# Chapter 1

# Promise of Deep Learning for Time Series Forecasting

Deep learning neural networks are able to automatically learn arbitrary complex mappings from inputs to outputs and support multiple inputs and outputs. These are powerful features that offer a lot of promise for time series forecasting, particularly on problems with complex-nonlinear dependencies, multivalent inputs, and multi-step forecasting. These features along with the capabilities of more modern neural networks may offer great promise such as the automatic feature learning provided by convolutional neural networks and the native support for sequence data in recurrent neural networks. In this tutorial, you will discover the promised capabilities of deep learning neural networks for time series forecasting. After reading this tutorial, you will know:

- The focus and implicit, if not explicit, limitations on classical time series forecasting methods.
- The general capabilities of Multilayer Perceptrons and how they may be harnessed for time series forecasting.
- The added capabilities of feature learning and native support for sequences provided by Convolutional Neural Networks and Recurrent Neural Networks.

Let's get started.

## 1.1 Time Series Forecasting

Time series forecasting is difficult. Unlike the simpler problems of classification and regression, time series problems add the complexity of order or temporal dependence between observations. This can be difficult as specialized handling of the data is required when fitting and evaluating models. This temporal structure can also aid in modeling, providing additional structure like trends and seasonality that can be leveraged to improve model skill. Traditionally, time series forecasting has been dominated by linear methods like ARIMA because they are well understood and effective on many problems. But these classical methods also suffer from some limitations, such as:

- **Focus on complete data**: missing or corrupt data is generally unsupported.

- **Focus on linear relationships**: assuming a linear relationship excludes more complex joint distributions.
- **Focus on fixed temporal dependence**: the relationship between observations at different times, and in turn the number of lag observations provided as input, must be diagnosed and specified.
- **Focus on univariate data**: many real-world problems have multiple input variables.
- **Focus on one-step forecasts**: many real-world problems require forecasts with a long time horizon.

Machine learning methods can be effective on more complex time series forecasting problems with multiple input variables, complex nonlinear relationships, and missing data. In order to perform well, these methods often require hand-engineered features prepared by either domain experts or practitioners with a background in signal processing.

> Existing techniques often depended on hand-crafted features that were expensive to create and required expert knowledge of the field.
>
> — *Deep Learning for Time-Series Analysis*, 2017.

## 1.2 Multilayer Perceptrons for Time Series

Simpler neural networks such as the Multilayer Perceptron or MLP approximate a mapping function from input variables to output variables. This general capability is valuable for time series for a number of reasons.

- **Robust to Noise**. Neural networks are robust to noise in input data and in the mapping function and can even support learning and prediction in the presence of missing values.
- **Nonlinear**. Neural networks do not make strong assumptions about the mapping function and readily learn linear and nonlinear relationships.

> ... one important contribution of neural networks - namely their elegant ability to approximate arbitrary nonlinear functions. This property is of high value in time series processing and promises more powerful applications, especially in the subfield of forecasting ...
>
> — *Neural Networks for Time Series Processing*, 1996.

More specifically, neural networks can be configured to support an arbitrary defined but fixed number of inputs and outputs in the mapping function. This means that neural networks can directly support:

- **Multivariate Inputs**. An arbitrary number of input features can be specified, providing direct support for multivariate forecasting.

- **Multi-step Forecasts**. An arbitrary number of output values can be specified, providing direct support for multi-step and even multivariate forecasting.

For these capabilities alone, feedforward neural networks may be useful for time series forecasting. Implicit in the usage of neural networks is the requirement that there is indeed a meaningful mapping from inputs to outputs to learn. Modeling a mapping of a random walk will perform no better than a persistence model (e.g. using the last seen observation as the forecast). This expectation of a learnable mapping function also makes one of the limitations clear: the mapping function is fixed or static.

- **Fixed Inputs**. The number of lag input variables is fixed, in the same way as traditional time series forecasting methods.
- **Fixed Outputs**. The number of output variables is also fixed; although a more subtle issue, it means that for each input pattern, one output must be produced.

> Sequences pose a challenge for [deep neural networks] because they require that the dimensionality of the inputs and outputs is known and fixed.
>
> — *Sequence to Sequence Learning with Neural Networks*, 2014.

Feedforward neural networks do offer great capability but still suffer from this key limitation of having to specify the temporal dependence upfront in the design of the model. This dependence is almost always unknown and must be discovered and teased out from detailed analysis in a fixed form.

## 1.3 Convolutional Neural Networks for Time Series

Convolutional Neural Networks or CNNs are a type of neural network that was designed to efficiently handle image data. They have proven effective on challenging computer vision problems both achieving state-of-the-art results on tasks like image classification and providing a component in hybrid models for entirely new problems such as object localization, image captioning and more.

They achieve this by operating directly on raw data, such as raw pixel values, instead of domain-specific or handcrafted features derived from the raw data. The model then learns how to automatically extract the features from the raw data that are directly useful for the problem being addressed. This is called representation learning and the CNN achieves this in such a way that the features are extracted regardless of how they occur in the data, so-called transform or distortion invariance.

> Convolutional networks combine three architectural ideas to ensure some degree of shift and distortion invariance: local receptive fields, shared weights (or weight replication), and, sometimes, spatial or temporal subsampling.
>
> — *Convolutional Networks for Images, Speech, and Time-Series*, 1998.

The ability of CNNs to learn and automatically extract features from raw input data can be applied to time series forecasting problems. A sequence of observations can be treated like a one-dimensional image that a CNN model can read and distill into the most salient elements.

> The key attribute of the CNN is conducting different processing units [...] Such a variety of processing units can yield an effective representation of local salience of the signals. Then, the deep architecture allows multiple layers of these processing units to be stacked, so that this deep learning model can characterize the salience of signals in different scales.

— *Deep Convolutional Neural Networks On Multichannel Time Series For Human Activity Recognition*, 2015.

This capability of CNNs has been demonstrated to great effect on time series classification tasks such as automatically detecting human activities based on raw accelerator sensor data from fitness devices and smartphones.

> The key advantages of [CNNs for activity recognition] are: i) feature extraction is performed in task dependent and non hand-crafted manners; ii) extracted features have more discriminative power w.r.t. the classes of human activities; iii) feature extraction and classification are unified in one model so their performances are mutually enhanced.

— *Deep Convolutional Neural Networks On Multichannel Time Series For Human Activity Recognition*, 2015.

CNNs get the benefits of Multilayer Perceptrons for time series forecasting, namely support for multivariate input, multivariate output and learning arbitrary but complex functional relationships, but do not require that the model learn directly from lag observations. Instead, the model can learn a representation from a large input sequence that is most relevant for the prediction problem.

- **Feature Learning**. Automatic identification, extraction and distillation of salient features from raw input data that pertain directly to the prediction problem that is being modeled.

## 1.4 Recurrent Neural Networks for Time Series

Recurrent neural networks like the Long Short-Term Memory network or LSTM add the explicit handling of order between observations when learning a mapping function from inputs to outputs, not offered by MLPs or CNNs. They are a type of neural network that adds native support for input data comprised of sequences of observations.

- **Native Support for Sequences**. Recurrent neural networks directly add support for input sequence data.

The addition of sequence is a new dimension to the function being approximated. Instead of mapping inputs to outputs alone, the network is capable of learning a mapping function for the inputs over time to an output.

> Long Short-Term Memory (LSTM) is able to solve many time series tasks unsolvable by feedforward networks using fixed size time windows.

— *Applying LSTM to Time Series Predictable through Time-Window Approaches*, 2001.

This capability of LSTMs has been used to great effect in complex natural language processing problems such as neural machine translation where the model must learn the complex interrelationships between words both within a given language and across languages in translating form one language to another. This capability can be used in time series forecasting. In addition to the general benefits of using neural networks for time series forecasting, recurrent neural networks can also automatically learn the temporal dependence from the data.

- **Learned Temporal Dependence**. The most relevant context of input observations to the expected output is learned and can change dynamically.

In the simplest case, the network is shown one observation at a time from a sequence and can learn what observations it has seen previously are relevant and how they are relevant to forecasting. The model both learns a mapping from inputs to outputs and learns what context from the input sequence is useful for the mapping, and can dynamically change this context as needed.

> Because of this ability to learn long term correlations in a sequence, LSTM networks obviate the need for a pre-specified time window and are capable of accurately modelling complex multivariate sequences.
>
> — *Long Short Term Memory Networks for Anomaly Detection in Time Series*, 2015.

## 1.5 Promise of Deep Learning

The capabilities of deep learning neural networks suggest a good fit for time series forecasting. By definition and with enough resources, neural networks in general should be able to subsume the capabilities of classical linear forecasting methods given their ability to learn arbitrary complex mapping from inputs to outputs.

- Neural networks learn arbitrary mapping functions.

It is good practice to manually identify and remove systematic structures from time series data to make the problem easier to model (e.g. make the series stationary), and this may still be a best practice when using recurrent neural networks. But, the general capability of these networks suggests that this may not be a requirement for a skillful model. Technically, the available context of the sequence provided as input may allow neural network models to learn both trend and seasonality directly.

- Neural networks may not require a scaled or stationary time series as input

Each of the three classes of neural network models discussed, MLPs, CNNs and RNNs offer capabilities that are challenging for classical time series forecasting methods, namely:

- Neural networks support multivariate inputs.
- Neural networks support multi-step outputs.

Although MLPs can operate directly on raw observations, CNNs offer efficiency and much greater performance at automatically learning to identify, extract and distill useful features from raw data.

- Convolutional neural networks support efficient feature learning.

Although MLPs and CNNs can learn arbitrary mapping functions, the explicit addition of support for input sequences in RNNs offers efficiency and greater performance for automatically learning the temporal dependencies both within the input sequence and from the input sequence to the output.

- LSTM networks support efficient learning of temporal dependencies.

These capabilities can also be combined, such as in the use of hybrid models like CNN-LSTMs and ConvLSTMs that seek to harness the capabilities of all three model types.

- Hybrid models efficiently combine the diverse capabilities of different architectures.

Traditionally, a lot of research has been invested into using MLPs for time series forecasting with modest results (covered in Chapter 10). Perhaps the most promising area in the application of deep learning methods to time series forecasting are in the use of CNNs, LSTMs and hybrid models. These areas will be our primary focus throughput this book.

## 1.6 Extensions

This section lists some ideas for extending the tutorial that you may wish to explore.

- **Your Expectations**. List three of your own expectations of deep learning methods for time series forecasting, perhaps expectations you had before reading this tutorial.
- **Known Limitations**. Research and list or quote three limitations of deep learning methods for time series forecasting listed in the literature.
- **Successful Example**. Find one research paper that presents a successful application of deep learning methods for time series forecasting.

If you explore any of these extensions, I'd love to know.

## 1.7 Further Reading

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

- *Deep Learning for Time-Series Analysis*, 2017.
  https://arxiv.org/abs/1701.01887
- *Neural Networks for Time Series Processing*, 1996.
  http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.5697

- *Sequence to Sequence Learning with Neural Networks*, 2014.
  https://arxiv.org/abs/1409.3215
- *Applying LSTM to Time Series Predictable through Time-Window Approaches*, 2001.
  https://link.springer.com/chapter/10.1007/3-540-44668-0_93
- *Long Short Term Memory Networks for Anomaly Detection in Time Series*, 2015.
  https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf
- *Convolutional Networks for Images, Speech, and Time-Series*, 1998.
  https://dl.acm.org/citation.cfm?id=303704
- *Deep Convolutional Neural Networks On Multichannel Time Series For Human Activity Recognition*, 2015.
  https://dl.acm.org/citation.cfm?id=2832806


## 1.8 Summary

In this tutorial, you discovered the promised capabilities of deep learning neural networks for time series forecasting. Specifically, you learned:

- The focus and implicit, if not explicit, limitations on classical time series forecasting methods.
- The general capabilities of Multilayer Perceptrons and how they may be harnessed for time series forecasting.
- The added capabilities of feature learning and native support for sequences provided by Convolutional Neural Networks and Recurrent Neural Networks.

### 1.8.1 Next

In the next lesson, you will discover a taxonomy that you can use to quickly learn a lot about your time series forecasting problem.

# Chapter 2

# Taxonomy of Time Series Forecasting Problems

When you are presented with a new time series forecasting problem, there are many things to consider. The choice that you make directly impacts each step of the project from the design of a test harness to evaluate forecast models to the fundamental difficulty of the forecast problem that you are working on. It is possible to very quickly narrow down the options by working through a series of questions about your time series forecasting problem. By considering a few themes and questions within each theme, you narrow down the type of problem, test harness, and even choice of algorithms for your project. In this tutorial, you will discover a framework that you can use to quickly understand and frame your time series forecasting problem. After reading this tutorial, you will know:

- A structured way of thinking about time series forecasting problems.
- A framework to uncover the characteristics of a given time series forecasting problem.
- A suite of specific questions, the answers to which will help to define your forecasting problem.

Let's get started.

## 2.1 Framework Overview

Time series forecasting involves developing and using a predictive model on data where there is an ordered relationship between observations. Before you get started on your project, you can answer a few questions and greatly improve your understanding of the structure of your forecast problem, the structure of the model requires, and how to evaluate it. The framework presented in this tutorial is divided into seven parts; they are:

1. Inputs vs. Outputs.
2. Endogenous vs. Exogenous.
3. Unstructured vs. Structured.

4. Regression vs. Classification.
5. Univariate vs. Multivariate.
6. Single-step vs. Multi-step.
7. Static vs. Dynamic.
8. Contiguous vs. Discontiguous.

I recommend working through this framework before starting any time series forecasting project. Your answers may not be crisp on the first time through and the questions may require to you study the data, the domain, and talk to experts and stakeholders. Update your answers as you learn more as it will help to keep you on track, avoid distractions, and develop the actual model that you need for your project.

## 2.2 Inputs vs. Outputs

Generally, a prediction problem involves using past observations to predict or forecast one or more possible future observations. The goal is to guess about what might happen in the future. When you are required to make a forecast, it is critical to think about the data that you will have available to make the forecast and what you will be guessing about the future. We can summarize this as what are the inputs and outputs of the model when making a single forecast.

- **Inputs**: Historical data provided to the model in order to make a single forecast.
- **Outputs**: Prediction or forecast for a future time step beyond the data provided as input.

The input data is not the data used to train the model. We are not at that point yet. It is the data used to make one forecast, for example the last seven days of sales data to forecast the next one day of sales data. Defining the inputs and outputs of the model forces you to think about what exactly is or may be required to make a forecast. You may not be able to be specific when it comes to input data. For example, you may not know whether one or multiple prior time steps are required to make a forecast. But you will be able to identify the variables that could be used to make a forecast.

***What are the inputs and outputs for a forecast?***

## 2.3 Endogenous vs. Exogenous

The input data can be further subdivided in order to better understand its relationship to the output variable. An input variable is endogenous if it is affected by other variables in the system and the output variable depends on it. In a time series, the observations for an input variable depend upon one another. For example, the observation at time $t$ is dependent upon the observation at $t-1$; $t-1$ may depend on $t-2$, and so on. An input variable is an exogenous variable if it is independent of other variables in the system and the output variable depends upon it. Put simply, endogenous variables are influenced by other variables in the system (including themselves) whereas as exogenous variables are not and are considered as outside the system.

- **Endogenous**: Input variables that are influenced by other variables in the system and on which the output variable depends.
- **Exogenous**: Input variables that are not influenced by other variables in the system and on which the output variable depends.

Typically, a time series forecasting problem has endogenous variables (e.g. the output is a function of some number of prior time steps) and may or may not have exogenous variables. Often, exogenous variables are ignored given the strong focus on the time series. Explicitly thinking about both variable types may help to identify easily overlooked exogenous data or even engineered features that may improve the model.

***What are the endogenous and exogenous variables?***

## 2.4 Regression vs. Classification

Regression predictive modeling problems are those where a quantity is predicted. A quantity is a numerical value; for example a price, a count, a volume, and so on. A time series forecasting problem in which you want to predict one or more future numerical values is a regression type predictive modeling problem. Classification predictive modeling problems are those where a category is predicted. A category is a label from a small well-defined set of labels; for example `hot`, `cold`, `up`, `down`, and `buy`, `sell` are categories. A time series forecasting problem in which you want to classify input time series data is a classification type predictive modeling problem.

- **Regression**: Forecast a numerical quantity.
- **Classification**: Classify as one of two or more labels.

***Are you working on a regression or classification predictive modeling problem?***

There is some flexibility between these types. For example, a regression problem can be reframed as classification and a classification problem can be reframed as regression. Some problems, like predicting an ordinal value, can be framed as either classification and regression. It is possible that a reframing of your time series forecasting problem may simplify it.

***What are some alternate ways to frame your time series forecasting problem?***

## 2.5 Unstructured vs. Structured

It is useful to plot each variable in a time series and inspect the plot looking for possible patterns. A time series for a single variable may not have any obvious pattern. We can think of a series with no pattern as unstructured, as in there is no discernible time-dependent structure. Alternately, a time series may have obvious patterns, such as a trend or seasonal cycles as structured. We can often simplify the modeling process by identifying and removing the obvious structures from the data, such as an increasing trend or repeating cycle. Some classical methods even allow you to specify parameters to handle these systematic structures directly.

- **Unstructured**: No obvious systematic time-dependent pattern in a time series variable.
- **Structured**: Systematic time-dependent patterns in a time series variable (e.g. trend and/or seasonality).

***Are the time series variables unstructured or structured?***

## 2.6 Univariate vs. Multivariate

A single variable measured over time is referred to as a univariate time series. Univariate means one variate or one variable. Multiple variables measured over time is referred to as a multivariate time series: multiple variates or multiple variables.

- **Univariate**: One variable measured over time.
- **Multivariate**: Multiple variables measured over time.

***Are you working on a univariate or multivariate time series problem?***

Considering this question with regard to inputs and outputs may add a further distinction. The number of variables may differ between the inputs and outputs, e.g. the data may not be symmetrical. For example, you may have multiple variables as input to the model and only be interested in predicting one of the variables as output. In this case, there is an assumption in the model that the multiple input variables aid and are required in predicting the single output variable.

- **Univariate and Multivariate Inputs**: One or multiple input variables measured over time.
- **Univariate and Multivariate Outputs**: One or multiple output variables to be predicted.

## 2.7 Single-step vs. Multi-step

A forecast problem that requires a prediction of the next time step is called a one-step forecast model. Whereas a forecast problem that requires a prediction of more than one time step is called a multi-step forecast model. The more time steps to be projected into the future, the more challenging the problem given the compounding nature of the uncertainty on each forecasted time step.

- **One-step**: Forecast the next time step.
- **Multi-step**: Forecast more than one future time steps.

***Do you require a single-step or a multi-step forecast?***

## 2.8 Static vs. Dynamic

It is possible to develop a model once and use it repeatedly to make predictions. Given that the model is not updated or changed between forecasts, we can think of this model as being static. Conversely, we may receive new observations prior to making a subsequent forecast that could be used to create a new model or update the existing model. We can think of developing a new or updated model prior to each forecasts as a dynamic problem.

For example, it if the problem requires a forecast at the beginning of the week for the week ahead, we may receive the true observation at the end of the week that we can use to update the model prior to making next weeks forecast. This would be a dynamic model. If we do not get a true observation at the end of the week or we do and choose to not re-fit the model, this would be a static model. We may prefer a dynamic model, but the constraints of the domain or limitations of a chosen algorithm may impose constraints that make this intractable.

- **Static**. A forecast model is fit once and used to make predictions.
- **Dynamic**. A forecast model is fit on newly available data prior to each prediction.

***Do you require a static or a dynamically updated model?***

## 2.9 Contiguous vs. Discontiguous

A time series where the observations are uniform over time may be described as contiguous. Many time series problems have contiguous observations, such as one observation each hour, day, month or year. A time series where the observations are not uniform over time may be described as discontiguous. The lack of uniformity of the observations may be caused by missing or corrupt values. It may also be a feature of the problem where observations are only made available sporadically or at increasingly or decreasingly spaced time intervals. In the case of non-uniform observations, specific data formatting may be required when fitting some models to make the observations uniform over time.

- **Contiguous**. Observations are made uniform over time.
- **Discontiguous**. Observations are not uniform over time.

***Are your observations contiguous or discontiguous?***

## 2.10 Framework Review

To review, the themes and questions you can ask about your problem are as follows:

1. **Inputs vs. Outputs**: What are the inputs and outputs for a forecast?
2. **Endogenous vs. Exogenous**: What are the endogenous and exogenous variables?
3. **Unstructured vs. Structured**: Are the time series variables unstructured or structured?

4. **Regression vs. Classification**: Are you working on a regression or classification predictive modeling problem? What are some alternate ways to frame your time series forecasting problem?
5. **Univariate vs. Multivariate**: Are you working on a univariate or multivariate time series problem?
6. **Single-step vs. Multi-step**: Do you require a single-step or a multi-step forecast?
7. **Static vs. Dynamic**: Do you require a static or a dynamically updated model?
8. **Contiguous vs. Discontiguous**: Are your observations contiguous or discontiguous?

## 2.11 Extensions

This section lists some ideas for extending the tutorial that you may wish to explore.

- **Apply Taxonomy**. Select a standard time series dataset and work through the questions in the taxonomy to learn more about the dataset.
- **Standard Form**. Transform the taxonomy into a form or spreadsheet that you can re-use on new time series forecasting projects going forward.
- **Additional Characteristic**. Brainstorm and list at least one additional characteristic of a time series forecasting problem and a question that you might used to identify it.

If you explore any of these extensions, I'd love to know.

## 2.12 Further Reading

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

- *Machine Learning Strategies for Time Series Forecasting*, 2013.
  http://link.springer.com/chapter/10.1007%2F978-3-642-36318-4_3
- *Recursive and direct multi-step forecasting: the best of both worlds*, 2012.
  https://econpapers.repec.org/paper/mshebswps/2012-19.htm

## 2.13 Summary

In this tutorial, you discovered a framework that you can use to quickly understand and frame your time series forecasting problem. Specifically, you learned:

- A structured way of thinking about time series forecasting problems.
- A framework to uncover the characteristics of a given time series forecasting problem.
- A suite of specific questions, the answers to which will help to define your forecasting problem.

### 2.13.1 Next

In the next lesson, you will discover a systematic process to ensure that you get better than average performance on your next time series forecasting project.

# Chapter 4

# How to Transform Time Series to a Supervised Learning Problem

Time series forecasting can be framed as a supervised learning problem. This re-framing of your time series data allows you access to the suite of standard linear and nonlinear machine learning algorithms on your problem. In this lesson, you will discover how you can re-frame your time series problem as a supervised learning problem for machine learning. After reading this lesson, you will know:

- What supervised learning is and how it is the foundation for all predictive modeling machine learning algorithms.
- The sliding window method for framing a time series dataset and how to use it.
- How to use the sliding window for multivariate data and multi-step forecasting.

Let's get started.

## 4.1 Supervised Machine Learning

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables ($X$) and an output variable ($y$) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X) \tag{4.1}$$

The goal is to approximate the real underlying mapping so well that when you have new input data ($X$), you can predict the output variables ($y$) for that data. Below is a contrived example of a supervised learning dataset where each row is an observation comprised of one input variable ($X$) and one output variable to be predicted ($y$).

```
X,  y
5,  0.9
4,  0.8
5,  1.0
3,  0.7
4,  0.9
```

Listing 4.1: Example of a small contrived supervised learning dataset.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by making updates. Learning stops when the algorithm achieves an acceptable level of performance. Supervised learning problems can be further grouped into regression and classification problems.

- **Classification**: A classification problem is when the output variable is a category, such as `red` and `blue` or `disease` and `no disease`.
- **Regression**: A regression problem is when the output variable is a real value, such as `dollars` or `weight`. The contrived example above is a regression problem.

## 4.2 Sliding Window

Time series data can be phrased as supervised learning. Given a sequence of numbers for a time series dataset, we can restructure the data to look like a supervised learning problem. We can do this by using previous time steps as input variables and use the next time step as the output variable. Let's make this concrete with an example. Imagine we have a time series as follows:

```
time,  measure
1,     100
2,     110
3,     108
4,     115
5,     120
```

Listing 4.2: Example of a small contrived time series dataset.

We can restructure this time series dataset as a supervised learning problem by using the value at the previous time step to predict the value at the next time step. Re-organizing the time series dataset this way, the data would look as follows:

```
X,   y
?,   100
100, 110
110, 108
108, 115
115, 120
120, ?
```

Listing 4.3: Example of time series dataset as supervised learning.

Take a look at the above transformed dataset and compare it to the original time series. Here are some observations:

- We can see that the previous time step is the input ($X$) and the next time step is the output ($y$) in our supervised learning problem.
- We can see that the order between the observations is preserved, and must continue to be preserved when using this dataset to train a supervised model.

- We can see that we have no previous value that we can use to predict the first value in the sequence. We will delete this row as we cannot use it.
- We can also see that we do not have a known next value to predict for the last value in the sequence. We may want to delete this value while training our supervised model also.

The use of prior time steps to predict the next time step is called the sliding window method. For short, it may be called the window method in some literature. In statistics and time series analysis, this is called a lag or lag method. The number of previous time steps is called the window width or size of the lag. This sliding window is the basis for how we can turn any time series dataset into a supervised learning problem. From this simple example, we can notice a few things:

- We can see how this can work to turn a time series into either a regression or a classification supervised learning problem for real-valued or labeled time series values.
- We can see how once a time series dataset is prepared this way that any of the standard linear and nonlinear machine learning algorithms may be applied, as long as the order of the rows is preserved.
- We can see how the width sliding window can be increased to include more previous time steps.
- We can see how the sliding window approach can be used on a time series that has more than one value, or so-called multivariate time series.

We will explore some of these uses of the sliding window, starting next with using it to handle time series with more than one observation at each time step, called multivariate time series.

## 4.3 Sliding Window With Multiple Variates

The number of observations recorded for a given time in a time series dataset matters. Traditionally, different names are used:

- **Univariate Time Series**: These are datasets where only a single variable is observed at each time, such as temperature each hour. The example in the previous section is a univariate time series dataset.
- **Multivariate Time Series**: These are datasets where two or more variables are observed at each time.

Most time series analysis methods, and even books on the topic, focus on univariate data. This is because it is the simplest to understand and work with. Multivariate data is often more difficult to work with. It is harder to model and often many of the classical methods do not perform well.

> Multivariate time series analysis considers simultaneously multiple time series. [...]
> It is, in general, much more complicated than univariate time series analysis

— Page 1, *Multivariate Time Series Analysis: With R and Financial Applications*, 2013.

The sweet spot for using machine learning for time series is where classical methods fall down. This may be with complex univariate time series, and is more likely with multivariate time series given the additional complexity. Below is another worked example to make the sliding window method concrete for multivariate time series. Assume we have the contrived multivariate time series dataset below with two observations at each time step. Let's also assume that we are only concerned with predicting `measure2`.

```
time,  measure1,  measure2
1,     0.2,       88
2,     0.5,       89
3,     0.7,       87
4,     0.4,       88
5,     1.0,       90
```

Listing 4.4: Example of a small contrived multivariate time series dataset.

We can re-frame this time series dataset as a supervised learning problem with a window width of one. This means that we will use the previous time step values of `measure1` and `measure2`. We will also have available the next time step value for `measure1`. We will then predict the next time step value of `measure2`. This will give us 3 input features and one output value to predict for each training pattern.

```
X1,   X2,   X3,   y
?,    ?,    0.2,  88
0.2,  88,   0.5,  89
0.5,  89,   0.7,  87
0.7,  87,   0.4,  88
0.4,  88,   1.0,  90
1.0,  90,   ?,    ?
```

Listing 4.5: Example of a multivariate time series dataset as a supervised learning problem.

We can see that as in the univariate time series example above, we may need to remove the first and last rows in order to train our supervised learning model. This example raises the question of what if we wanted to predict both `measure1` and `measure2` for the next time step? The sliding window approach can also be used in this case. Using the same time series dataset above, we can phrase it as a supervised learning problem where we predict both `measure1` and `measure2` with the same window width of one, as follows.

```
X1,   X2,   y1,   y2
?,    ?,    0.2,  88
0.2,  88,   0.5,  89
0.5,  89,   0.7,  87
0.7,  87,   0.4,  88
0.4,  88,   1.0,  90
1.0,  90,   ?,    ?
```

Listing 4.6: Example of a multivariate time series dataset as a multi-step or sequence prediction supervised learning problem.

Not many supervised learning methods can handle the prediction of multiple output values without modification, but some methods, like artificial neural networks, have little trouble. We can think of predicting more than one value as predicting a sequence. In this case, we were

predicting two different output variables, but we may want to predict multiple time steps ahead of one output variable. This is called multi-step forecasting and is covered in the next section.

## 4.4 Sliding Window With Multiple Steps

The number of time steps ahead to be forecasted is important. Again, it is traditional to use different names for the problem depending on the number of time steps to forecast:

- **One-step Forecast**: This is where the next time step (`t+1`) is predicted.
- **Multi-step Forecast**: This is where two or more future time steps are to be predicted.

All of the examples we have looked at so far have been one-step forecasts. There are a number of ways to model multi-step forecasting as a supervised learning problem. For now, we are focusing on framing multi-step forecast using the sliding window method. Consider the same univariate time series dataset from the first sliding window example above:

```
time, measure
1,    100
2,    110
3,    108
4,    115
5,    120
```

Listing 4.7: Example of a small contrived time series dataset.

We can frame this time series as a two-step forecasting dataset for supervised learning with a window width of one, as follows:

```
X1,   y1,   y2
?     100,  110
100,  110,  108
110,  108,  115
108,  115,  120
115,  120,  ?
120,  ?,    ?
```

Listing 4.8: Example of a univariate time series dataset as a multi-step or sequence prediction supervised learning problem.

We can see that the first row and the last two rows cannot be used to train a supervised model. It is also a good example to show the burden on the input variables. Specifically, that a supervised model only has $X1$ to work with in order to predict both $y1$ and $y2$. Careful thought and experimentation are needed on your problem to find a window width that results in acceptable model performance.

## 4.5 Implementing Data Preparation

In Chapter 6 we touch on data preparation an focus on how to transform time series data in order to meet the expectations of a three-dimensional structure by some deep learning methods. In Chapters 7, 8, and 9 we will explore how to implement deep learning methods for univariate,

multivariate and multi-step forecasting with a specific focus on how to prepare the data for model. In each chapter we will develop functions that can be reused to prepare time series data on future projects.

## 4.6 Extensions

This section lists some ideas for extending the tutorial that you may wish to explore.

- **Samples in Detail**. Explain how the choice of the number of time steps in a sample impacts training a model and making predictions with a trained model.
- **Work Through Dataset**. Select a standard time series forecasting dataset and demonstrate how it may be transformed into samples for supervised learning.
- **Develop Function**. Develop a function in Python to transform a time series into a samples for supervised learning.

If you explore any of these extensions, I'd love to know.

## 4.7 Further Reading

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

- *Multivariate Time Series Analysis: With R and Financial Applications*, 2013. https://amzn.to/2KD4rw7
- *Machine Learning for Sequential Data: A Review*, 2002. https://link.springer.com/chapter/10.1007/3-540-70659-3_2
- *Machine Learning Strategies for Time Series Forecasting*, 2013. https://link.springer.com/chapter/10.1007/978-3-642-36318-4_3

## 4.8 Summary

In this lesson, you discovered how you can re-frame your time series prediction problem as a supervised learning problem for use with machine learning methods. Specifically, you learned:

- Supervised learning is the most popular way of framing problems for machine learning as a collection of observations with inputs and outputs.
- Sliding window is the way to restructure a time series dataset as a supervised learning problem.
- Multivariate and multi-step forecasting time series can also be framed as supervised learning using the sliding window method.

### 4.8.1 Next

In the next lesson, you will discover naive and classical time series forecasting methods that you absolutely must evaluate before investigating more sophisticated deep learning methods.