INSIGHT.DATA.CLARITY.

September 2018

A NEW HYPE OR A NEW HOPE?

Neural networks applied to live market data

Executive summary

Recent advances in Artificial Intelligence (AI) are linked to the unquestionable success of neural networks. But can this technology be applied to the ultimate challenge of quantitative finance, predicting future prices? In this note we explain briefly how these systems work. Then we present a simple experiment where we turn a neural network into a predictor of price returns, by using the gigantic amount of data available on the markets at high-frequency scales. We also explore how much the results depend on the complexity of the network (its 'depth') in this particular case.

Contact details

Call us +33 1 49 49 59 49

Email us cfm@cfm.fr

Introduction

The field of Artificial Intelligence has been under the spotlight in recent years because of revolutionary results in domains as diverse as: computer vision [1], machine translation [2], speech recognition [3], natural language processing [4] and machines playing games [5].

We now have computers that can recognise the content of an image and describe it, beat humans at the game of Go, generate very realistic fake images or human-like voices, and use those voices to book a table in a restaurant!

The key actor of this revolution is the neural network, a not-so-recent machine-learning architecture whose incredible potential has only become more and more obvious in the last years.

Most practitioners in finance have been cautious with the direct application of sophisticated machine-learning: indeed the level of noise in financial data, compared to the level of signal, dwarfs what we see in the context of applications listed above. For that reason, neural networks are often associated with tasks that contribute only indirectly to the constitution of market signals: extracting market sentiment from news or tweets, recognising some activity on satellite photos, etc.

The finance industry has, nonetheless, a large competitive advantage in this field: the amount of financial data can be colossal, in particular when one considers very short time frequencies like the ones involved in execution algorithms or high-frequency trading strategies. In that case one can use 'tick-by-tick' data: that is, the information on each individual order received by the market.

In this note, we will run an experiment combining neural networks with proprietary tick-by-tick data and market indicators. We will obtain a predictor of future returns, whose performance will be compared to the result of a much more basic method: the linear regression.

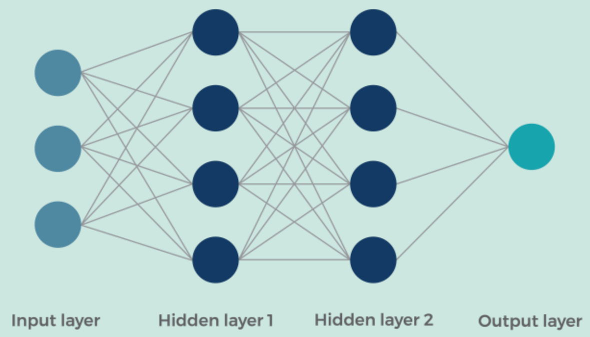
How neural networks work

The notion of a neural network covers a fairly diverse family of machine-learning architectures. Here we will only use the simplest of these architectures, called the multilayer perceptron.

This architecture is shown on Figure 1. It is composed of several layers, each of them containing a certain number of units called neurons. All the neurons in one layer are connected to the neurons of the previous layer, and each of these connections is associated with a parameter called its weight. This means that if you have 4 layers of 50 neurons each, you already have around 50 X 50 X 4 = 10,000 weights in the system.

To understand how these systems work, let us consider the favourite challenge of a quantitative shop: predict future prices! Let's say that we observe some market information on an asset at some point in time (like the content of its orderbook, cf. next section) and we want to predict the future return of the price on that same asset. We will call this return the target.

A neural network can take the market information as its inputs, combine them with its weights, and generate an output through a calculation described in Figure 1.



When feeding a neural network with some inputs $x_1,\cdots,x_N$, it calculates an output using the following algorithm:

- On the first layer, each neuron will directly read all the inputs, multiply each of them by a weight, sum them, add one more weight (called the *bias*) and apply a very basic function to the sum (called the *activation function*). This gives one output per neuron:

$$\text{Output}^j = \text{Activ}\left(\sum_{i=1}^{N} w_i^j \cdot x_i + b^j\right)$$

  The outputs are different for each neuron because they have different weights.
- On the second layer, the exact same process applies, except that this time the inputs are the results of the neurons of the first layer. We apply again some weights for each neuron, an activation function, and we obtain the results of the neurons of the second layer. This process goes on like this for each additional layer.
- On the last layer we have only one neuron: it weights and sums once again the results of the previous layer, and this sum (without applying an activation function) is the output of the full network.

Figure 1: Description of a neural network

Initially, the weights are completely random, so of course there is no reason for this output to be close to the target. This is why we need to train the network with past data:

- We read market information at a given point in time and we calculate the output of the neural network when fed with this data.
- We compare this output with the actual return observed at that point of time.
- Then we modify the weights of each neuron in each layer so that the new result that we would obtain with this new network is closer to the target; in other words, we correct the weights to make the network more predictive. This fundamental step of training is named backpropagation, because the difference between the output and the target is sent back into the network.

In the end, this whole process is arguably the most laborious one can imagine. We just read some data and tweak the weights to better fit them, then read more data and tweak the weights again, etc. So one wonders why it is only in recent years that neural networks have become so powerful. The answer fits in four letters: D-A-T-A Because of its large number of weights to tweak, this architecture actually needs an enormous amount of data to train it before it starts to provide sensible results. For image recognition in particular, this data was not easily accessible until recent years.

It is also often argued that advances in computing power have significantly helped to process all these data and train neural networks efficiently. In particular the rise of Graphical Processing Units (GPUs), originally developed to handle resource-intensive video games, has contributed significantly to the recent progress in artificial intelligence.

Finally, one should bear in mind the remarkable stream of innovation around the neural network technology. Many more complex architectures have been developed, lots of tricks and optimisations have been discovered to improve the training speed and the predictive power of these networks. It is not the subject of this note to present them, but some have been used in the experiment we now describe.

Experiment with orderbook data

Our goal is to investigate how these architectures and their training procedures perform on financial data, at a rather high-frequency scale where the data is abundant. The first input we will feed our networks with will be the content of the orderbook: it is the list of all buy and sell intentions sent to the market by different market participants, at different prices and with different sizes. We add to these raw data a few proprietary predictive signals, based on our own experience and understanding of market microstructure.

The setting of the experiment is the following:

- Architecture: we use a neural network with two hidden layers of 100 neurons each.
- The target is the return between the price at the moment of observation and the price one minute later.
- The universe of instruments is a pool of 1,000 liquid US stocks.
- We use two months of tick-by-tick data to train and two months to validate.

We trained the system on a Tesla v100 GPU for six hours. We compared the predictive power of that network to the result of a much simpler technique, the linear regression, on the same inputs. Results are shown on Figure 2: note how the predictive power of the neural network improves during training, as the network progressively adapts its weights.

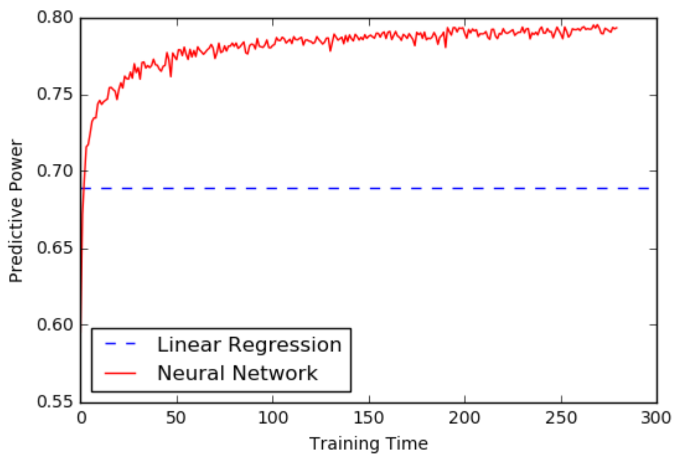


Figure 2: Performance of the neural network, compared to a linear regression

In the end, even though the simple regression already has good performances, the training produces a stronger predictor. An execution system can use this information by trading when the networks predict a (strong) positive return: this will lead to significant savings in trading costs, which are an important component of any active strategy [6].

It can be interesting to know how much value the network has extracted from each of its inputs. This information is straightforward when using a linear regression, but much less so with a neural network, which behaves a bit like a black box regarding how it handles its inputs. One way to get over this is simply to train the system in configurations where some inputs have been removed, and to observe the resulting degradation in performance.

This gives us the results shown on Figure 3: the neural network manages to extract value from raw orderbook data, and from all predictors except one (Predictor #1), which happens to be the least predictive of all.

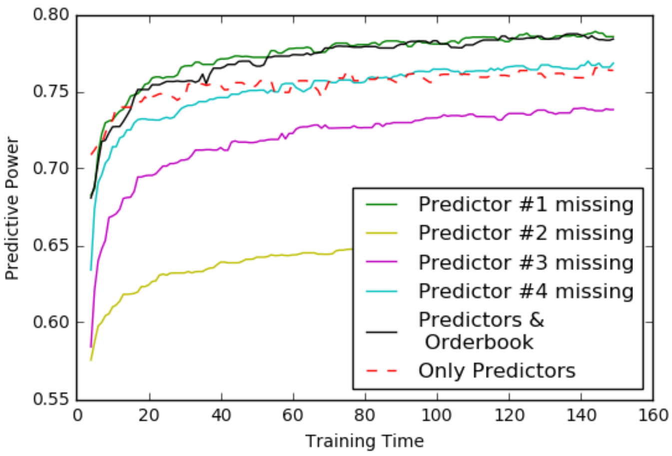


Figure 3: Performance obtained when removing predictors or orderbook data

Do we need to go deeper?

The architecture described above, even though it makes use of some modern features of neural networks, is not very deep: it does not contain a lot of layers. This is in contrast with what has happened for many tasks in A.I. like computer vision, where the number of layers of state-of-the-art neural networks has increased exponentially in the last years: cf. Figure 4.

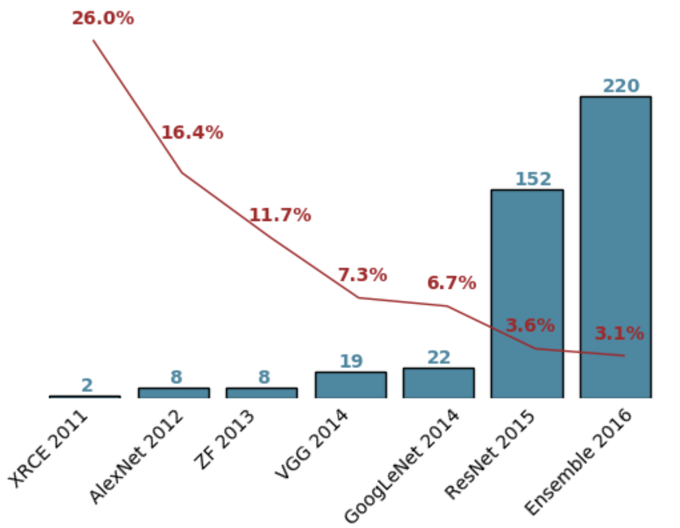


Figure 4: Evolution of the precision (expressed as percentage of errors, in red) vs. number of layers (in blue) of best-performing architectures on computer vision tasks.

This inflation of depth is the origin of the term 'deep learning' often used to describe the field.

To check whether a deeper architecture might prove beneficial in our setting, we trained several neural networks with many more layers, on the same data. The results in Figure 5 show that depth does not really improve the predictive power in our case. Why is that so?

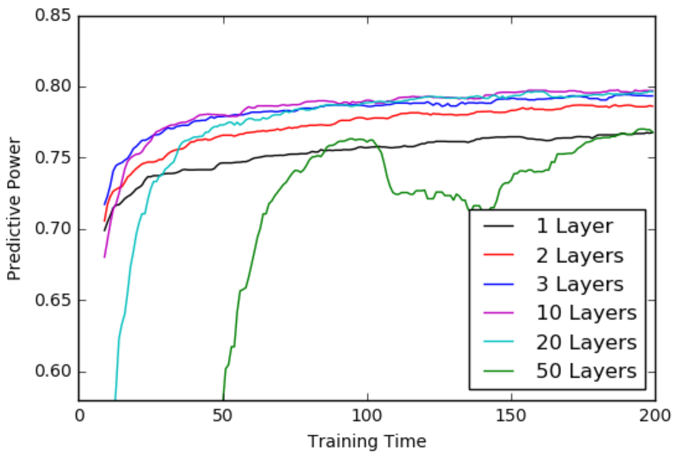


Figure 5: Evolution of performance when increasing the number of layers.

One way to grasp the idea of what the depth of a neural network means is to imagine that each layer represents a certain level of abstraction. The first layers will be a slight deformation of the inputs, but the more we progress into the network, the more we will represent the data in an abstract way, which fits more to the target.

Now our data (orderbook data and price indicators) are very simple structures, in contrast with real-life data which have more complex structure, like images, sounds or texts. Therefore, in our case, we probably do not need many levels of abstraction above our data to summarise all their information content. This might explain the modest contribution of depth that we observe.

Conclusion

Neural networks and artificial intelligence are not (at least for now) the magic solution to all the problems we encounter in quantitative finance. Understanding the behaviour of markets, assessing risk and market impact or building an optimised portfolio still require a scientific vision of the problem we face, rather than just applying a technical recipe.

However, this note shows that, when the amount of available data is vast, these techniques can provide remarkable results, and bring real added value to a more classical quantitative approach. The continuous progress in that research domain has to be closely followed by practitioners, provided they have the data and computing capabilities to make a sensible use of these technologies.

References

[1] Alex Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[2] Ilya Sutskever et al. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, pages 3104–3112. MIT Press, 2014.

[3] Geoffrey Hinton et al. Deep neural networks for acoustic modeling in speech recognition. Signal Processing Magazine, 2012.

[4] Ronan Collobert et al. Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12:2493–2537, November 2011.

[5] David Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, January 2016.

[6] Executing with impact: why the price you want is not the price you get! Technical Note, Capital Fund Management, November 2016.

Disclaimer

CFM has pioneered and applied an academic and scientific approach to financial markets, creating award winning strategies and a market leading investment management firm.

Contact us

Capital Fund Management S.A.
23, rue de l'Université, 75007
Paris, France
T +33 1 49 49 59 49
E cfm@cfm.fr

CFM International Inc.
The Chrysler Building, 405 Lexington Avenue - 55th Fl.,
New York, NY, 10174, U.S.A
T +1 646 957 8018
E cfm@cfm.fr

Capital Fund Management LLP – Sydney office
Level 16, 333 George Street
Sydney, NSW, 2000, Australia
T +61 04 39 16 0125
E cfm@cfm.fr

CFM Asia KK
9F Marunouchi Building, 2-4-1, Marunouchi, Chiyoda-Ku,
100-6309 Tokyo, Japan
T +81 3 5219 6180
E cfm@cfm.fr

Capital Fund Management LLP
64 St James's Street, London
SW1A 1NF, UK
T +44 207 659 9750
E cfm@cfm.fr