Python for Finance and Algorithmic Trading

Machine Learning, Deep Learning, Time Series Analysis, Risk and Portfolio Management for Metatrader™ 5 Live Trading

POWERED BY

jupyter

Quantreo

Lucas Inglese

Book Link : https://www.amazon.com/gp/product/B09HG18CYL

EXTRACT Chapter 12: Recurrent neural network

In this chapter, we will talk about Recurrent Neural Networks (RNN). This algorithm specialized in time series can be a precious ally to create a trading strategy because we use time series. Moreover, it is one of the most complex deep learning algorithms. We will implement the strategies using the Netflix stock.

12.1 Principles of RNN

In this section, we are going to learn how an RNN works. Whereas we will see the principles of RNN, if you want to go deeper into the subject, I invite you to check some internet papers.

12.1.1. How an RNN works

In this part, we will speak about the intuition behind the RNN[1]. RNN models are specialized in time series modeling. Indeed, the network structure is slightly different from the ANN because the hidden layers are interconnected.

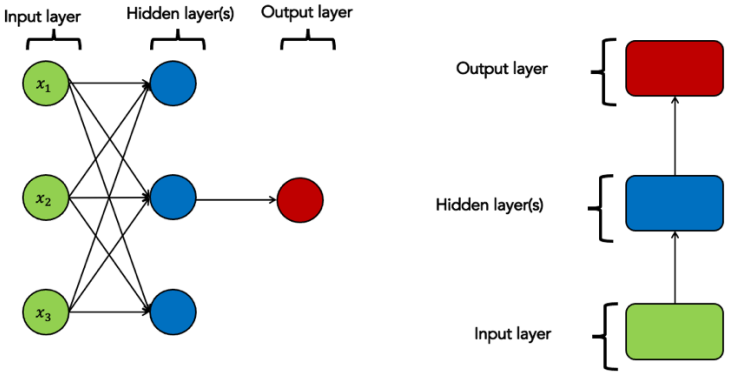
The interconnection between the hidden layer allows the model to "remember" the past. In the RNN, there are two types of memory instead of the ANN. The RNN has its long-term memory but also a short-term memory created by the interconnection that we speak previously.

To better understand how an RNN works, we need to do a little representation. To do it, we need to simplify the representation of the ANN. We are going to schematize the ANN only with his layer and not with all neurons. It allows us to visualize the model straightforward to create the representation for the RNN easier. Let us see figure 12.1.

---

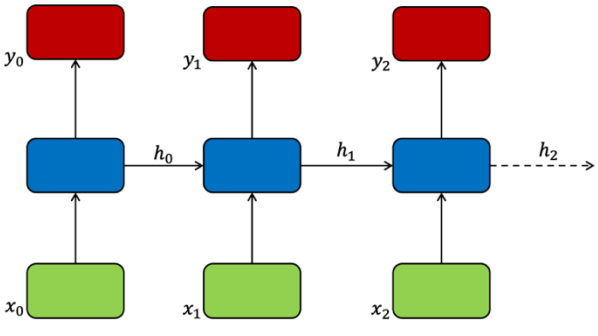
[1] **Additional lecture**: Recurrent Neural Networks cheatsheet, Afshine Amidi

Figure 12.1: New schematization of the ANN



This figure showed the representation of an ANN using the layer only with the square instead of the left with all neurons. However, it is the same ANN.

Figure 12.2: Schematization of RNN



As we can see, the RNN is nothing else that ANN connected. We see that the information of the hidden layer is given directly to the next hidden layer, and it is not integrated only with the weights of the models.

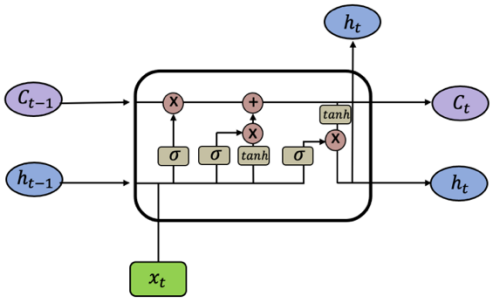
The issue with the RNN is that if we use the same neurons as the ANN, the model can have some issues with the gradient. Indeed, there are many more derivations than in the ANN, and it can be caused vanishing gradient issues. At the same time, the researcher has found a solution to this problem. They are created new neurons named Long

Short-Term Memory (LSTM) neuron. It is slightly different from the structure of a neuron we see in the last part, so we need to study it.

12.1.2. LSTM neuron

In this part, we are going to explain briefly how works an LSTM cell. Naturally, it is not a deep learning course. So, if you want to understand better the LSTM cell, you need to go on the internet because this subject is not relevant in this book. Let us know what it looks like for an LSTM cell. Then, we explain the difference between the dense cell and the LSTM cell.

Figure 12.3: LSTM cell



This figure shows the functioning of an LSTM cell where the memory at time t is C_t, the information h_t. So, we can say that is the long-term memory and is the short-term memory.

To better understanding the LSTM cell, we are going to take a little example. Suppose you have an algorithm that predicts the next word, for example. You need C, which is the sentence, and you need h, which is some adjective, and x, the previous word.

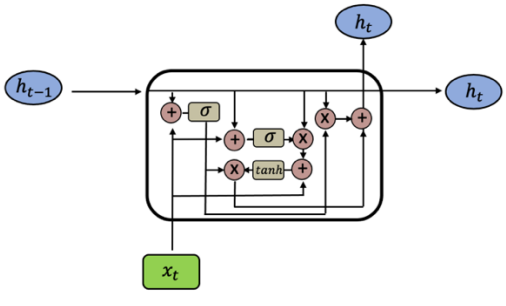
The sigma is the sigmoid activation function. The plus addition the information and the time work like a plumbing valve it will let pass information or not using the property of the sigmoid function.

So, we are not going more profound, and I can understand if this notion is not understood, but this section has only to give you some information to understand how we will create the strategy using RNN and not be a complete deep learning course.

12.1.3. GRU cell

In this part, we will see derivatives of the LSTM neuron, the Gated Recurrent Unit (GRU). It is an excellent alternative to the LSTM because it takes only the information from the previous neuron and not the memory, which can be problematic in our case. Indeed, if you remember how the market reacted 15 years ago, it will not be the same as the market's reaction now because the market institutions changed, the technologies have been involved, etc...

Figure 12.4: GRU cell



As we can see in the GRU cell, only the information h, we do not consider the memory C_t instead of the LSTM.

Even if the GRU and the LSTM are different, you need to test both when creating your trading strategy on an RNN model.

In this section, we have seen much information about RNN. You can reread this section if you are uncomfortable with this notion or go to

the next section because the theory is necessary to understand what we do but not to create the model. After all, we already know the functioning of an ANN, and in fact, the RNN is similar even if it has some specificities. Now, you are armed to begin the code of the RNN using TensorFlow.

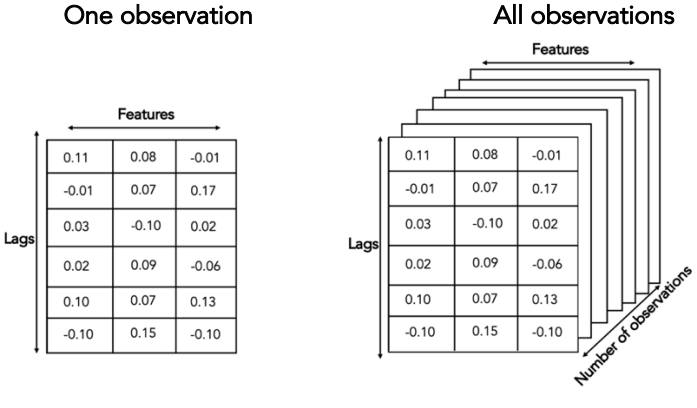
12.2. RNN for classification

In this section, we are going to speak about the RNN for classification tasks. First, we will see how to transform our data into 3d data (necessary for classification and regression). We are going to implement our model and see how to backtest the created strategy.

12.2.1. Transform 2d data to 3d data

In this part, we will explain why and how to transform our 2d data into 3d data. If you are understood the theory behind the RNN very well (even if it is very hard with the bit of information that I give you), you have understood that this network takes for each x a matrix of shape (lags, m) where lag is the number of lag and m is the number of features. So, if we are n observations, we are a matrix with a shape (n, lags, m). Figure 12.5 represents the database we need.

Figure 12.5: Schematization of the data for an RNN



Book Link : https://www.amazon.com/gp/product/B09HG18CYL

As we can see, the database for an RNN needs to be 3 dimensional. Indeed, it is the same rules as for the ANN, but for the ANN, we have a line as observation so a 1-dimensional array, and when we have more than one, the matrix become 2-dimensional instead of the RNN, which needs a 2-dimensional array by observation and naturally if you have more than one, the array will be 3-dimensional.