Financial Pricing Engine with Machine Learning

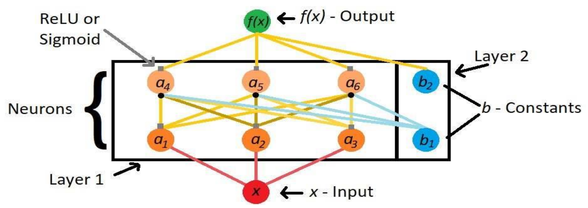
Vital Mendonca Filho(MA), Pavee Phongsopa(MA), Nicholas Wotton(MA)
1st-Advisor: Song Qinghuo(Department of Mathematical Science)
2nd-Advisor: Gu Wang (Department of Mathematical Science)
3rd-Advisor: Li Yanhua (Department of Computer Science)

Abstract

The advancement of computing power has allowed mathematicians, statisticians, computer scientists and other professionals to solve complex problems using powerful machines. The progress in data science, more specifically in deep learning opened ways to solve high dimensional equations, bypassing the curse of dimensionality, a phenomena which illustrates the complexity of equations as they grow in degree. In this project we seek to use deep learning tools to price exotic financial products, combining stochastic partial differential equation and neural networks.

Neural Network

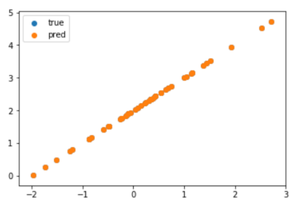
Deep learning refers to the study of large, complex data sets through neural networks. The idea is that we feed the system a functions and neurons in deep layers will be able to interpret a result for such input after extensive ”training”. A typical model usually consists of a few neurons in each layer as shown below.



$$\left(\begin{bmatrix} x \end{bmatrix}\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}+\begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}\right)\begin{bmatrix} a_4 \\ a_5 \\ a_6 \end{bmatrix}+\begin{bmatrix} b_4 \end{bmatrix}=\begin{bmatrix} f(x) \end{bmatrix}$$

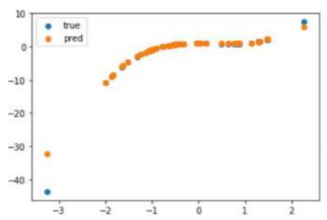
Approximation

Data show promising result in approximating linear function and varying degree polynomials.




```
model = nn.Sequential(
    nn.Linear(1, 3),
    nn.ReLu(),
    nn.Linear(3, 1)
)
```

```
model = nn.Sequential(
    nn.Linear(1, 20),
    nn.ReLu(),
    nn.Linear(20, 20),
    nn.Linear(20, 20),
    nn.Linear(20, 1)
)
```



Deep Learning in Existing Models

With success in our preliminary results, we look to apply the same method of approximation to financial models.

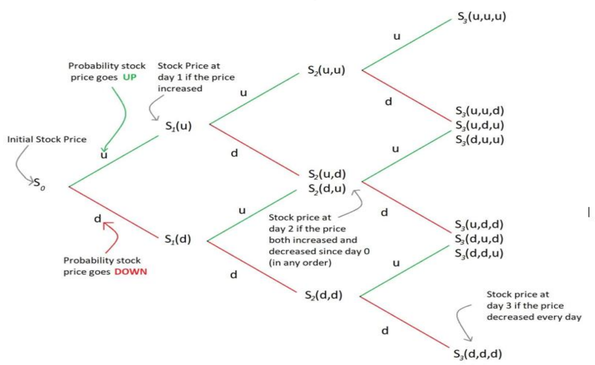
Black-Scholes Model

This is a mathematical model use for the derivation of European style option value with partial differential equation in the model known as Black-Scholes equation. Black-Scholes model assume the distribution of stock as lognormal that writes:

$$\ln\frac{S(T)}{S(0)} \sim \mathcal{N}\left((r-\frac{1}{2}\sigma^2)T,\ \sigma^2 T\right)$$

Binomial Option Pricing Model

Like Black-Scholes Model, the model produces a theoretical option value given a set of data. However, the model is broken down into number of “steps” which is further broken down into a binomial tree of “up” and “down.”



Implied Volatility

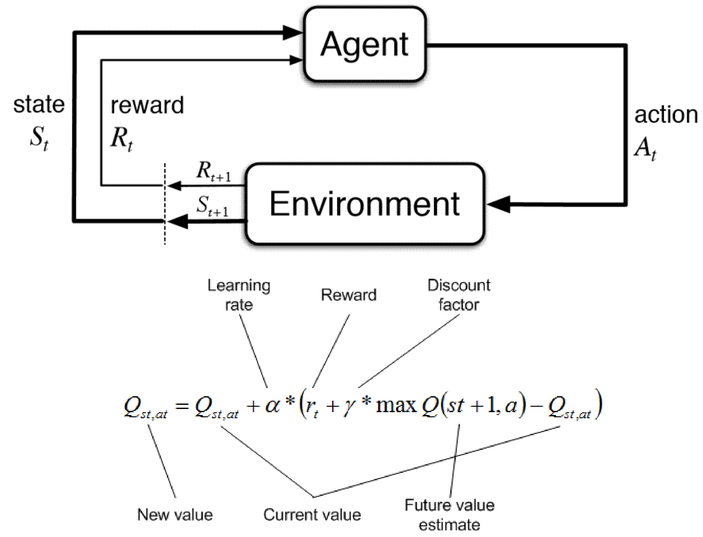
Option Pricing such as Black-Scholes model or CRR model requires variety of inputs to calculate the theoretical value of an option which includes the value of volatility. In most cases, volatility is not known.

References

- https://en.wikipedia.org/wiki/Binomial_options_pricing_model
- https://aaronschlegel.me/implied-volatility-functions-python.html
- https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model
- http://www.deeplearningbook.org/
- https://github.com/songqsh/MQP2019/blob/master/doc/HJE18_deep_bsde.pdf
- https://github.com/songqsh/MQP2019/blob/master/doc/EHJ17_deep_bsde.pdf

Reinforcement Learning aka Q-learning

After the completion of deep supervised learning on the 3 main models, we plan to tackle reinforcement learning, a method which has seen a lot of success in other fields in recent years.
What separate reinforcement learning from other type of machine learning is how the machine is getting continuous feedback loop from its environment according to its performance instead of being “trained” by numbers of data set.



Application

Currently, many online stockbroker websites offer Application Program Interface (API) that allows programs to gain access to information from the trading sites. Together with our algorithm, we can potentially create a software that will automatically receive information from the server, analyze that information, and send a trade order accordingly.