Long Short-Term Memory Networks With Python

7-Day Mini-Course

Jason Brownlee

MACHINE LEARNING MASTERY

Disclaimer

Long Short-Term Memory Networks With Python Mini-Course

Edition: v1.6

Find the latest version of this guide online at: http://MachineLearningMastery.com

Contents

Before We Get Started...

The Long Short-Term Memory network, or LSTM for short, is a type of recurrent neural network that achieves state-of-the-art results on challenging prediction problems. In this mini-course, you will discover how you can develop a suite of different types of LSTM models in 7 days. Let's get started.

Who Is This Mini-Course For?

Before we get started, let's make sure you are in the right place. The list below provides some general guidelines as to who this course was designed for. Don't panic if you don't match these points exactly, you might just need to brush up in one area or another to keep up.

- **You're a Developer**: This is a course for developers. You are a developer of some sort. You know how to read and write code. You know how to develop and debug a program.
- **You know Python**: This is a course for Python people. You know the Python programming language, or you're a skilled enough developer that you can pick it up as you go along.
- **You know some Machine Learning**: This is a course for novice machine learning practitioners. You know some basic practical machine learning, or you can figure it out quickly.

This mini-course is neither a textbook on Python or a textbook on LSTMs. It will take you from a developer that knows a little machine learning to a developer who can bring LSTMs to your sequence prediction project.

Note: This mini-course assumes you have a working Python 2 or 3 SciPy environment with at least NumPy, Pandas, scikit-learn and Keras 2 installed with either the Theano or TensorFlow backend. If you need help with your environment, see the post:

- *How to Setup a Python Environment for Machine Learning and Deep Learning.* https://goo.gl/QwffqZ

Mini-Course Overview (what to expect)

This mini-course is broken down into 7 lessons. You could complete one lesson per day (recommended) or complete all of the lessons in one day (hardcore). It really depends on the time you have available and your level of enthusiasm.

Below are a list of the 7 lessons that will get you started and productive with LSTMs in Python:

- **Lesson 01**: Long Short-Term Memory Networks.
- **Lesson 02**: Develop Vanilla LSTMs.
- **Lesson 03**: Develop Stacked LSTMs.
- **Lesson 04**: Develop CNN LSTMs.
- **Lesson 05**: Develop Encoder-Decoder LSTMs.
- **Lesson 06**: Develop Bidirectional LSTMs.
- **Lesson 07**: Develop Generative LSTMs.

Each lesson could take you about 30-to-60 minutes. Take your time and complete the lessons at your own pace. The lessons expect you to go off and find out how to do things. I will give you hints, but part of the point of each lesson is to force you to learn where to go to look for help on and about LSTMs.

Here's a tip: All of the answers to these lessons can be found on my blog http://MachineLearningMastery.com. Use the search feature.

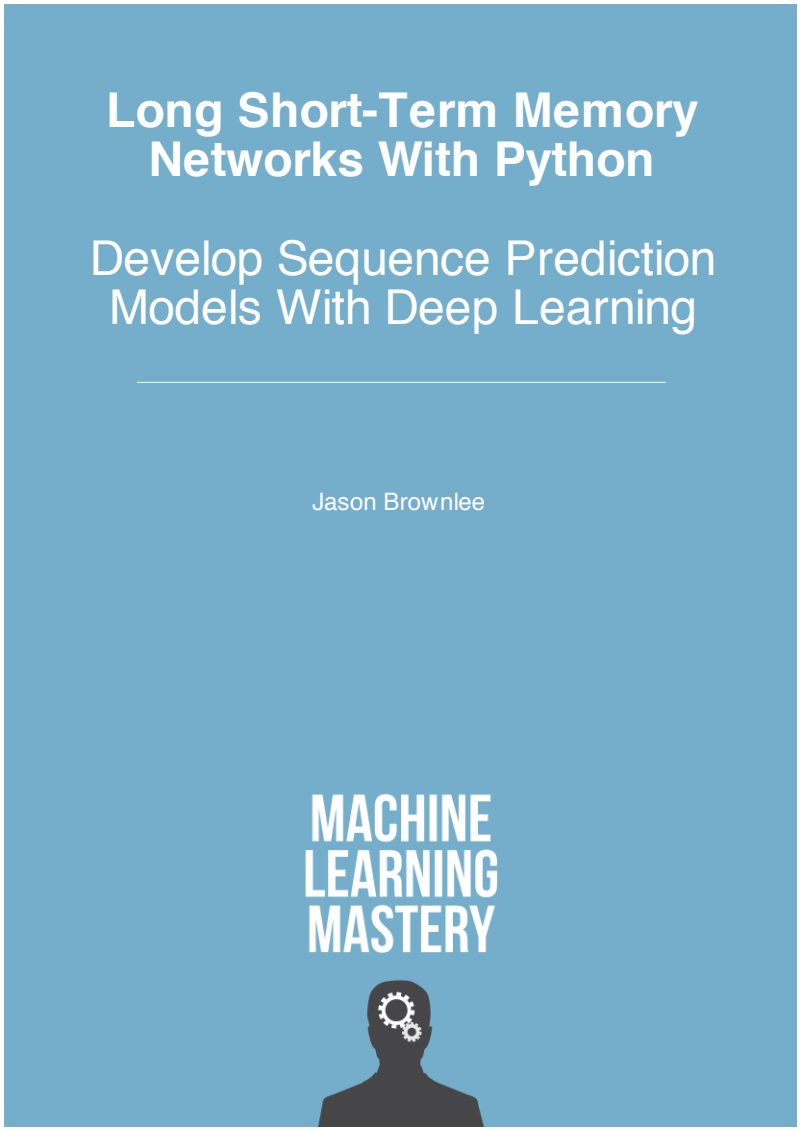
Post your solutions to the exercises and experiments on line and send me the link. I'd love to see what you come up with!

Hang in there, don't give up!

Take the Next Step

Looking for more help with Long Short-Term Memory networks?

Grab my new book:
Long Short-Term Memory Networks With Python
https://machinelearningmastery.com/lstms-with-python/



Long Short-Term Memory Networks

Goal

The goal of this lesson is to understand LSTMs from a high-level sufficiently so that you can explain what they are and how they work to a colleague or manager.

Long Short-Term Memory Networks

Sequence prediction is different to other types of supervised learning problems. The sequence imposes an order on the observations that must be preserved when training models and making predictions.

The Long Short-Term Memory, or LSTM, network is a type of Recurrent Neural Network (RNN) designed for sequence problems. Given a standard feedforward Multilayer Perceptron network, an RNN can be thought of as the addition of loops to the architecture. The recurrent connections add state or memory to the network and allow it to learn and harness the ordered nature of observations within input sequences. The internal memory means outputs of the network are conditional on the recent context in the input sequence, not what has just been presented as input to the network. In a sense, this capability unlocks sequence prediction for neural networks and deep learning.

Exercises

This section lists questions for you to research and answer.

- What is sequence prediction and what are some general examples?
- What are the limitations of traditional neural networks for sequence prediction?
- What is the promise of RNNs for sequence prediction?
- What is the LSTM and what are its constituent parts?
- What are some prominent applications of LSTMs?

Develop Vanilla LSTMs

Goal

The goal of this lesson is to learn how to develop and evaluate vanilla LSTM models.

Vanilla LSTM

A simple LSTM configuration is the Vanilla LSTM. It is named Vanilla in this course to differentiate it from deeper LSTMs and the suite of more elaborate configurations. It is the LSTM architecture defined in the original 1997 LSTM paper[1] and the architecture that will give good results on most small sequence prediction problems. The Vanilla LSTM is defined as:

1. Input layer.
2. Fully connected LSTM hidden layer.
3. Fully connected output layer.

In Keras, a Vanilla LSTM is defined below, with ellipsis for the specific configuration of the number of neurons in each layer.

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

Listing 1: Example of defining a Vanilla LSTM model.

Exercises

- What is the vanilla LSTM architecture?
- What are some examples where the vanilla LSTM has been applied?

Experiment

Design and execute an experiment that demonstrates a vanilla LSTM on a sequence prediction problem.

[1] http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

Develop Stacked LSTMs

Goal

The goal of this lesson is to learn how to develop and evaluate stacked LSTM models.

Stacked LSTMs

The Stacked LSTM is a model that has multiple hidden LSTM layers where each layer contains multiple memory cells. Given that LSTMs operate on sequence data, it means that the addition of layers adds levels of abstraction of input observations over time. In effect, chunking observations over time or representing the problem at different time scales.

In Keras, LSTM layers can be stacked where the prior layer must return a sequence of outputs (3D output) rather than a single output at the end of the sequence (2D output). This can be done by setting the `return_sequences` argument to `True`.

```
model = Sequential()
model.add(LSTM(..., return_sequences=True, input_shape=(...)))
model.add(LSTM(...))
model.add(Dense(...))
```

Listing 2: Example of defining a Stacked LSTM with 2 hidden layers.

Exercises

- What are the difficulties in using a vanilla LSTM on a sequence problem with hierarchical structure?
- What are stacked LSTMs?
- What are some examples of where the stacked LSTM has been applied?
- What benefits do stacked LSTMs provide?
- How can a stacked LSTM be implemented in Keras?

Experiment

Design and execute an experiment that demonstrates a stacked LSTM on a sequence prediction problem with hierarchical input structure.

Develop CNN LSTMs

Goal

The goal of this lesson is to learn how to develop LSTM models that use a Convolutional Neural Network on the front end.

CNN LSTMs

The CNN LSTM architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction. This architecture is used for the task of generating textual descriptions of images.

We can define a CNN LSTM model to be trained jointly in Keras. A CNN LSTM can be defined by adding CNN layers on the front end followed by `LSTM` layers with a `Dense` layer on the output. The same input features are used for each step in the output sequence. This requires that the input CNN model be wrapped in a `TimeDistributed` layer.

```
model = Sequential()
model.add(TimeDistributed(Conv2D(...))
model.add(TimeDistributed(MaxPooling2D(...)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(...))
model.add(Dense(...))
```

Listing 3: Example of a CNN LSTM model.

Exercises

- What are the difficulties of using a vanilla LSTM with spatial input data?
- What is the CNN LSTM architecture?
- What benefits does the CNN LSTM provide?
- How can the CNN LSTM architecture be implemented in Keras?

Experiment

Design and execute an experiment that demonstrates a CNN LSTM on a sequence prediction problem with spatial input.

Develop Encoder-Decoder LSTMs

Goal

The goal of this lesson is to learn how to develop encoder-decoder LSTM models.

Encoder-Decoder LSTMs

Sequence-to-sequence prediction problems (seq2seq) are a more challenging type of problem that takes a sequence as input and require a sequence prediction as output. The Encoder-Decoder LSTM architecture was designed for seq2seq and is comprised of two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence.

The Encoder-Decoder architecture can be implemented in Keras using one or more LSTM layers for the encoder, repeating the output of the encoder once for each time step in the output sequence with a `RepeatVector` layer, using one or more `LSTM` layers as the decoder, then using the same `Dense` layer to output each step in the output sequence.

```
model = Sequential()
model.add(LSTM(..., input_shape=(...)))
model.add(RepeatVector(...))
model.add(LSTM(..., return_sequences=True))
model.add(TimeDistributed(Dense(...)))
```

Listing 4: Example of an Encoder-Decoder model.

Exercises

- What are sequence-to-sequence (seq2seq) prediction problems?
- What are the difficulties of using a vanilla LSTM on seq2seq problems?
- What is the encoder-decoder LSTM architecture?
- What are the benefits of encoder-decoder LSTMs?

Experiment

Design and execute an experiment that demonstrates an encoder-decoder LSTM on a sequence-to-sequence prediction problem.

Develop Bidirectional LSTMs

Goal

The goal of this lesson is to learn how to develop Bidirectional LSTM models.

Bidirectional LSTMs

Bidirectional LSTMs focus on the problem of getting the most out of the input sequence by stepping through input time steps in both the forward and backward directions. In practice, this architecture involves duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second.

Bidirectional LSTMs are supported in Keras via the `Bidirectional` layer wrapper that essentially merges the output from two parallel LSTMs, one with input processed forward and one with output processed backwards. This wrapper takes a recurrent layer (e.g. the first hidden `LSTM` layer) as an argument.

```
model = Sequential()
model.add(Bidirectional(LSTM(...), input_shape=(...)))
...
```

Listing 5: Example of a `Bidirectional` wrapped `LSTM` layer.

Exercises

- What is a bidirectional LSTM?
- What are some examples where bidirectional LSTMs have been used?
- What benefit does a bidirectional LSTM offer over a vanilla LSTM?
- What concerns regarding time steps does a bidirectional architecture raise?
- How can bidirectional LSTMs be implemented in Keras?

Experiment

Design and execute an experiment that compares forward, backward, and bidirectional LSTM models on a sequence prediction problem.

Develop Generative LSTMs

Goal

The goal of this lesson is to learn how to develop LSTMs for use in generative models.

Generative LSTMs

LSTMs can be used as a generative model. Given a large corpus of sequence data, such as text documents, LSTM models can be designed to learn the general structural properties of the corpus, and when given a seed input, can generate new sequences that are representative of the original sequences.

A Generative LSTM is not really architecture, it is more a change in perspective about what an LSTM predictive model learns and how the model is used. We could conceivably use any LSTM architecture as a generative model, such as a Vanilla LSTM.

Key to the development of generative LSTMs is the choice of representation for sequences. For example, if text data is used, then one could choose to model words or characters. Words provide a high-level of abstraction but the encoding would have to be updated for each new word. Characters are more flexible, but may require a larger model and more training.

Exercises

- What are generative models?
- How can LSTMs be used as generative models?
- What are some examples of LSTMs as generative models?
- What benefits do LSTMs have as generative models?

Experiment

Design and execute an experiment to learn a corpus of text and generate new samples of text with the same syntax, grammar, and style.

Final Word Before You Go...

You made it. Well done! Take a moment and look back at how far you have come. You discovered:

- What LSTM networks are and how they deliver on the promise if recurrent neural networks.
- How to develop Vanilla LSTMs for sequence prediction.
- How to develop Stacked LSTMs for sequence prediction with multiple levels of abstraction.
- How to develop CNN LSTMs for sequence prediction with image input.
- How to develop Encoder-Decoder LSTMs for sequence-to-sequence prediction.
- How to develop Bidirectional LSTMs for improving the performance of sequence prediction models.
- How to develop Generative LSTMs for inventing new sequences given a corpus of examples.

Don't make light of this, you have come a long way in a short amount of time. This is just the beginning of your Long Short-Term Memory network journey with Python. Keep practicing and developing your skills.

How Did You Go With The Mini-Course?

Did you enjoy this mini-course?
Do you have any questions or sticking points?

Let me know, send me an email at: **jason@MachineLearningMastery.com**

Take the Next Step

Looking for more help with Long Short-Term Memory networks?

Grab my new book:
Long Short-Term Memory Networks With Python
https://machinelearningmastery.com/lstms-with-python/