.

Let's get started.



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.

START MY EMAIL COURSE

5 Examples of Simple Sequence Prediction Problems for Learning LSTM Recurrent Neural Networks

Photo by [Geraint Otis Warlow](#), some rights reserved.

Tutorial Overview

This tutorial is divided into 5 sections; they are:

Get Your Start in Machine Learning

1. Sequence Learning Problem
2. Value Memorization
3. Echo Random Integer
4. Echo Random Subsequences
5. Sequence Classification

Properties of Problems

The sequence problems were designed with a few properties in mind:

- **Narrow.** To focus on one aspect of the sequence prediction, such as memory or function approximation.
- **Scalable.** To be made more or less difficult along the chosen narrow focus.
- **Reframed.** Two or more framings of the each problem are presented to support the exploration

I tried to provide a mixture of narrow focuses, problem difficulties, and required network architectures

If you have ideas for further extensions or similarly carefully designed problems, please let me know

Need help with LSTMs for Sequence Prediction?

Take my free 7-day email course and discover 6 different LSTM architectures

Click to sign-up and also get a free PDF Ebook version of the book

[Start Your FREE Mini-Course Now!](#)

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.

[START MY EMAIL COURSE](#)

1. Sequence Learning Problem

[Get Your Start in Machine Learning](#)

In this problem, a sequence of contiguous real values between 0.0 and 1.0 are generated. Given one or more time steps of past values, the model must predict the next item in the sequence.

We can generate this sequence directly, as follows:

```
1 from numpy import array
2
3 # generate a sequence of real values between 0 and 1.
4 def generate_sequence(length=10):
5     return array([i/float(length) for i in range(length)])
6
7 print(generate_sequence())
```

Running this example prints the generated sequence:

```
1 [ 0.  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9]
```

This could be framed as a memorization challenge where given the observation at the previous time

```
1 X (samples),    y
2 0.0,            0.1
3 0.1,            0.2
4 0.2,            0.3
5 ...
```

The network could memorize the input-output pairs, which is quite boring, but would demonstrate the

The problem could be framed as randomly chosen contiguous subsequences as input time steps and

```
1 X (timesteps),    y
2 0.4, 0.5, 0.6,    0.7
3 0.0, 0.2, 0.3,    0.4
4 0.3, 0.4, 0.5,    0.6
5 ...
```

This would require the network to learn either to add a fixed value to the last seen observation or to memorize all possible subsequences of the generated problem.

This framing of the problem would be modeled as a many-to-one sequence prediction problem.

This is an easy problem that tests primitive features of sequence learning. This problem could be so

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.

START MY EMAIL COURSE

Get Your Start in Machine Learning

2. Value Memorization

The problem is to remember the first value in the sequence and to repeat it at the end of the sequence.

This problem is based on “Experiment 2” used to demonstrate LSTMs in the 1997 paper [Long Short Term Memory](#).

This can be framed as a one-step prediction problem.

Given one value in the sequence, the model must predict the next value in the sequence. For example, given a value of “0” as an input, the model must predict the value “1”.

Consider the following two sequences of 5 integers:

```
1 3, 0, 1, 2, 3
2 4, 0, 1, 2, 4
```

The Python code will generate two sequences of arbitrary length. You could generalize it further if you

```
1 def generate_sequence(length=5):
2     return [i for i in range(length)]
3
4 # sequence 1
5 seq1 = generate_sequence()
6 seq1[0] = seq1[-1] = seq1[-2]
7 print(seq1)
8 # sequence 2
9 seq1 = generate_sequence()
10 seq1[0] = seq1[-1]
11 print(seq1)
```

Running the example generates and prints the above two sequences.

```
1 [3, 1, 2, 3, 3]
2 [4, 1, 2, 3, 4]
```

The integers could be normalized, or more preferably one hot encoded.

The patterns introduce a wrinkle in that there is conflicting information between the two sequences and that the model must know the context of each one-step prediction (e.g. the sequence it is currently predicting) in order to correctly predict each full

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.

START MY EMAIL COURSE

We can see that the first value of the sequence is repeated as the last value of the sequence. This is the indicator that provides context to the model as to which sequence it is working on.

The conflict is the transition from the second to last items in each sequence. In sequence one, a “2” is given as an input and a “3” must be predicted, whereas in sequence two, a “2” is given as input and a “4” must be predicted.

```

1 Sequence 1:
2
3 X (samples),    y
4 ...
5 1,              2
6 2,              3
7
8 Sequence 2:
9
10 X (samples),   y
11 ...
12 1,             2
13 2,             4

```

This wrinkle is important to prevent the model from memorizing each single-step input-output pair of model may be inclined to do.

This framing would be modeled as a one-to-one sequence prediction problem.

This is a problem that a multilayer Perceptron and other non-recurrent neural networks cannot learn. remembered across multiple samples.

This problem could be framed as providing the entire sequence except the last value as input time s

```

1 X (timesteps),    y
2 3, 0, 1, 2,      3
3 4, 0, 1, 2,      4

```

Each time step is still shown to the network one at a time, but the network must remember the value at the first time step. The difference is, the network can better learn the difference between the sequence, and between long sequences via backpropagation through time.

This framing of the problem would be modeled as a many-to-one sequence prediction problem.

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.

START MY EMAIL COURSE

Get Your Start in Machine Learning

Again, this problem could not be learned by a multilayer Perceptron.

3. Echo Random Integer

In this problem, random sequences of integers are generated. The model must remember an integer at a specific lag time and echo it at the end of the sequence.

For example, a random sequence of 10 integers may be:

```
1 5, 3, 2, 1, 9, 9, 2, 7, 1, 6
```

The problem may be framed as echoing the value at the 5th time step, in this case 9.

The code below will generate random sequences of integers.

```
1 from random import randint
2
3 # generate a sequence of random numbers in [0, 99]
4 def generate_sequence(length=10):
5     return [randint(0, 99) for _ in range(length)]
6
7 print(generate_sequence())
```

Running the example will generate and print a random sequence, such as:

```
1 [47, 69, 76, 9, 71, 87, 8, 16, 32, 81]
```

The integers can be normalized, but more preferably a one hot encoding can be used.

A simple framing of this problem is to echo the current input value.

```
1 yhat(t) = f(X(t))
```

For example:

```
1 X (timesteps),      y
2 5, 3, 2, 1, 9,      9
```

This trivial problem can easily be solved by a multilayer Perceptron and could be used for calibration

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.

START MY EMAIL COURSE

Get Your Start in Machine Learning

A more challenging framing of the problem is to echo the value at the previous time step.

```
1 yhat(t) = f(X(t-1))
```

For example:

```
1 X (timesteps),      y
2 5, 3, 2, 1, 9,      1
```

This is a problem that cannot be solved by a multilayer Perceptron.

The index to echo can be pushed further back in time, putting more demand on the LSTMs memory.

Unlike the “Value Memorization” problem above, a new sequence would be generated each training epoch. This would require that the model learn a generalization echo solution rather than memorize a specific sequence or sequences of random numbers.

In both cases, the problem would be modeled as a many-to-one sequence prediction problem.

4. Echo Random Subsequences

This problem also involves the generation of random sequences of integers.

Instead of echoing a single previous time step as in the previous problem, this problem requires the model to remember and output the whole input sequence.

The simplest framing would be the echo problem from the previous section. Instead, we will focus on a more challenging framing where the model is required to remember and output the whole input sequence.

For example:

```
1 X (timesteps),      y
2 5, 3, 2, 4, 1,      5, 3, 2, 4, 1
```

This could be modeled as a many-to-one sequence prediction problem where the output sequence is output directly at the end of the last value in the input sequence.

This can also be modeled as the network outputting one value for each input time step, e.g. a one-to-many sequence prediction problem.

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.

START MY EMAIL COURSE

for

Get Your Start in Machine Learning

A more challenging framing is to output a partial contiguous subsequence of the input sequence.

For example:

```
1 X (timesteps),      y
2 5, 3, 2, 4, 1,      5, 3, 2
```

This is more challenging because the number of inputs does not match the number of outputs. A many-to-many model of this problem would require a more advanced architecture such as the encoder-decoder LSTM.

Again, a one hot encoding would be preferred, although the problem could be modeled as normalized integer values.

5. Sequence Classification

The problem is defined as a sequence of random values between 0 and 1. This sequence is taken as input and the output is a binary label (0 or 1) one per timestep.

A binary label (0 or 1) is associated with each input. The output values are all 0. Once the cumulative sum of the input values reaches a threshold, then the output value flips from 0 to 1.

A threshold of 1/4 the sequence length is used.

For example, below is a sequence of 10 input timesteps (X):

```
1 0.63144003 0.29414551 0.91587952 0.95189228 0.32195638 0.60742236 0.83895793 0.18023048 0.00000000 0.00000000
```

The corresponding classification output (y) would be:

```
1 0 0 0 1 1 1 1 1 1 1
```

We can implement this in Python.

```
1 from random import random
2 from numpy import array
3 from numpy import cumsum
4
5 # create a sequence classification instance
6 def get_sequence(n_timesteps):
```

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.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

7     # create a sequence of random numbers in [0,1]
8     X = array([random() for _ in range(n_timesteps)])
9     # calculate cut-off value to change class values
10    limit = n_timesteps/4.0
11    # determine the class outcome for each item in cumulative sequence
12    y = array([0 if x < limit else 1 for x in cumsum(X)])
13    return X, y
14
15 X, y = get_sequence(10)
16 print(X)
17 print(y)

```

Running the example generates a random input sequence and calculates the corresponding output sequence of binary values.

```

1 [ 0.31102339  0.66591885  0.7211718  0.78159441  0.50496384  0.56941485
2  0.60775583  0.36833139  0.180908   0.80614878]
3 [0 0 0 0 1 1 1 1 1 1]

```

This is a sequence classification problem that can be modeled as one-to-one. State is required to input the entire input sequence to produce the output sequence flips from 0 to 1.

Further Reading

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

- [Long Short-Term Memory, 1997](#)
- [How to use Different Batch Sizes for Training and Predicting in Python with Keras](#)
- [Demonstration of Memory with a Long Short-Term Memory Network in Python](#)
- [How to Learn to Echo Random Integers with Long Short-Term Memory Recurrent Neural Networks](#)
- [How to use an Encoder-Decoder LSTM to Echo Sequences of Random Integers](#)
- [How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras](#)

Summary

In this tutorial, you discovered a suite of carefully designed contrived sequence prediction problems that you can use to explore the learning and memory capabilities of LSTM recurrent neural networks.

Specifically, you learned:

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.

START MY EMAIL COURSE

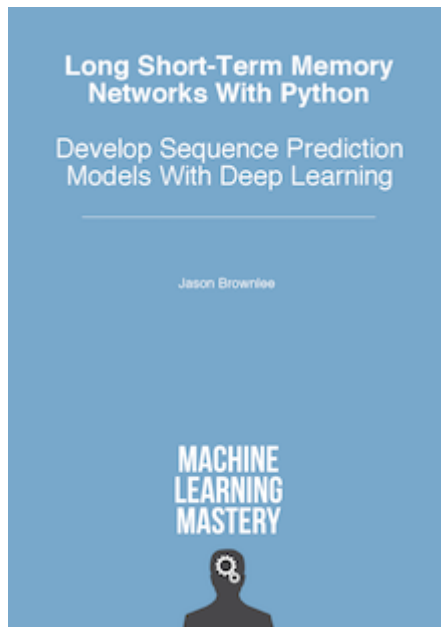
Get Your Start in Machine Learning

- Simple memorization tasks to test the learned memory capability of LSTMs.
- Simple echo tasks to test learned temporal dependence capability of LSTMs.
- Simple arithmetic tasks to test the interpretation capability of LSTMs.

Do you have any questions?

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

Develop LSTMs for Sequence Prediction Today!



Develop Your Own LSTM models in Minutes

...with just a few lines of python

Discover how in my new
[Long Short-Term Memory Networks](#)

It provides **self-study tutorial** on
CNN LSTMs, Encoder-Decoder LSTMs, generative models, data

Finally Bring LSTM Recurrent Your Sequence Prediction

Skip the Academics. Just

[Click to learn more](#)

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.

START MY EMAIL COURSE



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](#) →

< Gentle Introduction to Models for Sequence Prediction with Recurrent Neural Networks

A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size >

6 Responses to 5 Examples of Simple Sequence Prediction Problems for Learning LSTM Recurrent Neural Networks



Shantam August 1, 2017 at 1:44 am #

Thanks a lot for your valuable insights on LSTMs. Your blogs have been a great learning platform.

I have been lately trying different architectures using LSTMs for time series forecasting.

In keras the default shape of input tensor is given by [batch size, timesteps, features].

do you think the lags/percent changes over lags should be passed as features or as timesteps while res

e.g : Assuming we are using lags/percent changes over lags as features for the past 4 days(assuming n

for 5 inputs and 1 output

should it be modeled as [1,5] or [5,1].

the input shape of the tensor should be [batch size, 1, 5(current day's value :past 4 day's value)] or [batch size, 5, 1].

At a higher level, do you think LSTMs can learn better across time steps or over across features within the time-steps.

Thanks

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.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Jason Brownlee August 1, 2017 at 8:02 am #

REPLY ↩

Great question. A good general answer is to brainstorm and then try/bake-off everything in terms of model skill.

In terms of normative use, I would encourage you to treat one sample as one series of many time steps, with one or more features at each time step.

Also, I have found LSTMs to be not super great at autoregression tasks. Please baseline performance with a well tuned MLP.

Does that help?



Shantam August 3, 2017 at 7:47 am #

REPLY ↩

Thanks for your feedback. I have been testing multiple architecture settings(still working). I wish

LSTMs and other time series methods(ARIMA,ETS, HW) are working great for a many to one input output
long term forecasts.

I am eagerly waiting for your blog on Bayesian hyper-parameter optimization.

(P.S. I have more than a 1000 nets to optimize)

Thanks a lot ! 😊

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.

START MY EMAIL COURSE



Jason Brownlee August 4, 2017 at 6:43 am #

Wow, that is a lot of nets!

Error accumulates over longer forecast periods. The problem is hard.



Forw September 18, 2017 at 7:11 pm #

REPLY ↩

I have to implement an ANN algorithm that produces an estimate of unknown state variables using their representation. Basically, it should be divided
in 2 steps: in the Prediction one an estimate of the current state is created based on the state in the prev

Get Your Start in Machine Learning

information from the current time step is used to produce a new accurate estimate of the state. So, if I consider my system as a black-box what I have is.

Input U = Control vector, that indicates the magnitude of the control system's on the situation

Input Z = Measurement vector, it contains the real-world measurement we received in a time step

Output X_n = Newest estimate of the current "true" state

Output P_n = Newest estimate of the average error for each part of the state The algorithm that must be implemented must rely on a RNN architecture, more specific an LSTM. What I am struggling is how to correlate these inputs/outputs that I have to the one of an LSTM structure. How can I feed the network making in compliant to my state estimation problem?

Thanks for any kind of help!



Jason Brownlee September 19, 2017 at 7:35 am #

Perhaps this post will help you prepare your data:

<https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/>

Leave a Reply

Name (required)

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.

START MY EMAIL COURSE

Get Your Start in Machine Learning

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.

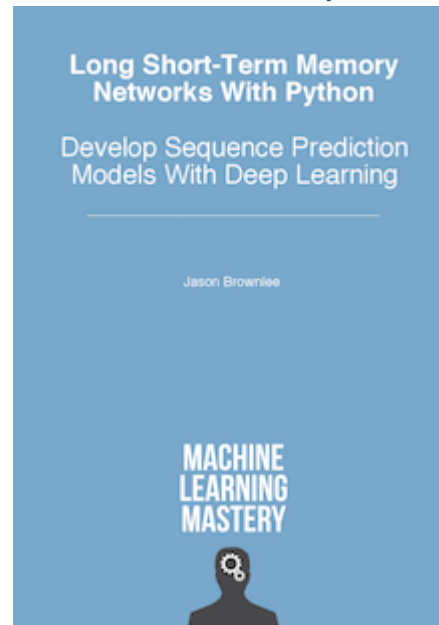
Get Your Start in Machine Learning



You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)[Get Your Start in Machine Learning](#)

[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

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.

[START MY EMAIL COURSE](#)[Get Your Start in Machine Learning](#)



Time Series Forecasting with the Long Short-Term Memory Network in Python

APRIL 7, 2017



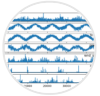
Multi-Class Classification Tutorial with the Keras Deep Learning Library

JUNE 2, 2016



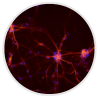
Regression Tutorial with the Keras Deep Learning Library in Python

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.

START MY EMAIL COURSE

© 2017 Machine Learning Mastery. All Rights Reserved.

[Privacy](#) | [Contact](#) | [About](#)

Get Your Start in Machine Learning