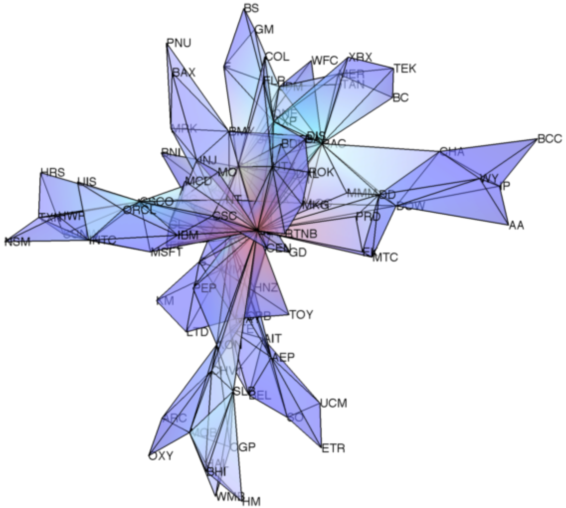
Deep learning the Limit Order Book What machines can learn & What can we learn form them?



Tomaso Aste

Financial Computing and Analytics Group

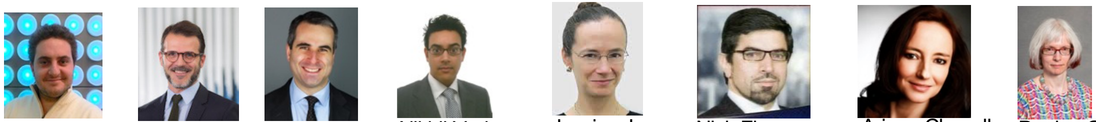
http://fincomp.cs.ucl.ac.uk/

The Financial Computing and Analytics research group investigates socio- economic systems using methods from computer science, applied mathematics, computational statistics and network theory.

UCL Centre for Blockchain Technologies

Systemic Risk Centre

Silvia Bartolucci   Philip Treleaven   Guido Germano   Fabio Caccioli   Christopher Clack   Giacomo Livan   Paolo Barucca   Carolyn Phelan   Daniel Hulme   Robert E Smith   Jiahua Xu   Tomaso Aste



Simone Righi   Paolo Tasca   Geoff Goodell   Nikhil Vadgama   Jessica James   Nick Firoozye   Ariane Chapelle   Denise Gorse

EPSRC Pioneering research and skills

Research Councils UK Digital Economy Transforming Business and Society

European Commission

E·S·R·C ECONOMIC & SOCIAL RESEARCH COUNCIL

Education:
PhD Doctoral Training Centre in Financial Computing
MSc Computational Finance
MSc Financial Risk Management
MSc Financial Technology, forthcoming 2021-22
MSc Emerging Digital Technologies, forthcoming 2021-22

Research
- computational finance
- data-driven modelling
- artificial intelligence
- financial risk management
- blockchain technology
- digital economy
- systemic risk
- numerical pricing of derivatives
- agent-based simulation
- empirical finance
- market microstructure
- algorithmic trading
- high-frequency trading
- data science
- big data analytics
- network analysis
- machine learning
- price formation
- portfolio optimization

What can machines learn?

The perceptron

(Rosemblatt ‘57, Minsky Papert ‘69)

Frank Rosenblatt with a Mark I Perceptron computer in 1960

New York Times 1958

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

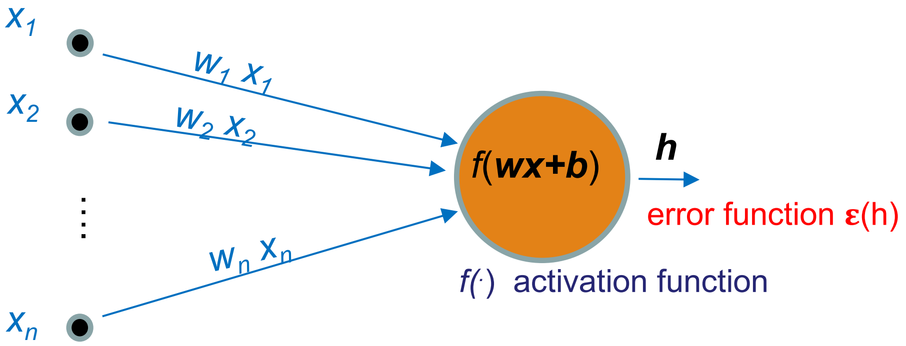
In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

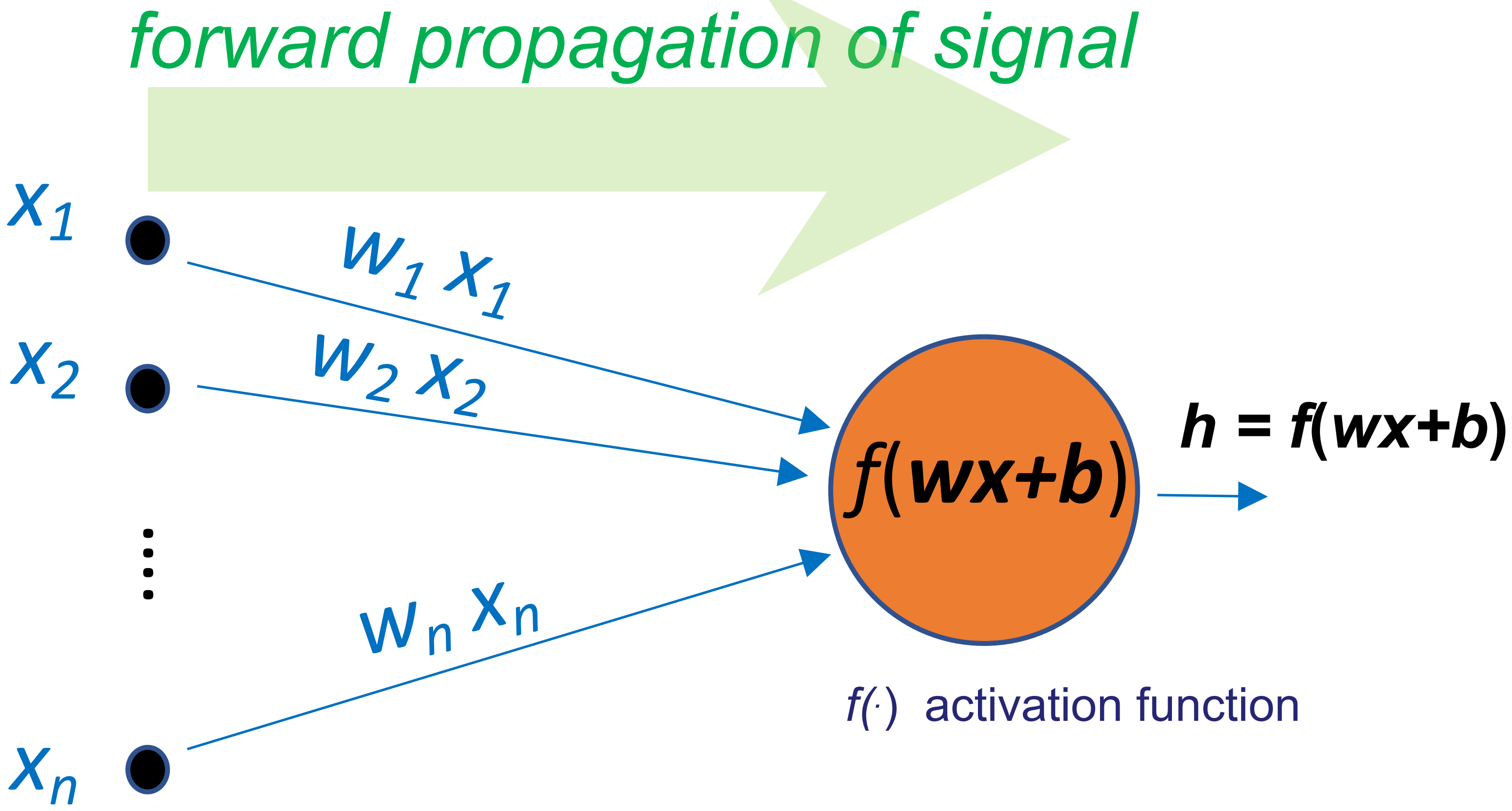
In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

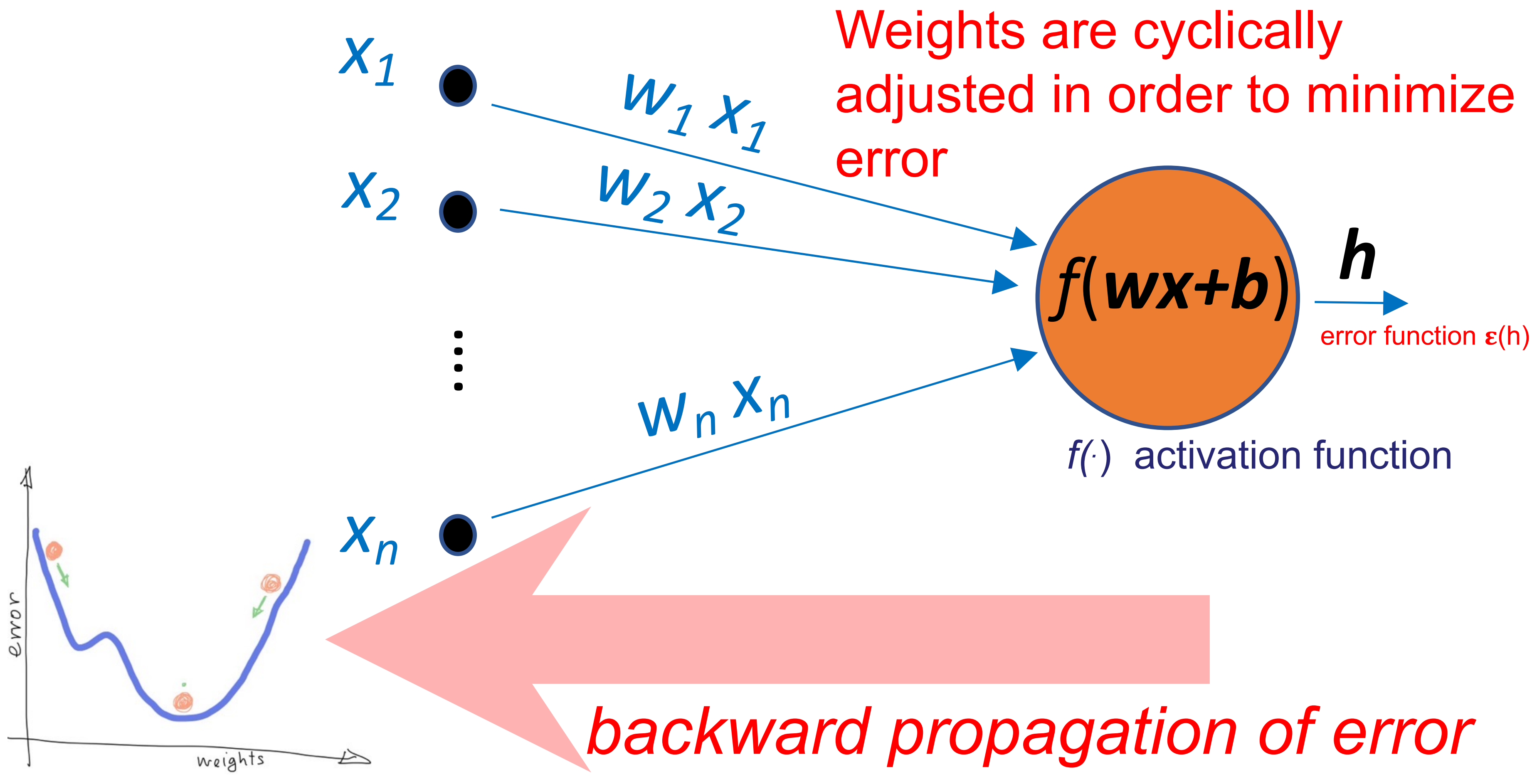
The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.



It learns from examples

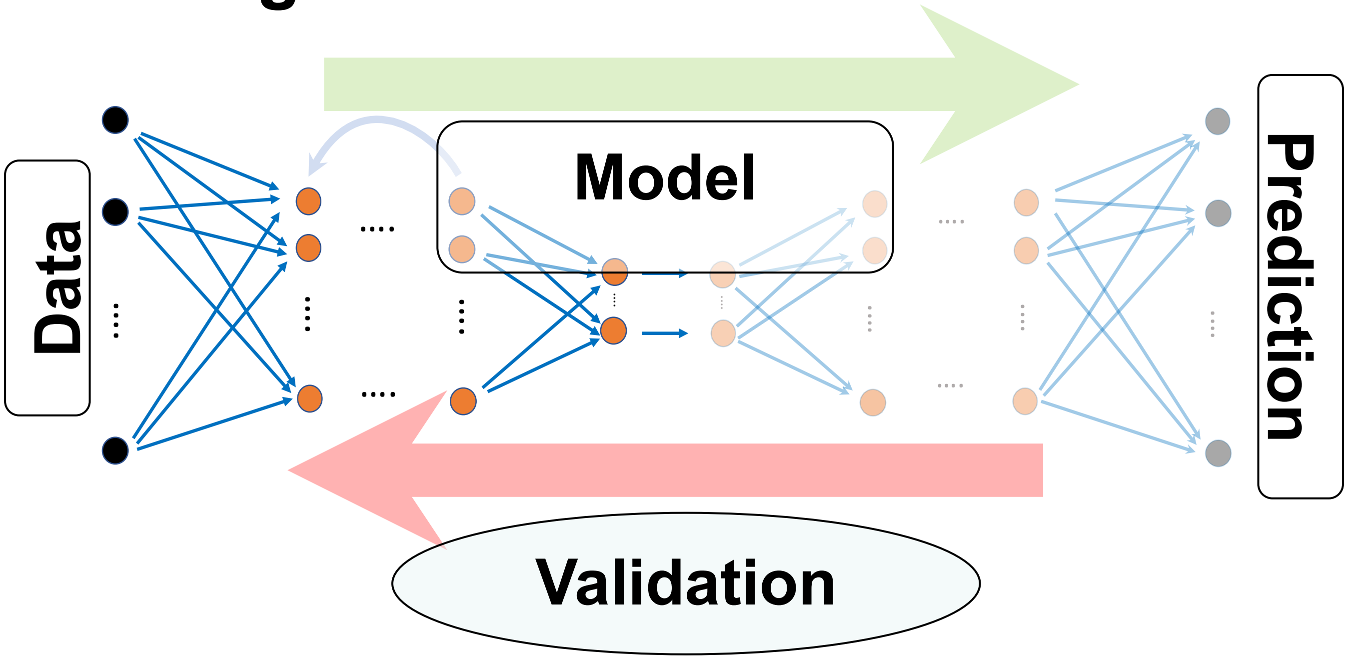


forward propagation of signal
x_1
x_2
x_n
$w_1 x_1$
$w_2 x_2$
$w_n x_n$
$f(wx+b)$
$h = f(wx+b)$
$f(\cdot)$ activation function



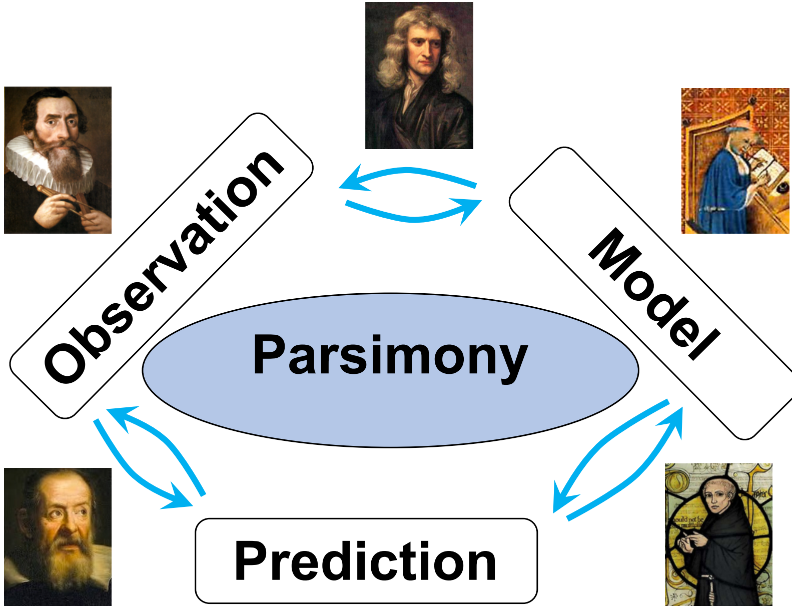
Weights are cyclically adjusted in order to minimize error
x_1
x_2
x_n
$w_1 x_1$
$w_2 x_2$
$w_n x_n$
$f(wx+b)$
h
error function ε(h)
$f(\cdot)$ activation function
error
weights
backward propagation of error

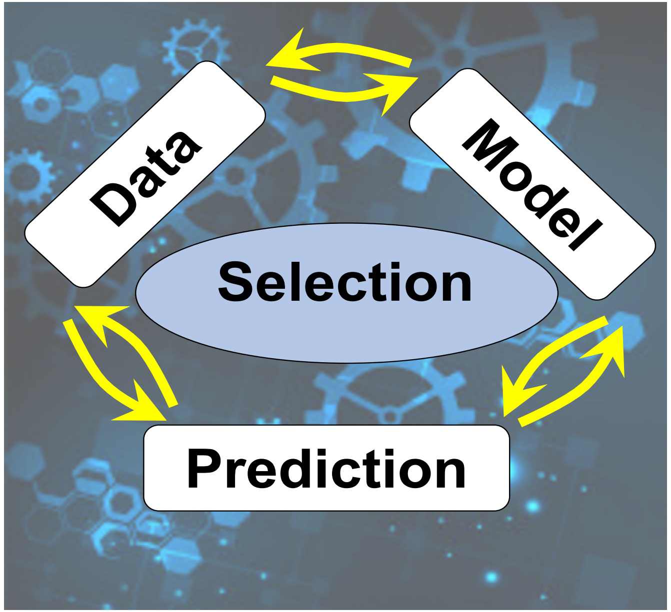
Machine learning method



The <u>circular learning approach</u> where a model is automatically learned from data through validation and optimization is <u>not new</u>

Scientific method vs. machine learning method



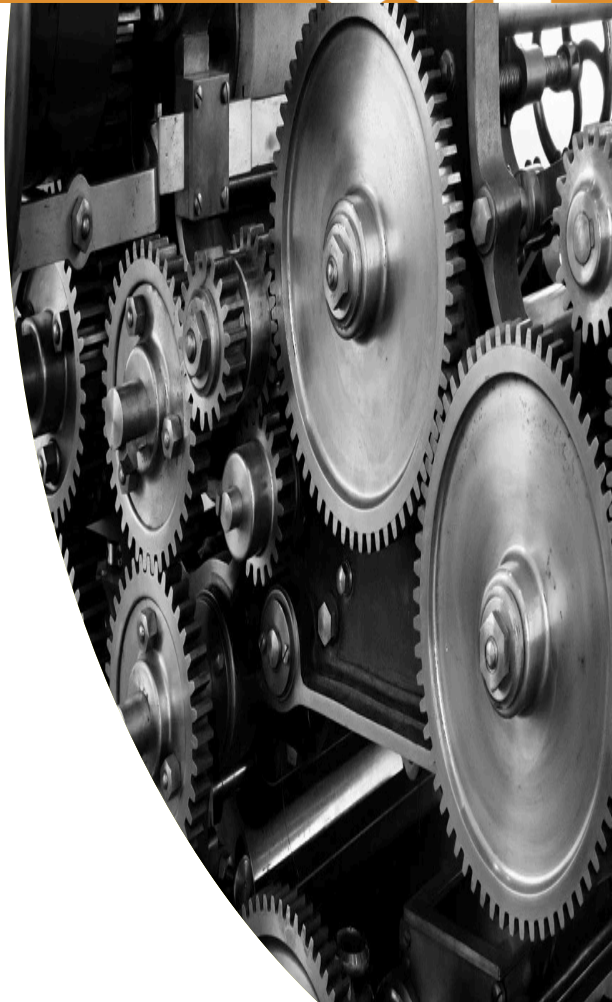


- What is novel is the 'bottom up' approach that does <u>not need human intuition</u> for the formulation of the model
- Parsimony is no longer central
- A very large number of models are automatically generated and the model <u>selection</u> part becomes central

What can machines learn?

Machine learning, artificial intelligence and deep learning had an enormous success in recent years

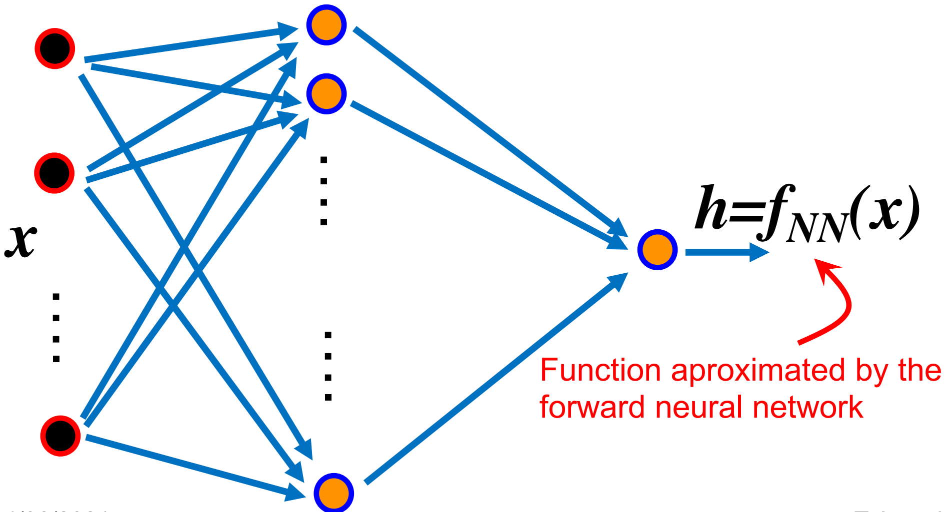
Their successful application has been mainly in the domain of image recognition/manipulation and games



Universal approximation theorem

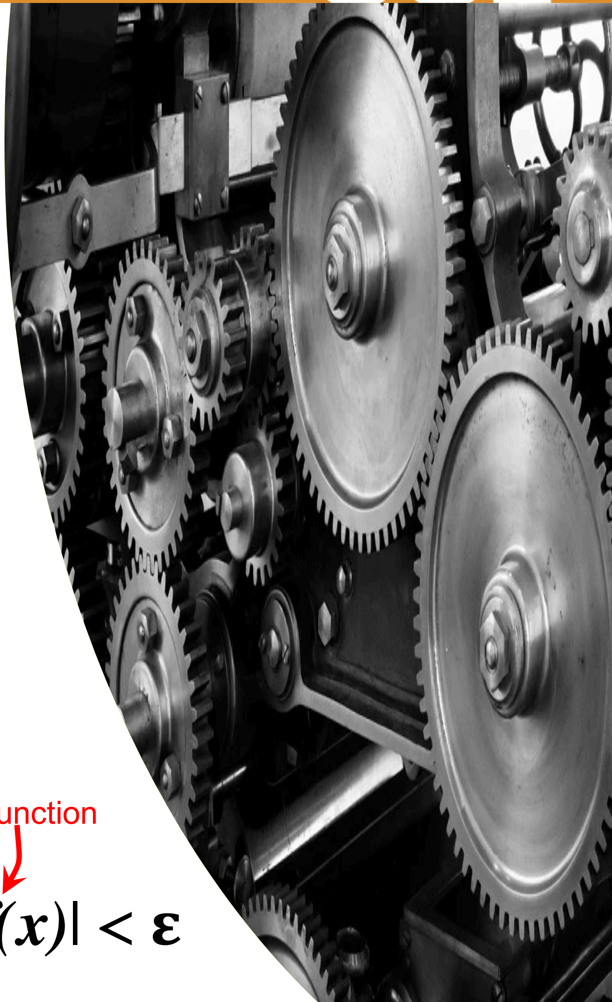
Cybenko 1989, Hornik 1991

A forward neural network with more than one layer can approximate any function as far as the network has a large enough number of neurons



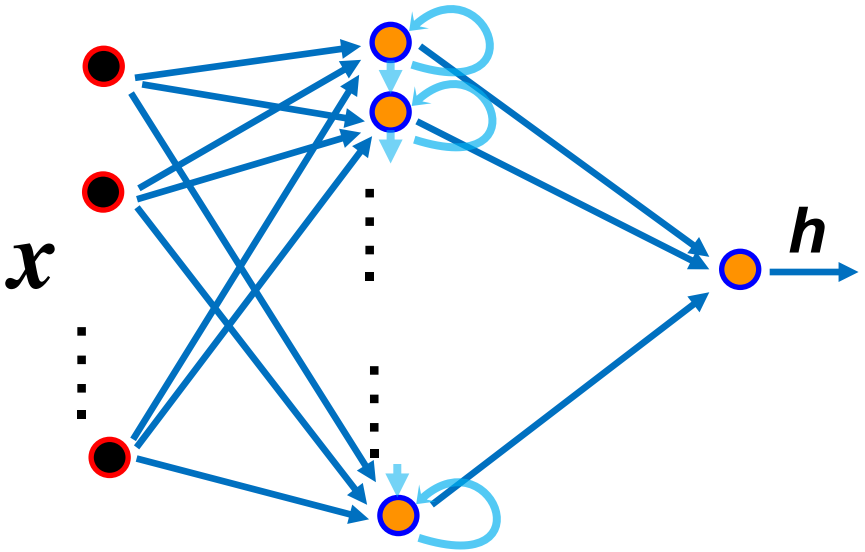
Reference function

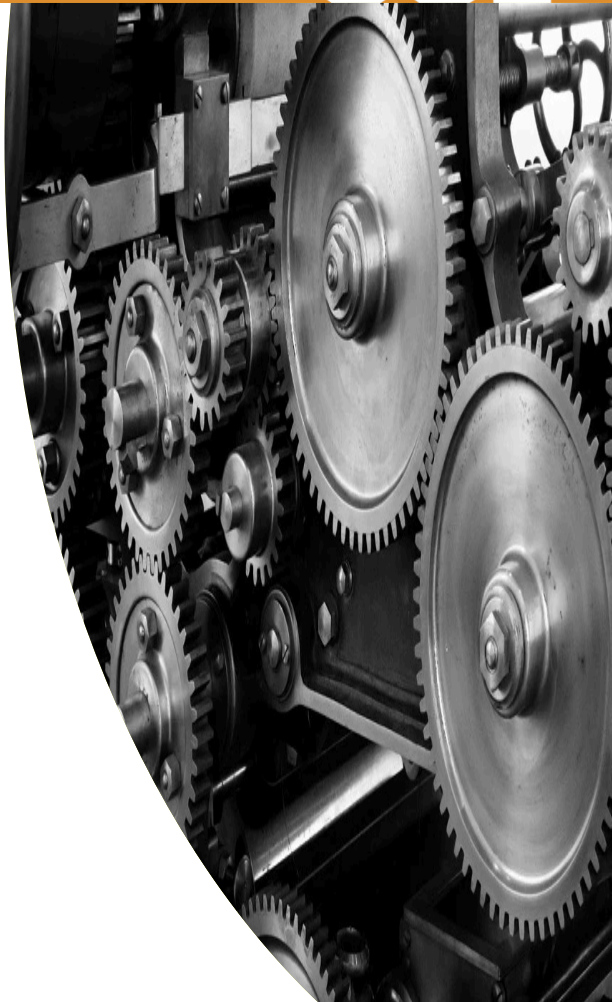
$$|f_{NN}(x) - f(x)| < \varepsilon$$



Recurrent neural networks are Turing complete

By adding loops (within a layer or backwards) a recurrent neural network is Turing complete and therefore it can perform any computation





Can machines learn markets?

Markets

are gigantic computation environments where machines and humans interact to calculate the price of things

Mercati

Markets are complex systems

- Markets are complex systems where a large and heterogeneous number of variables interact within an intricated system of relations
- In markets trades are executed at speeds ranging from nanoseconds to years, this are 10^{15} order of magnitude (comparable to our distance to Proxima centaury in meters)
- Market variables have statistical properties that are non-normal with fat-tails power law distributions
- Markets adapt and change, they are not stationary

Can machines learn markets?

Has a machine that can trade successfully learned something about the market?

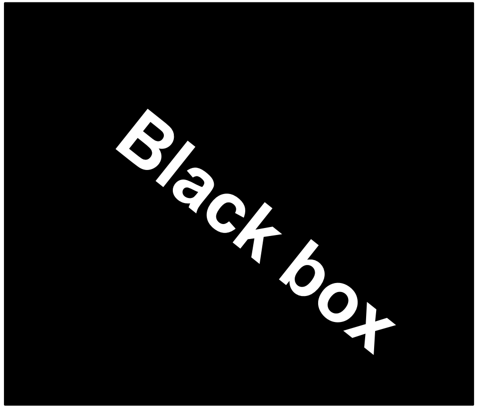
Can we define what does it mean learn in the case of markets?

A <u>pragmatic</u> perspective:

can machines learn to automate tasks so far performed by humans?

What can we learn from machines?

Black boxes?



Deep learning models return equations or algorithms; these are the same kind of outputs of human-made models

They are however extremely complicated being

- **high-dimensional and**
- **non-linear**

High dimensionality

High dimensional spaces are very different from the low-dimensional space we live in.

In particular, the subdivision of high-dimensional spaces into regions (the basins of optimal solutions) have non-intuitive properties

- Most of the volume of the region is near the surface
- The number of neighboring regions grows at least exponentially with dimension
- The number of interfaces between regions grow combinatorically with dimension
- Any 'gradient descent optimizer' will be always and unavoidably stuck in some saddle-point frontier

Tomaso Aste and , Denis Weaire, *The pursuit of perfect packing*. CRC Press, 2008.
T. Aste, and N. Rivier. "Random cellular froths in spaces of any dimension and curvature." Journal of Physics A: Mathematical and General 28, no. 5 (1995): 1381.
T Aste, "Dynamical partitions of space in any dimension." Journal of Physics A: Mathematical and General 31, no. 43 (1998): 8577.

Linearity

Linear solutions of linear problems are neat and unique, they have convenient properties:

1. Small changes in the input produce proportionally small changes in the output
2. Small changes in the solution structure or parameters also produce small changes in the output
3. Approximate solutions with similar errors are similar

Non-linear solutions are very different

Non-linearity

Non-linear solutions solutions are very different

1. Small changes in the input can produce very large changes in the output
2. Small changes in the solution structure or parameters can produce very large changes in the output
3. Approximate solutions with similar errors can be completely different and there is a combinatorial large number of them

Intriguingly, the way deep learning systems are trained (specifically methods such as: data sampling, drop off, knockout, data augmentations, loss functions, weight regularization…) seem to overcome several of these issues, at least in some cases. We have a lot of to learn about how these systems discover approximate solutions

What machines can learn about our complex world - and what can we learn from them?

They learn approximate solutions that are high-dimensional and non-linear

The structure of the solution tells us very little about the model

However, if the learning process if done properly, these solutions are quite robust and can work also in circumstances different from the training examples

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3797711

Deep learning the limit Order Book

Experiment 1

A comparative perspective

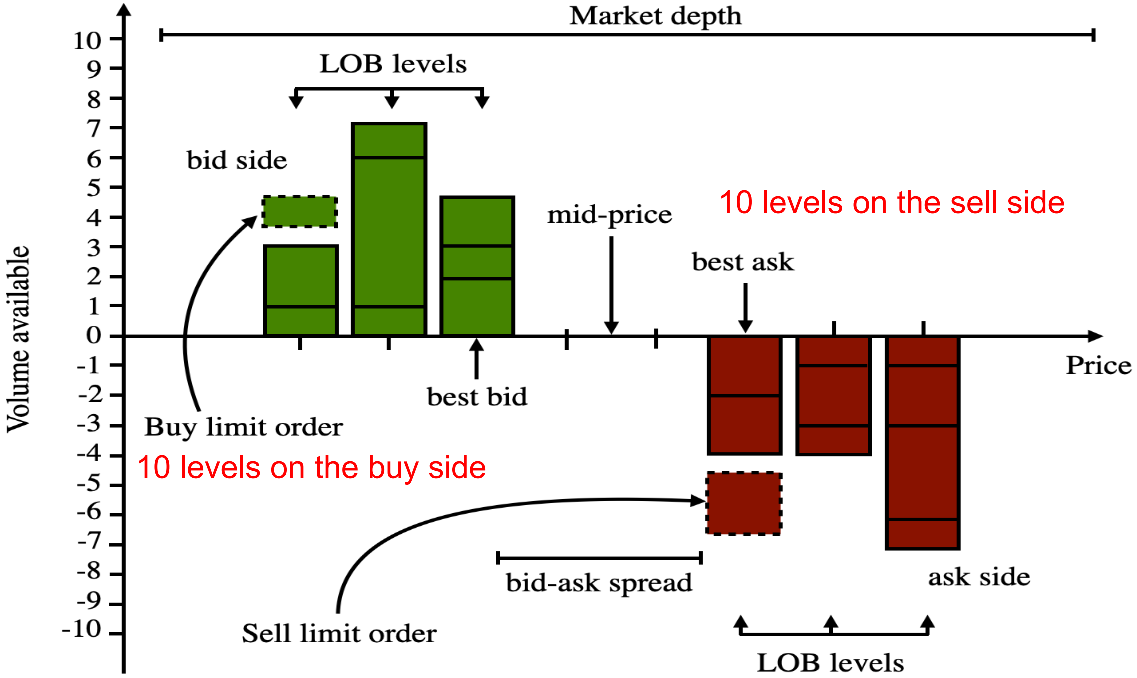
https://arxiv.org/pdf/2007.07319.pdf


Jeremy Turiel


Antonio Briola

The limit order book (LOB)

The Limit Order Book (LOB) is a self-organizing system where a large number of players interact with offers and bids and eventually agree on a transaction price



About 10 transactions per second for liquid assets on NASDAQ

The machine learning system

INPUT

LOB prices and volumes for 10 previous ticks

400-dimesions

Machine learning system (7 models)

Testing with 6 million datapoints

OUTPUT

Transaction price range at a given horizon

3-dimensions

Training with 18 million examples

Input

$$[(p,v)_0^a, (p,v)_0^b, (p,v)_1^a, (p,v)_1^b, \ \dots, (p,v)_{10}^a, (p,v)_{10}^b]$$

20 levels of LOB

x_t = 40-dimensions

10 previous ticks ***(x_{t-9},….x_t)***

Overall a vector of 400-dimesions

Data:

NASDAQ, Intel Corp. (INTC) LOB data

Period:

training 3 months:

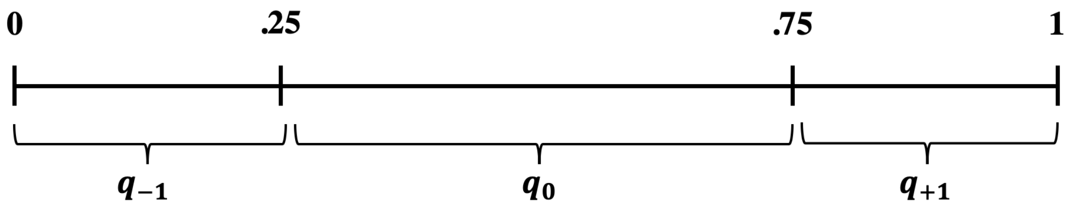
04-02-2019 to 31-05-2019 -> ~ 18 million transactions

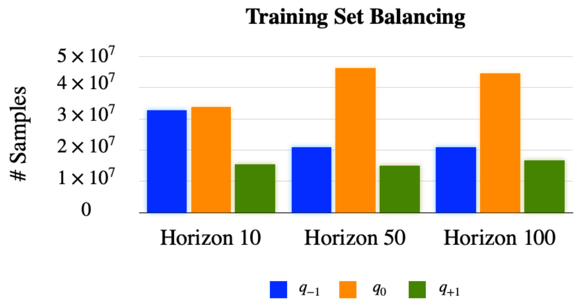
testing 1 month:

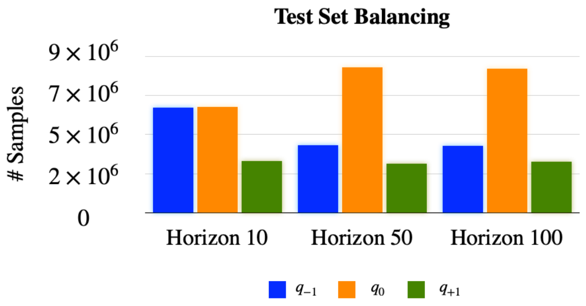
03-06-2019 to 28-06-2019 -> ~ 6 million transactions

Output

Probability of transaction after 10, 50 or 100 ticks at a price within a given quantile range established from the training set

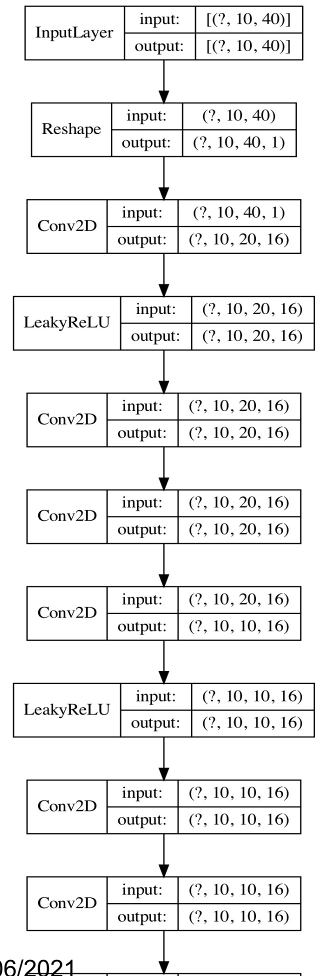




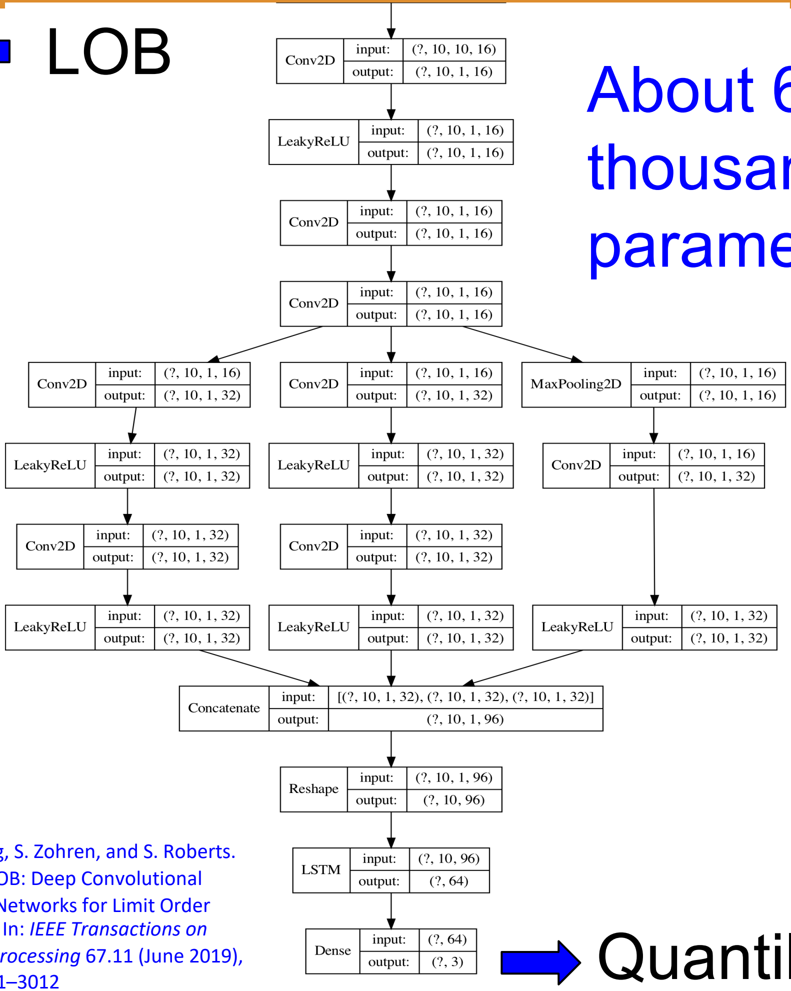


Horizons
10, 50, 100 ticks

CNN-LSTM



LOB



About 60 thousands parameters

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 10, 40)]	0	
reshape (Reshape)	(None, 10, 40, 1)	0	input_1[0][0]
conv2d (Conv2D)	(None, 10, 20, 16)	48	reshape[0][0]
leaky_re_lu (LeakyReLU)	(None, 10, 20, 16)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 10, 20, 16)	1040	leaky_re_lu[0][0]
conv2d_2 (Conv2D)	(None, 10, 20, 16)	1040	conv2d_1[0][0]
conv2d_3 (Conv2D)	(None, 10, 10, 16)	528	conv2d_2[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 10, 10, 16)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 10, 10, 16)	1040	leaky_re_lu_1[0][0]
conv2d_5 (Conv2D)	(None, 10, 10, 16)	1040	conv2d_4[0][0]
conv2d_6 (Conv2D)	(None, 10, 1, 16)	2576	conv2d_5[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 10, 1, 16)	0	conv2d_6[0][0]
conv2d_7 (Conv2D)	(None, 10, 1, 16)	1040	leaky_re_lu_2[0][0]
conv2d_8 (Conv2D)	(None, 10, 1, 16)	1040	conv2d_7[0][0]
conv2d_9 (Conv2D)	(None, 10, 1, 32)	544	conv2d_8[0][0]
conv2d_11 (Conv2D)	(None, 10, 1, 32)	544	conv2d_8[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 10, 1, 32)	0	conv2d_9[0][0]
leaky_re_lu_5 (LeakyReLU)	(None, 10, 1, 32)	0	conv2d_11[0][0]
max_pooling2d (MaxPooling2D)	(None, 10, 1, 16)	0	conv2d_8[0][0]
conv2d_10 (Conv2D)	(None, 10, 1, 32)	3104	leaky_re_lu_3[0][0]
conv2d_12 (Conv2D)	(None, 10, 1, 32)	5152	leaky_re_lu_5[0][0]
conv2d_13 (Conv2D)	(None, 10, 1, 32)	544	max_pooling2d[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 10, 1, 32)	0	conv2d_10[0][0]
leaky_re_lu_6 (LeakyReLU)	(None, 10, 1, 32)	0	conv2d_12[0][0]
leaky_re_lu_7 (LeakyReLU)	(None, 10, 1, 32)	0	conv2d_13[0][0]
concatenate (Concatenate)	(None, 10, 1, 96)	0	leaky_re_lu_4[0][0] leaky_re_lu_6[0][0] leaky_re_lu_7[0][0]
reshape_1 (Reshape)	(None, 10, 96)	0	concatenate[0][0]
lstm (LSTM)	(None, 64)	41216	reshape_1[0][0]
dense (Dense)	(None, 3)	195	lstm[0][0]

Total params: 60,691
Trainable params: 60,691
Non-trainable params: 0

Z. Zhang, S. Zohren, and S. Roberts. “DeepLOB: Deep Convolutional Neural Networks for Limit Order Books”. In: *IEEE Transactions on Signal Processing* 67.11 (June 2019), pp. 3001–3012

Quantile probability

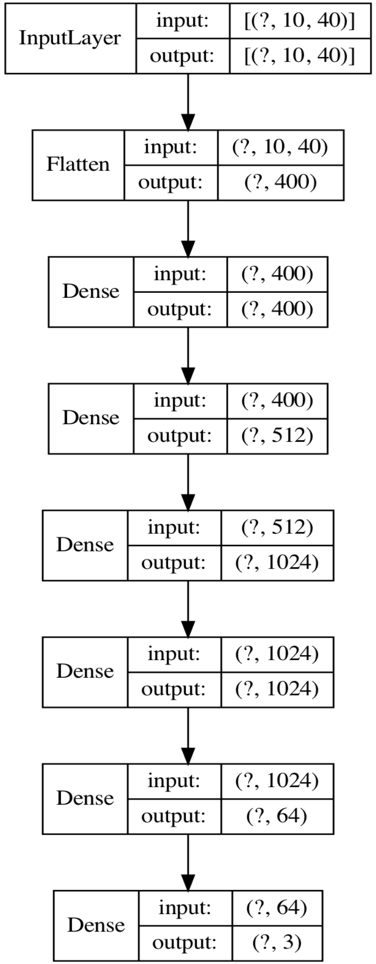
Models

Can a Multilayer Perceptron learn the LOB efficiently?
And how will it do with respect to other simpler and more complicated models?

We investigate and compare 7 different models with increasing levels of complexity

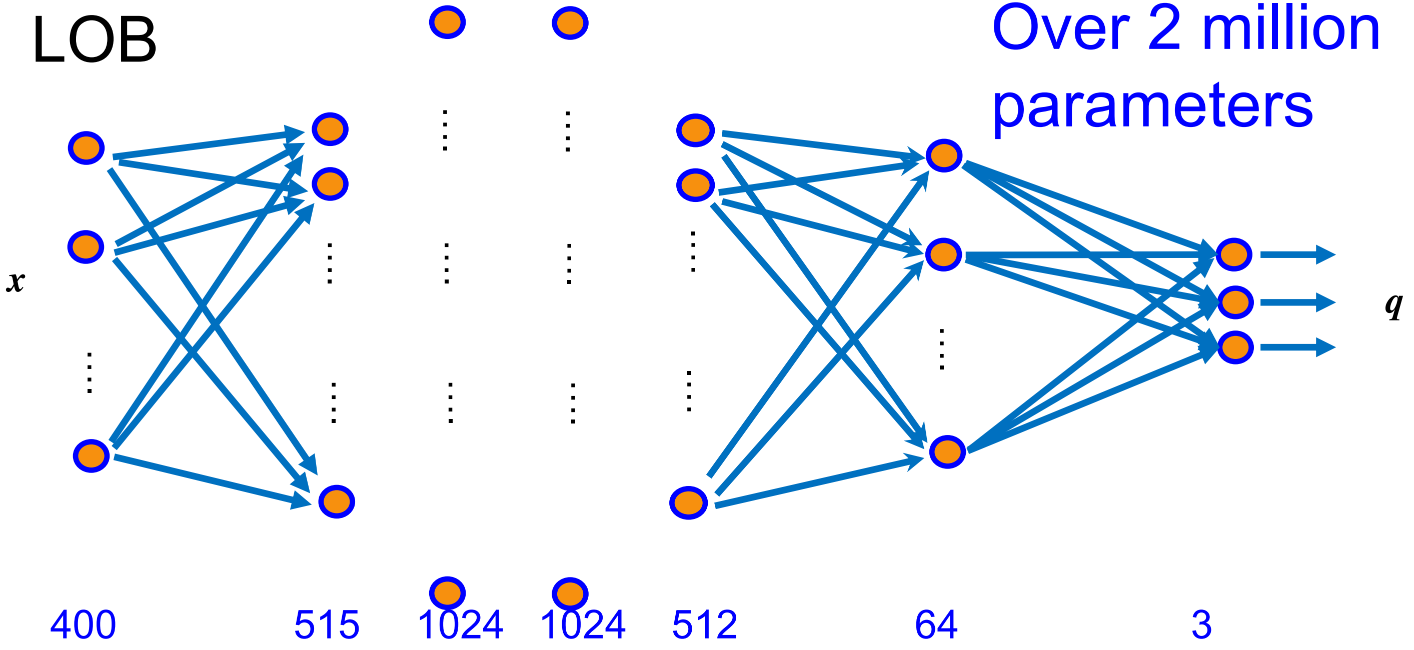
1. **Random Model** – uniform probability outcome prediction
2. **Naive Model** – output most represented in training
3. **Logistic Regression** – simple perceptron
4. **Multilayer Perceptron** – deep learning model
5. **Shallow LSTM** – deep learning with memory
6. **Self-Attention LSTM** – deep learning with memory & loop
7. **CNN-LSTM** – state of the art deep learning

Multilayer perceptron



LOB

Over 2 million parameters

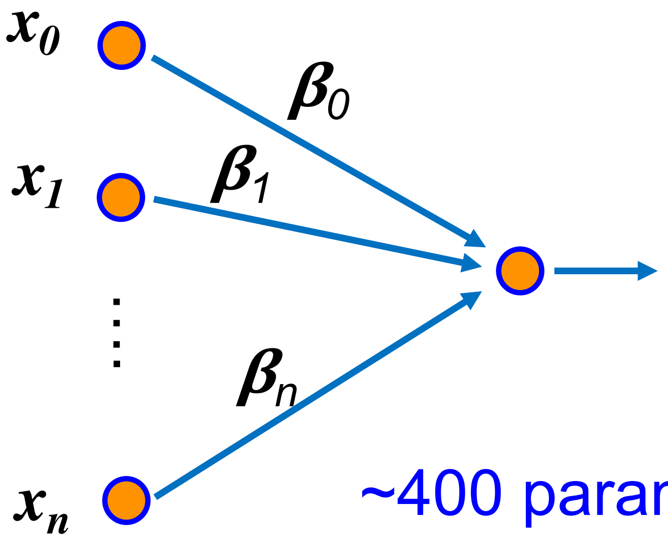


Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 10, 40)]	0
flatten (Flatten)	(None, 400)	0
dense (Dense)	(None, 400)	160400
dense_1 (Dense)	(None, 512)	205312
dense_2 (Dense)	(None, 1024)	525312
dense_3 (Dense)	(None, 1024)	1049600
dense_4 (Dense)	(None, 64)	65600
dense_5 (Dense)	(None, 3)	195

Total params: 2,006,419. Trainable params: 2,006,419. Non-trainable params: 0

Quantile probability

Logistic regression

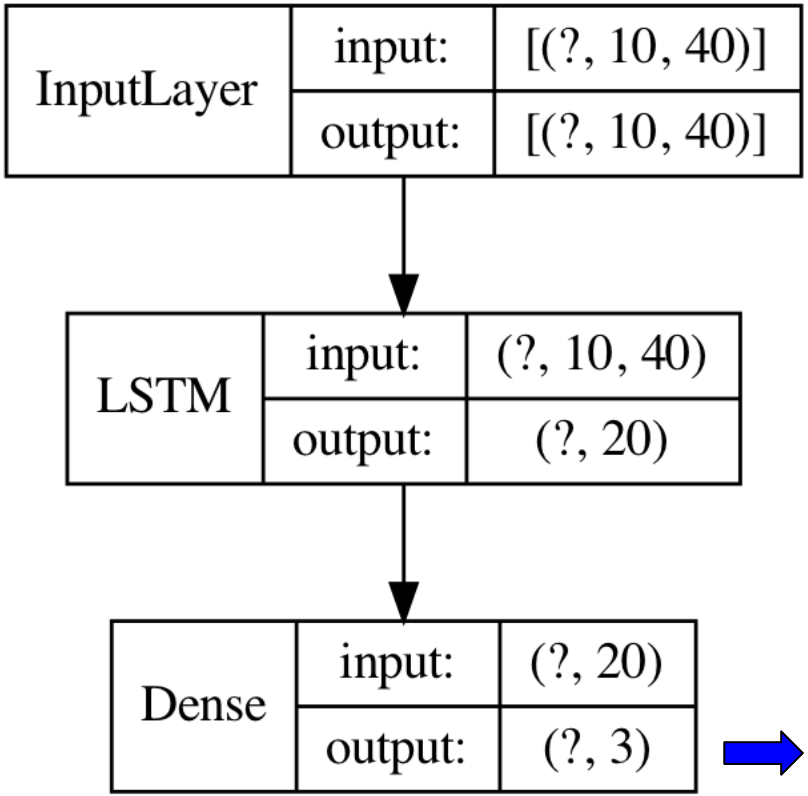


$$p = \frac{e^{\sum_i \beta_i x_i}}{1 + e^{\sum_i \beta_i x_i}}$$

~400 parameters

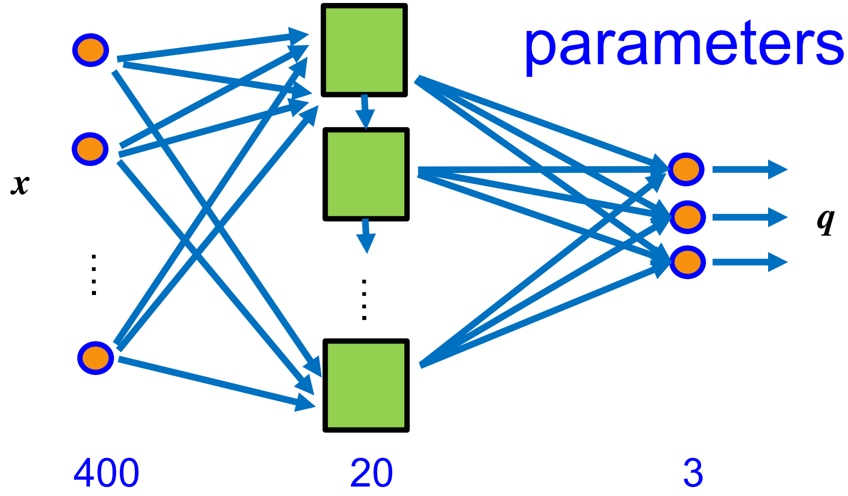
One-layer perceptron ($x_0=1$)

Shallow LSTM



LOB

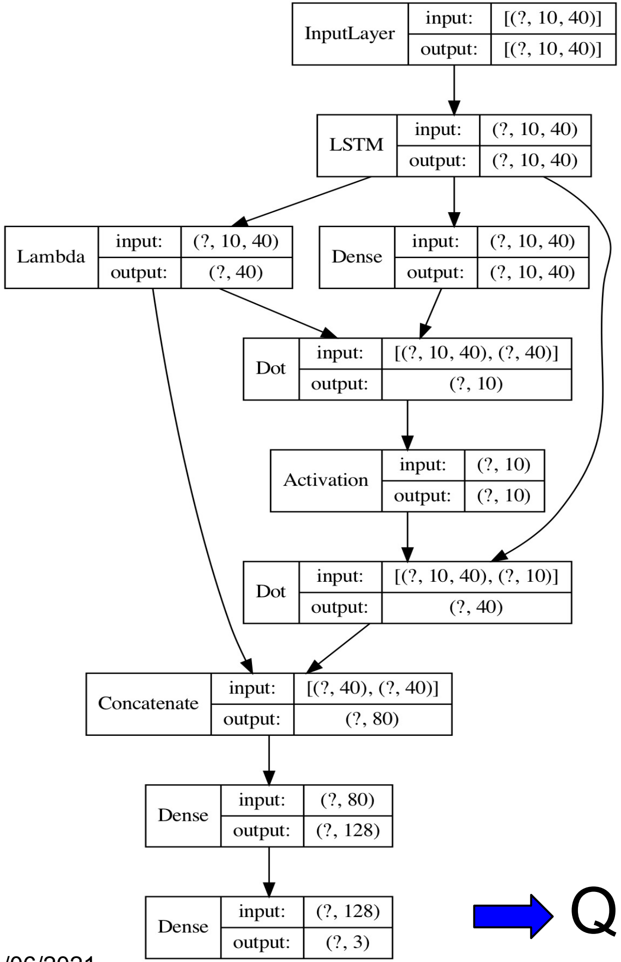
Quantile probability



About 5 thousands parameters

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10, 40)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 20)                4880
_________________________________________________________________
dense (Dense)                (None, 3)                 63
=================================================================
Total params: 4,943
Trainable params: 4,943
Non-trainable params: 0
_________________________________________________________________
```

Self-Attention LSTM –



LOB

Quantile probability

About 25 thousands parameters

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 10, 40)]	0	
lstm (LSTM)	(None, 10, 40)	12960	input_1[0][0]
attention_score_vec (Dense)	(None, 10, 40)	1600	lstm[0][0]
last_hidden_state (Lambda)	(None, 40)	0	lstm[0][0]
attention_score (Dot)	(None, 10)	0	attention_score_vec[0][0] last_hidden_state[0][0]
attention_weight (Activation)	(None, 10)	0	attention_score[0][0]
context_vector (Dot)	(None, 40)	0	lstm[0][0] attention_weight[0][0]
attention_output (Concatenate)	(None, 80)	0	context_vector[0][0] last_hidden_state[0][0]
attention_vector (Dense)	(None, 128)	10240	attention_output[0][0]
dense (Dense)	(None, 3)	387	attention_vector[0][0]

Total params: 25,187
Trainable params: 25,187
Non-trainable params: 0

Model Details

The models range a parameter space dimension from zero to one million

Model	Input	Number of layers $^{(*)}$	Number of parameters$^{(**)}$	Learning rate	Training epochs
Random	-	0	0	–	-
Naive	LOB	1	1	–	-
Logistic Reg.	LOB	2	4×10^2	10^{-3}	30
MLP	LOB	7	2.0×10^6	10^{-3}	30
Shallow LSTM	LOB	3	4.9×10^3	10^{-3}	30
Self-Attention LSTM	LOB	4	2.5×10^4	10^{-3}	30
CNN-LSTM	LOB	28	6.0×10^4	10^{-3}	30

Table 1: Summary of the inputs and hyperparameters used in the models in this article. $^{(*)}$The number of layers includes the input and the output layer. $^{(**)}$The number of parameters is approximated to the nearest order of magnitude and truncated for readability.

Performance metrics

We tested performances of the prediction for each of the quantile regions by computing:
Precision, **Recall** and **F-measure**

In order to correct for class imbalance we also tested:
balanced Accuracy, weighted Precision, weighted Recall and **weighted F-score**

Furthermore, two multi-class correlation metrics between forecasted and real labels computed:
Matthews Correlation Coefficient (MCC) and **Cohen's Kappa**

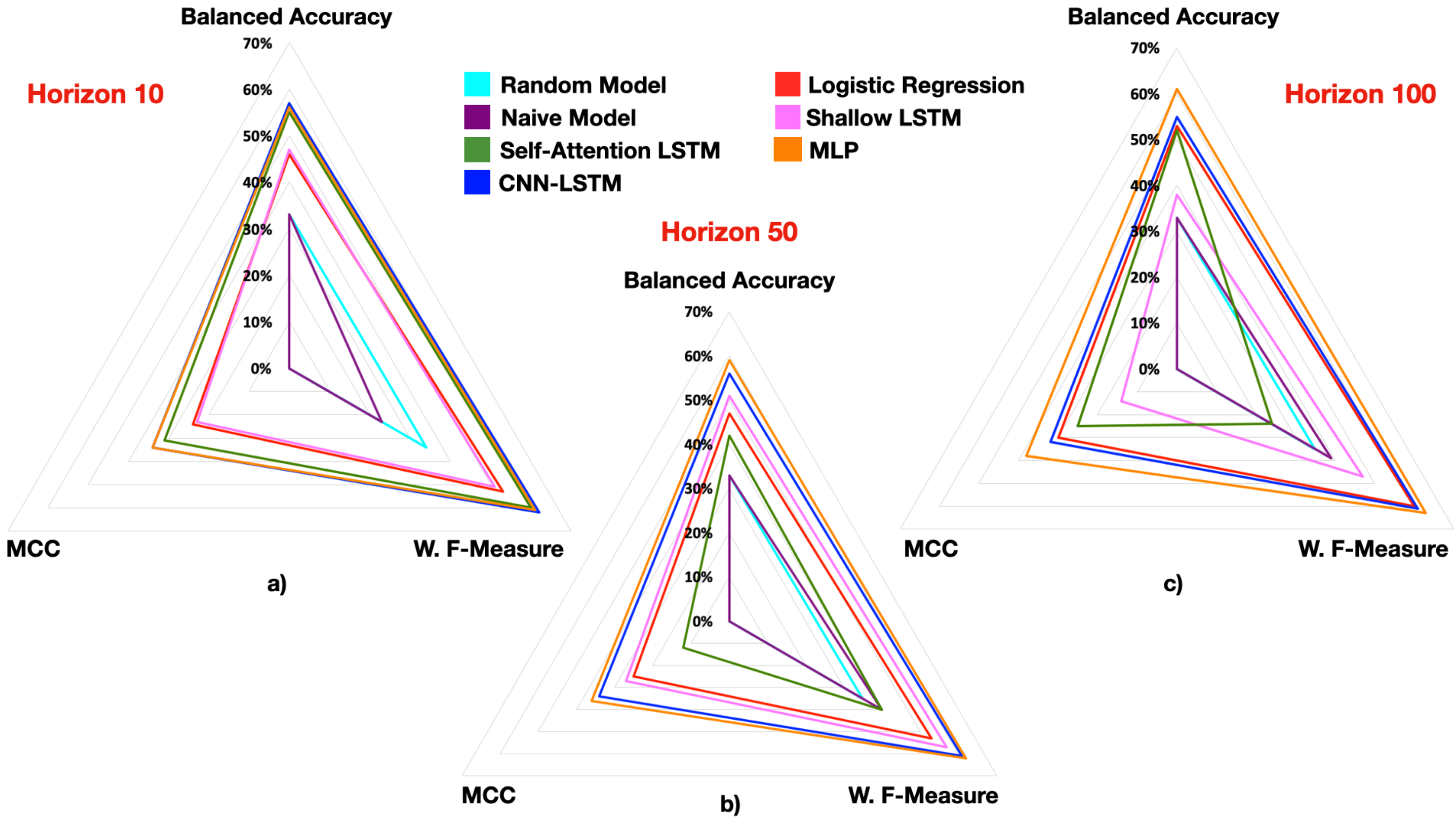
Results

Performances at three horizons: 10, 50, 100 ticks

	Random Model			Naive Model			Logistic Regression			Shallow LSTM			Self-Attention LSTM			CNN-LSTM			Multilayer Perceptron		
	H10	**H50**	**H100**	**H10**	**H50**	**H100**	**H10**	**H50**	**H100**	**H10**	**H50**	**H100**	**H10**	**H50**	**H100**	**H10**	**H50**	**H100**	**H10**	**H50**	**H100**
Balanced Accuracy	0.33	0.33	0.33	0.33	0.33	0.33	0.46	0.47	0.53	0.47	0.51	0.38	0.55	0.42	0.52	0.57	0.56	0.55	0.56	0.59	0.61
Weighted Precision	0.41	0.41	0.41	0.16	0.30	0.30	0.54	0.56	0.62	0.58	0.57	0.65	0.61	0.50	0.47	0.62	0.61	0.61	0.62	0.62	0.63
Weighted Recall	0.33	0.33	0.33	0.40	0.55	0.54	0.59	0.59	0.61	0.58	0.58	0.57	0.61	0.45	0.34	0.62	0.62	0.62	0.62	0.63	0.63
Weighted F-Measure	0.34	0.35	0.35	0.23	0.40	0.39	0.53	0.53	0.60	0.51	0.57	0.47	0.60	0.40	0.24	0.62	0.61	0.61	0.61	0.62	0.63
Precision quantile [0, 0.25]	0.26	0.26	0.26	0	0	0	0.57	0.57	0.57	0.57	0.56	0.58	0.60	0.37	0.55	0.59	0.59	0.59	0.59	0.59	0.59
Precision quantile [0.25, 0.75]	0.55	0.55	0.55	0.40	0.55	0.54	0.59	0.60	0.63	0.59	0.62	0.56	0.62	0.55	0.52	0.65	0.64	0.63	0.64	0.66	0.67
Precision quantile [0.75, 1]	0.20	0.20	0.20	0	0	0	0.31	0.38	0.58	0.57	0.43	0.97	0.57	0.52	0.26	0.57	0.57	0.57	0.59	0.58	0.57
Recall quantile [0, 0.25]	0.33	0.33	0.33	0	0	0	0.55	0.59	0.59	0.06	0.62	0.22	0.35	0.81	0.62	0.54	0.55	0.53	0.60	0.59	0.60
Recall quantile [0.25, 0.75]	0.33	0.33	0.33	1	1	1	0.82	0.81	0.76	0.85	0.69	0.93	0.76	0.44	0.01	0.71	0.72	0.74	0.74	0.70	0.68
Recall quantile [0.75, 1]	0.33	0.33	0.33	0	0	0	0	0	0.23	0.51	0.23	0	0.53	0.004	0.92	0.46	0.42	0.39	0.34	0.46	0.54
F-Measure quantile [0, 0.25]	0.29	0.29	0.29	0	0	0	0.56	0.57	0.58	0.11	0.59	0.31	0.44	0.503	0.58	0.57	0.57	0.56	0.59	0.59	0.59
F-Measure quantile [0.25, 0.75]	0.42	0.42	0.42	0.57	0.71	0.70	0.70	0.70	0.68	0.69	0.65	0.70	0.68	0.489	0.02	0.68	0.68	0.68	0.69	0.68	0.67
F-Measure quantile [0.75, 1]	0.25	0.25	0.25	0	0	0	0	0	0.33	0.54	0.30	0	0.55	0.009	0.40	0.51	0.48	0.46	0.43	0.51	0.56
MCC	0	0	0	0	0	0	0.24	0.25	0.30	0.23	0.27	0.14	0.31	0.120	0.25	0.34	0.34	0.32	0.34	0.36	0.38
Cohen's Kappa	0	0	0	0	0	0	0.21	0.22	0.30	0.20	0.27	0.10	0.30	0.105	0.16	0.34	0.33	0.32	0.33	0.36	0.38

Best

Results


Horizon 10
Balanced Accuracy
70%
60%
50%
40%
30%
20%
10%
0%
MCC
W. F-Measure
a)
Random Model
Naive Model
Self-Attention LSTM
CNN-LSTM
Logistic Regression
Shallow LSTM
MLP
Horizon 50
Balanced Accuracy
MCC
W. F-Measure
b)
Horizon 100
Balanced Accuracy
MCC
W. F-Measure
c)

Results

Bayesian correlated t-test from MCC measure

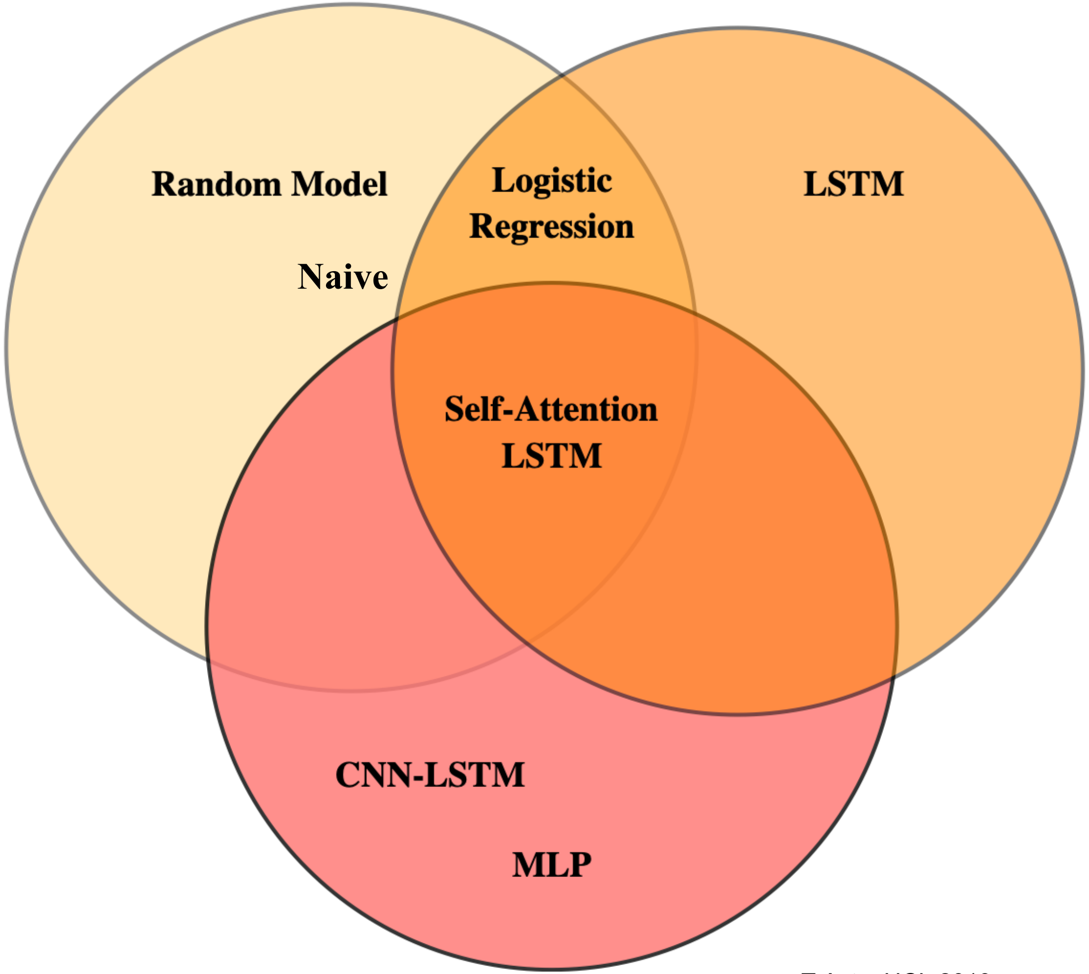
$\mathbf{H}_{\Delta\tau}\vert\Delta\tau = 10$	$\mathbf{H}_{\Delta\tau}\vert\Delta\tau = 50$	$\mathbf{H}_{\Delta\tau}\vert\Delta\tau = 100$
Multilayer Perceptron CNN - LSTM	Multilayer Perceptron CNN - LSTM	Multilayer Perceptron CNN - LSTM
Self-Attention LSTM	Shallow LSTM Multinomial Logistic Regression	Multinomial Logistic Regression
Shallow LSTM Multinomial Logistic Regression	Self-Attention LSTM	Self-Attention LSTM
Naive Model Random Model	Naive Model Random Model	Shallow LSTM
		Random Model

Bayesian correlated t-test from F measure

$\mathbf{H}_{\Delta\tau}\vert\Delta\tau = 10$	$\mathbf{H}_{\Delta\tau}\vert\Delta\tau = 50$	$\mathbf{H}_{\Delta\tau}\vert\Delta\tau = 100$
Multilayer Perceptron CNN - LSTM Self-Attention LSTM	Multilayer Perceptron CNN - LSTM	Multilayer Perceptron
Shallow LSTM Multinomial Logistic Regression	Shallow LSTM	CNN - LSTM Multinomial Logistic Regression
Naive Model	Multinomial Logistic Regression	Shallow LSTM
Random Model	Self-Attention LSTM	Self-Attention LSTM
	Naive Model	Naive Model
	Random Model	Random Model

- Multilayer perceptron is the best performing
- CNN-LSTM is second-best with comparable performances
- Logistic regression has good performances comparable with LSTM and self-attention LSTM
- Naïve and Random are worst

Results



One can attempt to cluster the models together based on their performances

One can note that memory/recurrent loops (LSTM) play little role in performances. Most of the information is processed forwards from the LOB input

Deep learning the limit Order Book

Experiment 2

Reinforcement Learning

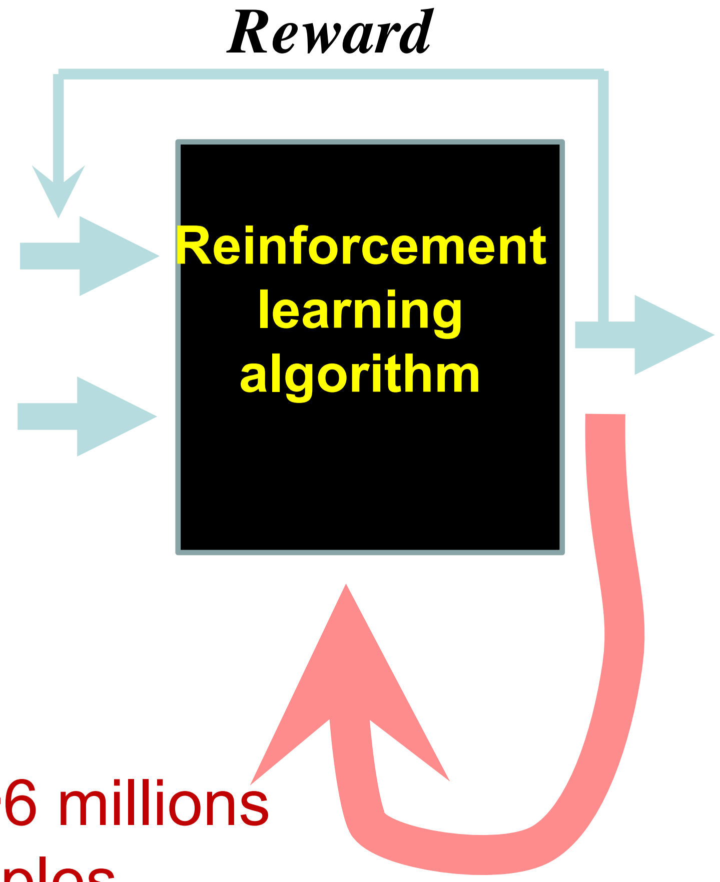
https://arxiv.org/abs/2101.07107

Testing with 6 million examples

INPUT

LOB prices and volumes for 10 previous ticks

400-dimesions



OUTPUT

Action:

1. Sell,
2. stay,
3. buy,
4. stop_loss

Training with ~6 millions selected examples

Deep reinforcement learning: architecture

The algorithm buys or sells or holds one unit of Intel Corporation stock (INTC) on NASDAQ during the month of June 2019

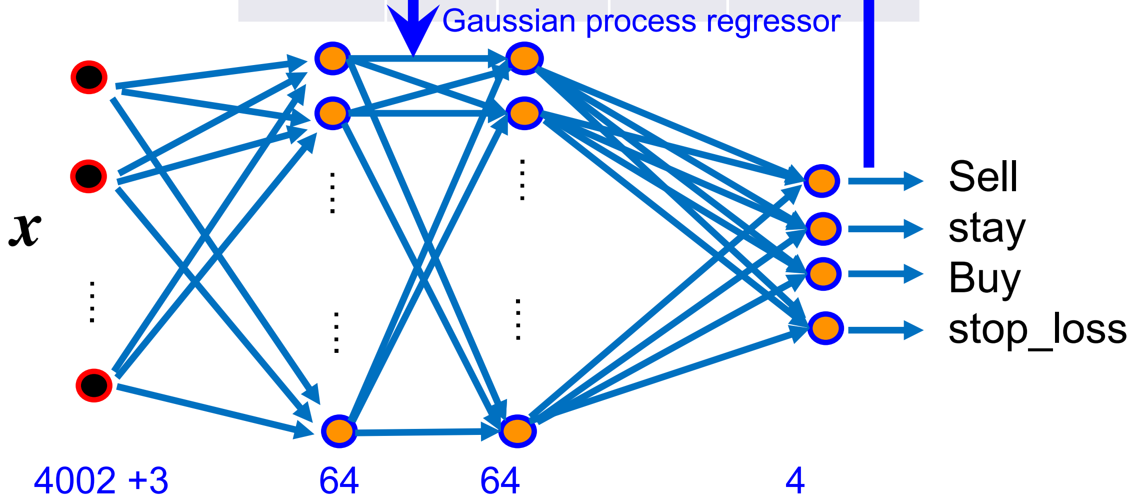
It is trained during the previous 3 months

Q-learning approach

Profits of the 'agent' depending on the action

Q	sell	stay	buy	Stop_loss
neutral				
Long				
short				

Reward



About 30,000 parameters

Deep reinforcement learning: training

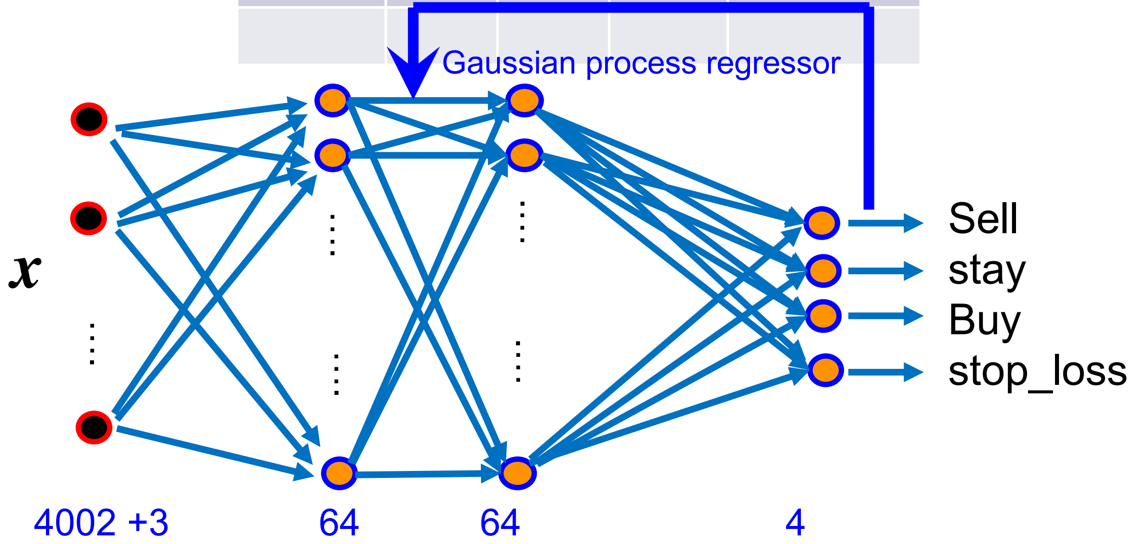
Profits of the 'agent' depending on the action

Q	sell	stay	buy	Stop_loss
neutral				
Long				
short				

Reward

We train on 25 selected significant events containing about **250,000** ticks

- Training on 60 days 04/02/2019-30/04/2019
- Validation on 22 days 01/05/2019-31/05/2019
- Testing on 20 days 03/06/2019-28/06/2019



About 30,000 parameters

Deep reinforcement learning: input

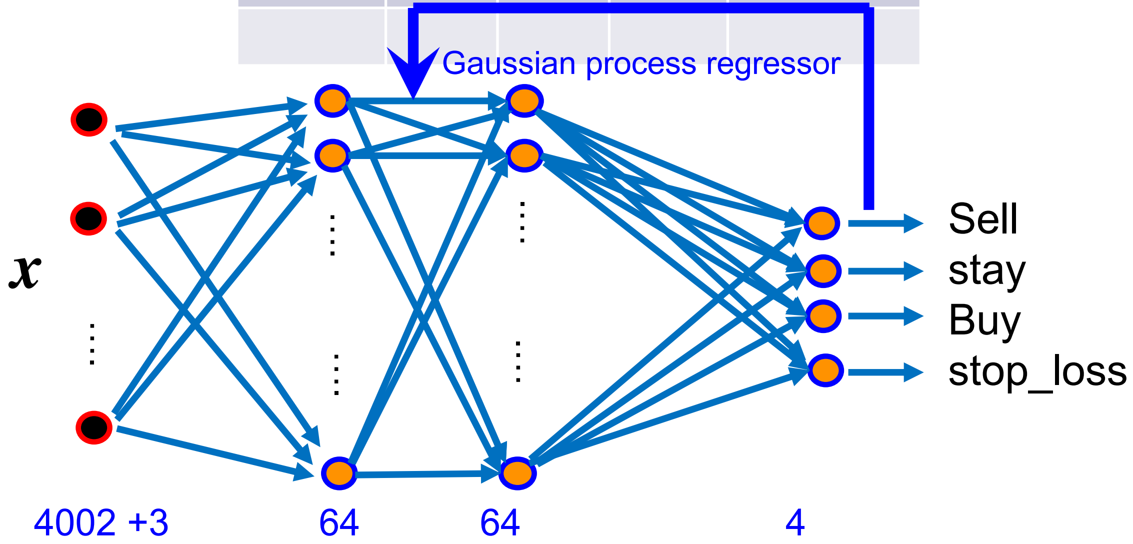
We test for three sets of input information; all have the full LOB (400 dimension) plus:

1. State of the agent
2. State of the agent & market price minus price paid for the unit (mark to market profit)
3. State of the agent & price paid for the unit (mark to market) & bid-ask spread

Profits of the 'agent' depending on the action

Q	sell	stay	buy	Stop_loss
neutral				
Long				
short				

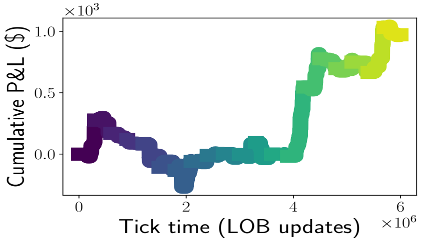
Reward

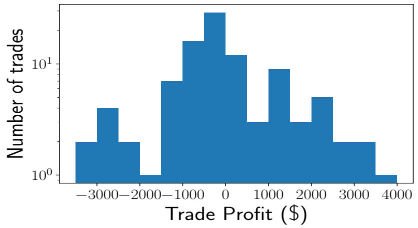


About 30,000 parameters

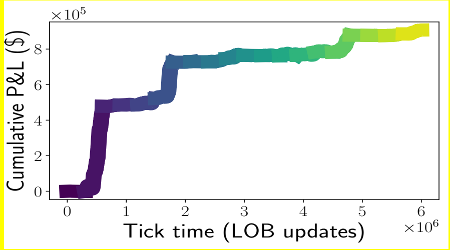
Deep reinforcement learning: performances

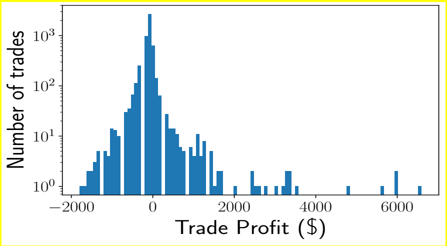
1. LOB +
State of the agent



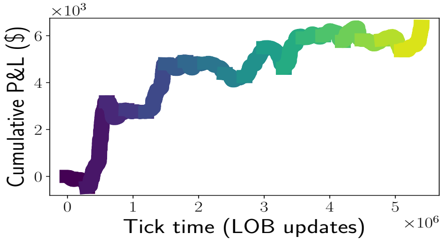


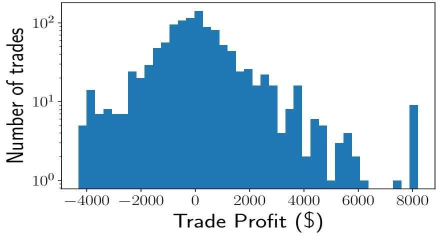
2. LOB +
State of the agent &
price paid for the unit
(mark to market)





3. LOB +
State of the agent & price paid
for the unit (mark to market) &
bid-ask spread





Deep reinforcement learning: performances

What the agent learned?

- The agent increases profits by about 100 folds by trading 10 to 100 times more often using information about the reference unit price (mark to market). The agent learns to increase profits while increasing trading frequency despite the bid-ask spread cost
- Risk is reduced considerably
- The extra information on the bid-ask spread does not increase performances

Conclusions

- Artificial intelligence is providing increasingly powerful new instruments
- It is almost a surprise that a complicated self-trained machine, such as the Multi Layer Perception, can learn something about the price formation mechanism on the LOB
- It is almost a surprise that a self-learning agent can discover trading strategies
- Results are very good but not ground-breaking. Are we at the beginning or at the end of this journey?

Links and references

LINKS

FCA Group Page:
http://fincomp.cs.ucl.ac.uk/introduction/

My articles:
https://scholar.google.co.uk/citations?user=27pUbTUAAAAJ&hl=en

Software:
TMFG & Clique Forests
https://github.com/cvborkulo/NetworkComparisonTest/pull/5
https://uk.mathworks.com/matlabcentral/fileexchange/56444-tmfg

RELEVANT PAPERS

Aste, Tomaso. "What machines can learn about our complex world-and what can we learn from them?." Available at SSRN 3797711 (2021).

Briola, Antonio, Jeremy Turiel, Riccardo Marcaccioli, and Tomaso Aste. "Deep Reinforcement Learning for Active High Frequency Trading." arXiv preprint arXiv:2101.07107 (2021).

Briola, Antonio, Jeremy Turiel, and Tomaso Aste. "Deep Learning Modeling of the Limit Order Book: A Comparative Perspective." Available at SSRN 3714230 (2020).