Reinforcement Learning for Trading

Dr Tom Starke

About

- PhD in Physics
- Author of 20 journal papers and 3 patents
- Lecturer at Oxford uni
- Principal engineer at Rolls Royce
- Quant developer at numerous hedge funds and prop firms
- Now CEO of AAAQuants

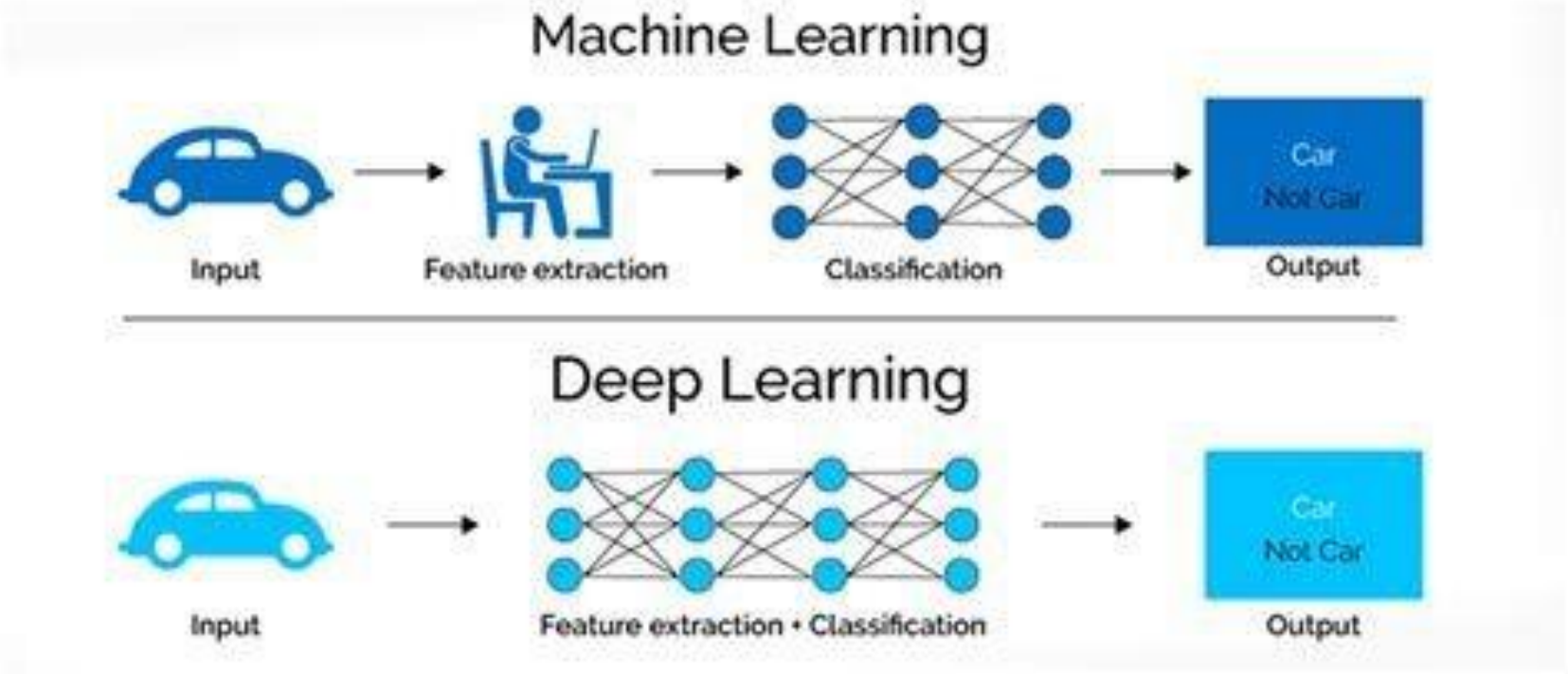
Wouldn’t it be great to make money while you sleep?

Big move from manual trading to algorithms

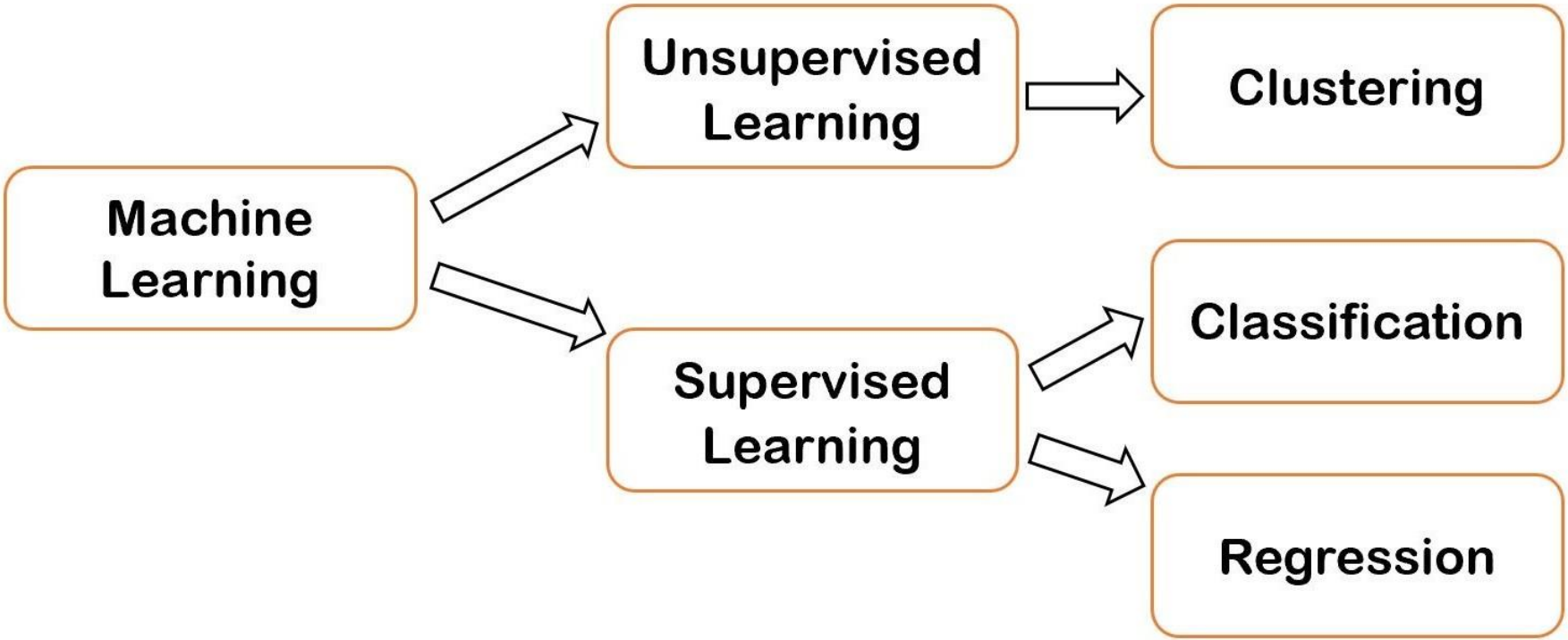
- Cheaper computers
- Cheaper and better data
- Better online resources

What kinds of algorithms could help us making automated trading systems possible?

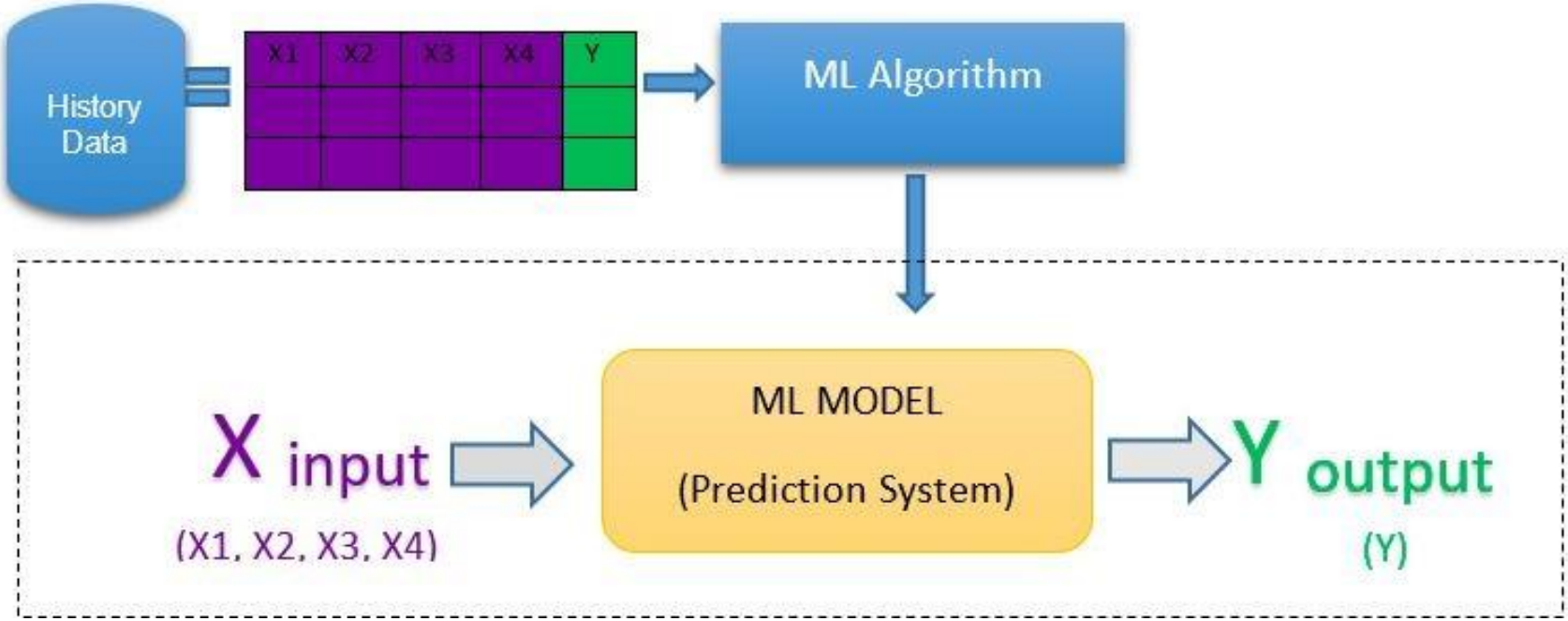
Let’s take a step back: what is machine learning?



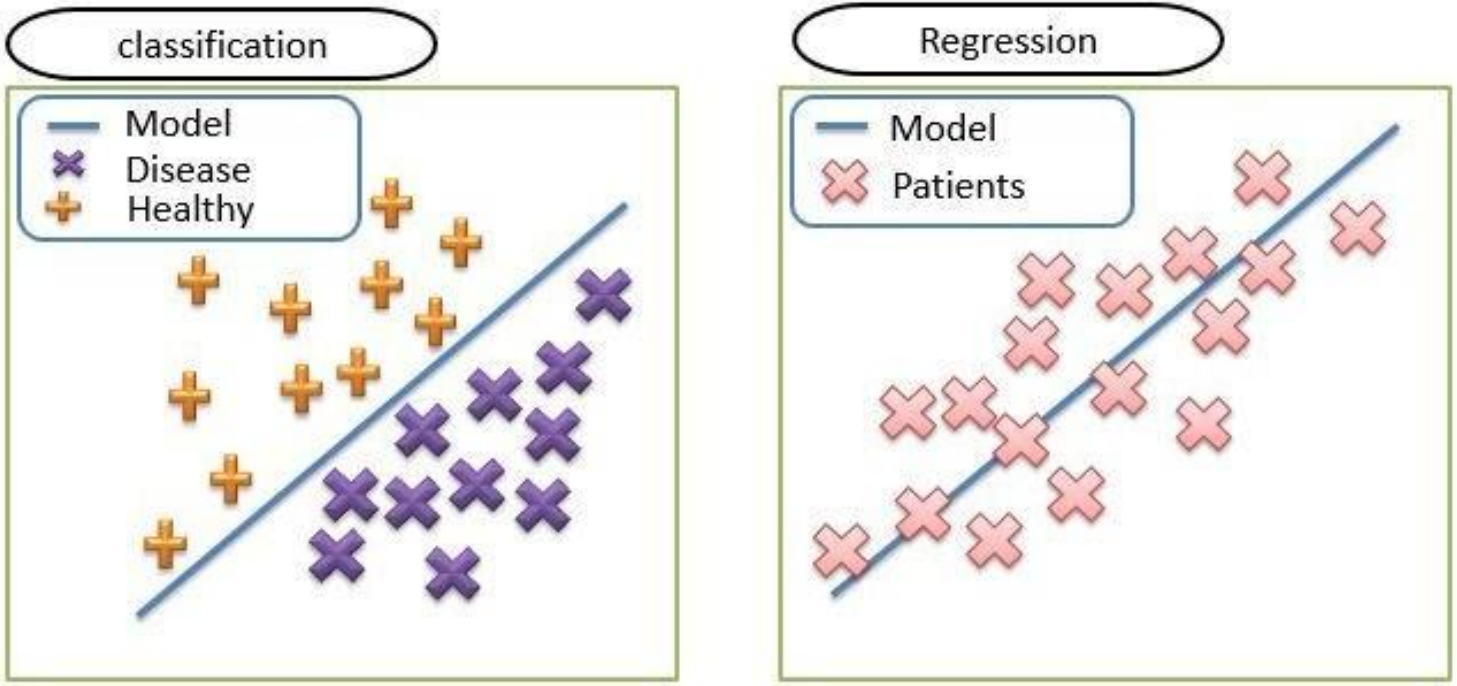
Supervised and unsupervised learning



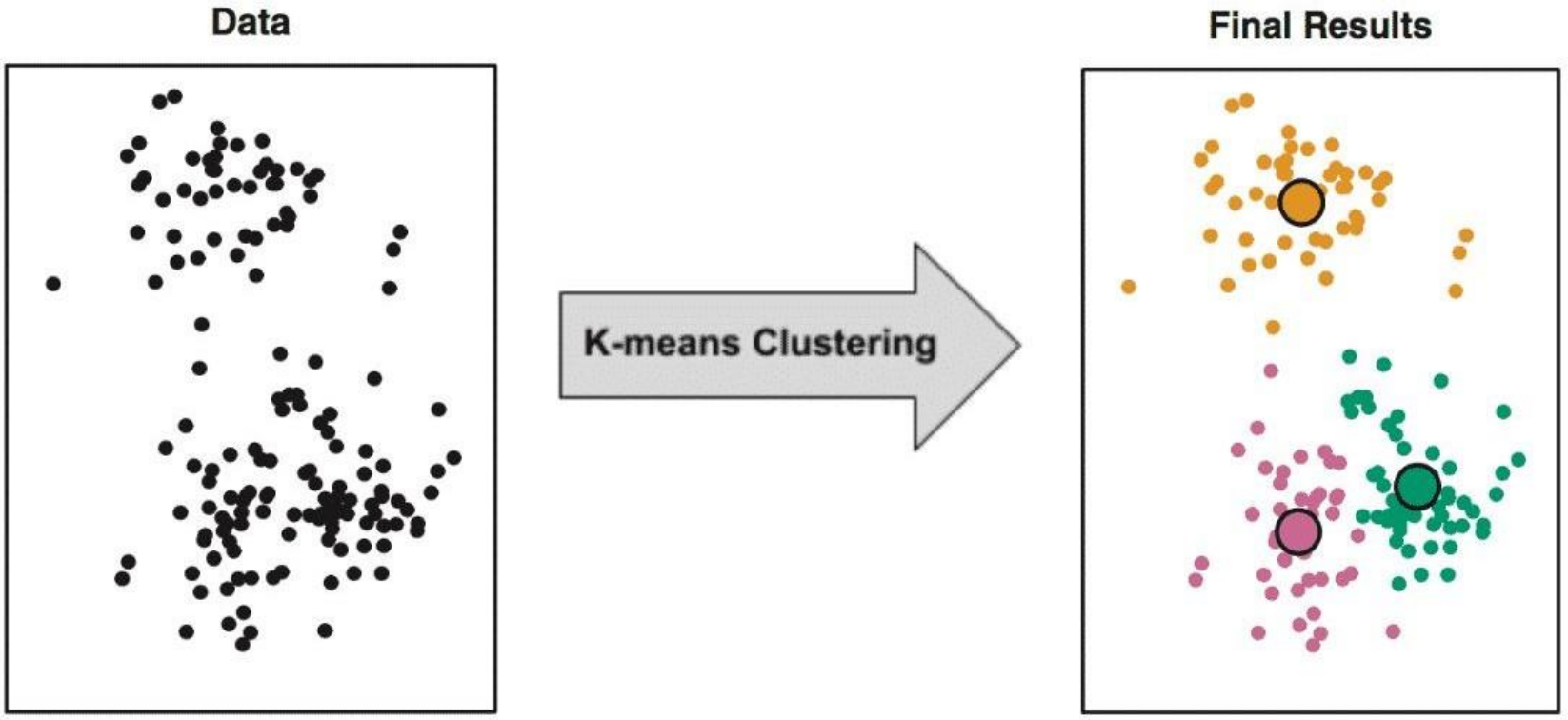
Supervised learning



Classification and regression



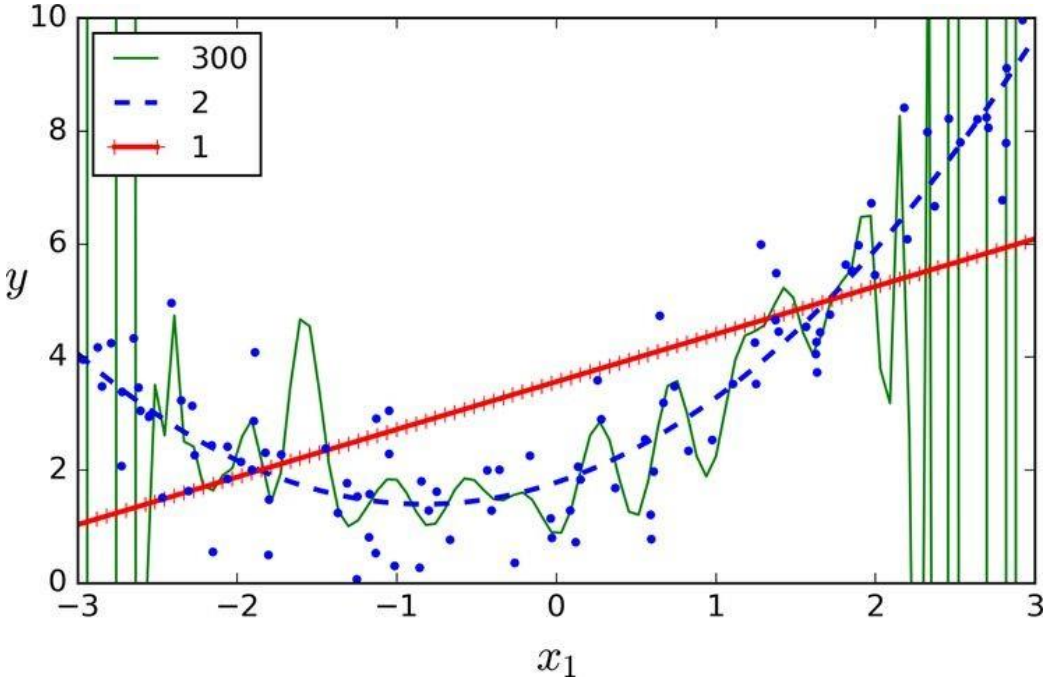
Unsupervised learning



Sklearn: my favourite package

Machine learning and deep learning looks like a fantastic way to make billions, but where’s the catch?

Overfitting is one of the biggest issues we have to deal with



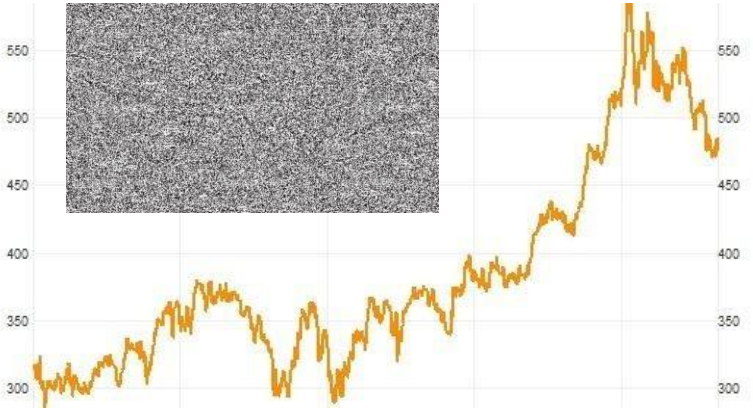
Good:

- SVM
- Linear Regression
- Naive Bayes

Bad:

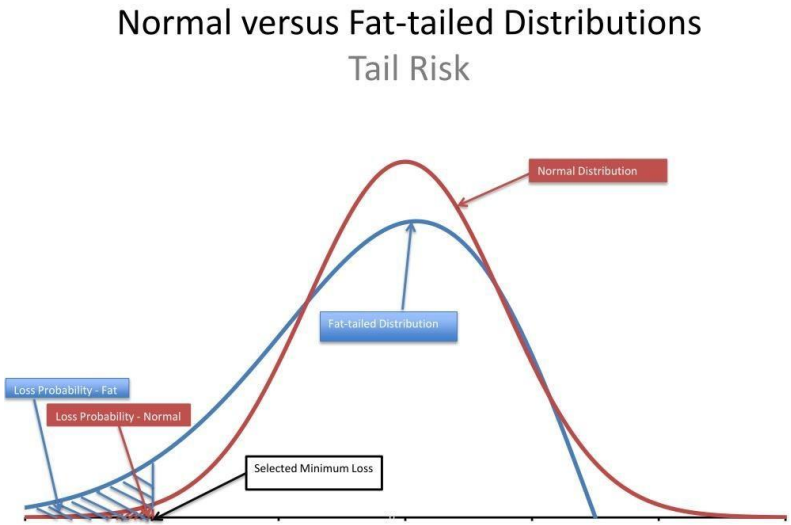
- Decision Trees
- High-parameter polynomials
- Neural networks

Another issue: random, fat tailed stock market returns which are worse than white noise



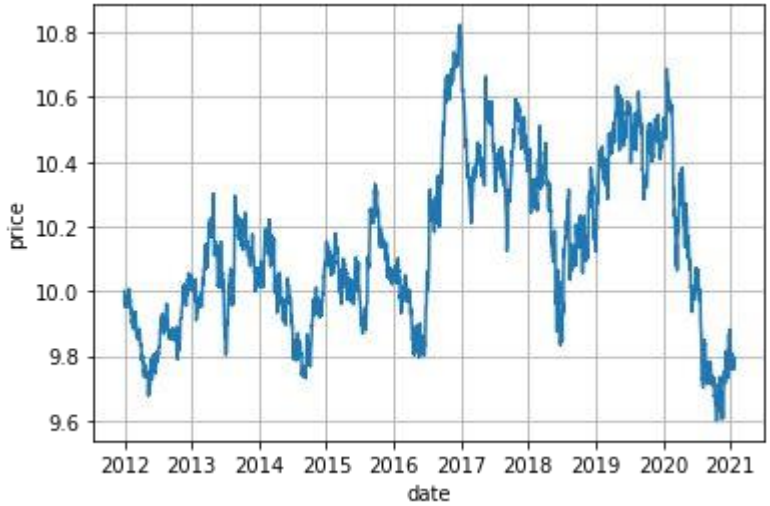
First approximation: normal distribution of returns

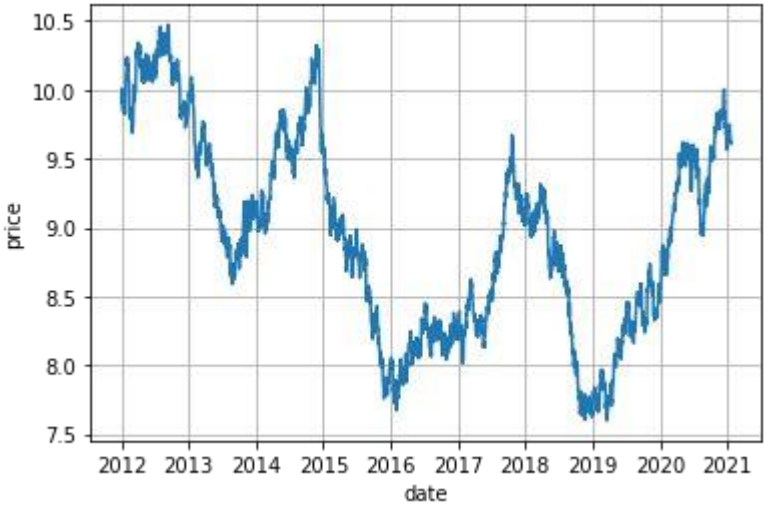
More accurately: Fat-tailed Cauchy distribution



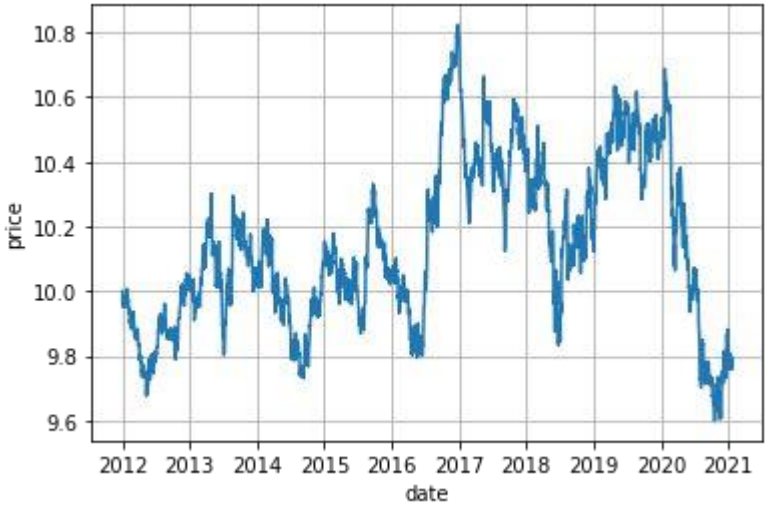
So, is there anything at all that we can use for predicting financial time series?

Can you tell which of these price charts is more predictable?

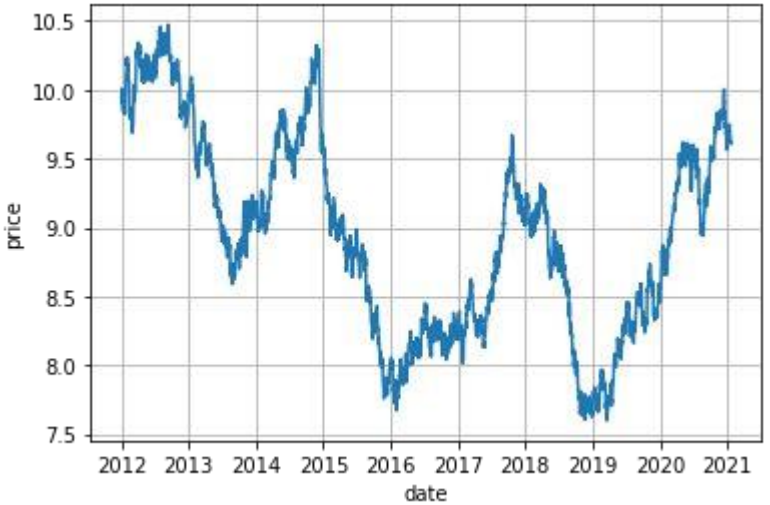




What makes a time series predictable:

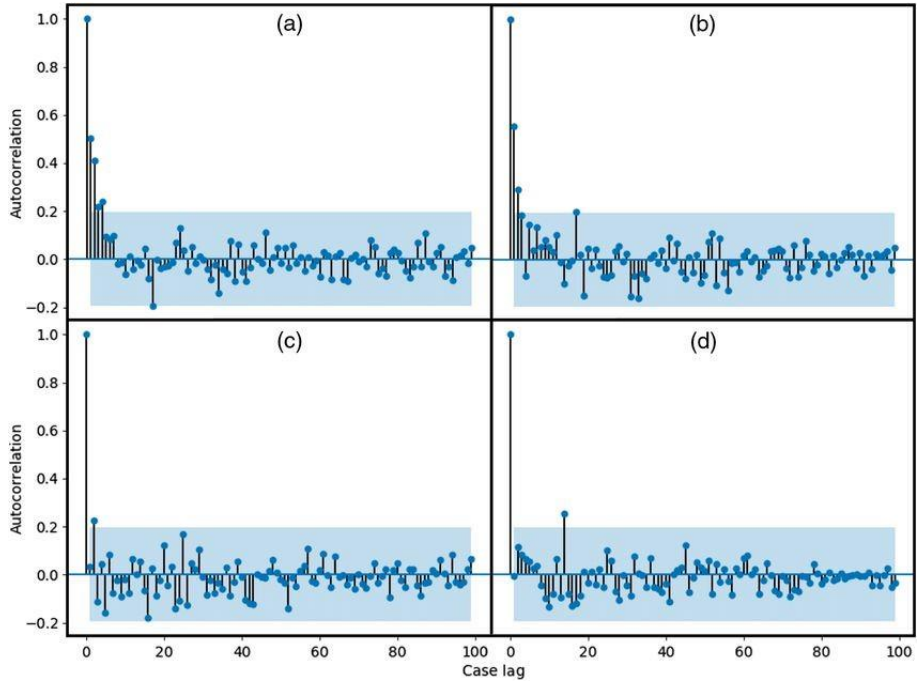


Highly predictable

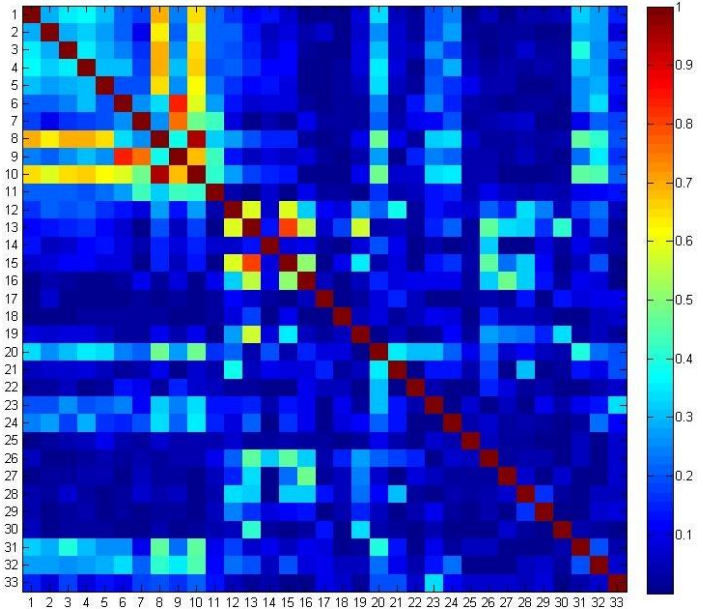


Unpredictable

Correlation to the rescue…



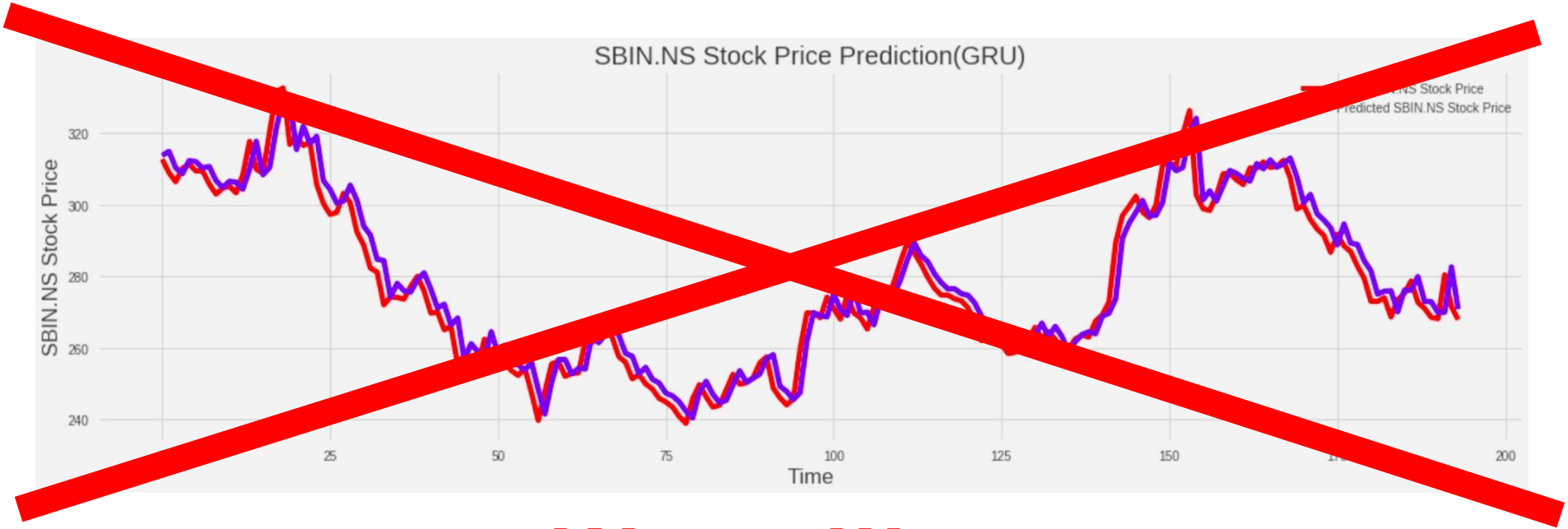
Autocorrelation



Factor correlation

Stock market indicators are really just noise filters

Never predict stock prices!



Wrong!!!

Is it really as simple as finding linear relationships between returns?

NO! Of course not.

Market Complexity: non-linear correlations

Keynsian beauty contest:

Predicting the winner

vs

Predicting who everyone else thinks is the winner

More Complexity: delayed gratification

The marshmallow experiment:

Small reward now

vs

Larger reward later

Is the optimal strategy to wait?

What if:

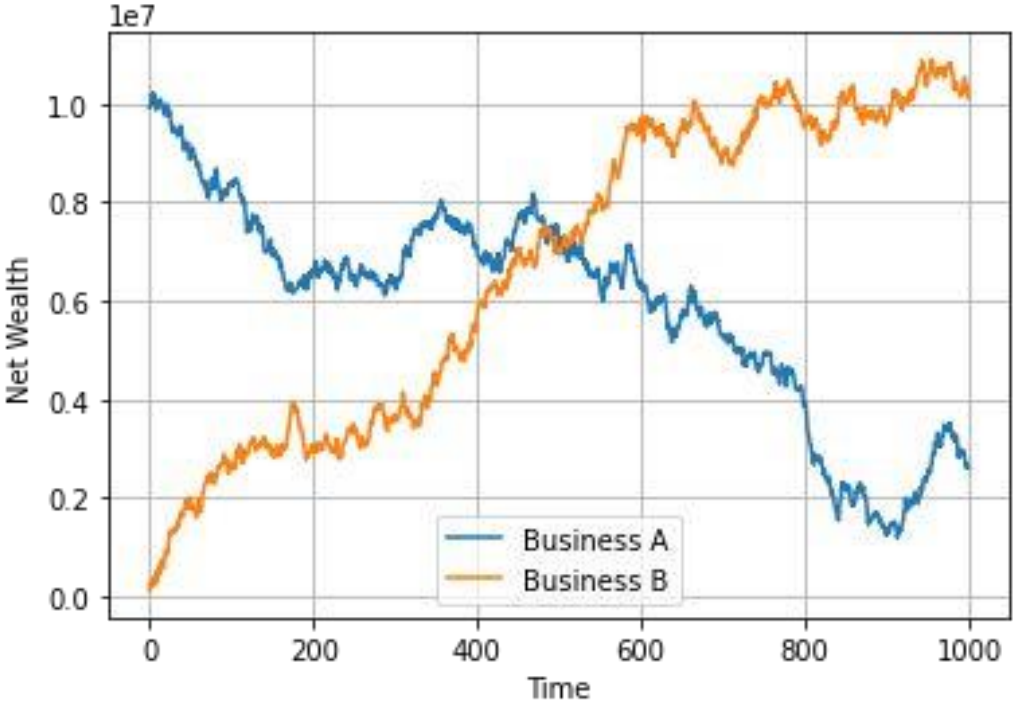
The higher reward is received a VERY long time in the future?

The future reward is only marginally bigger than the current reward?

The trading equivalent of the marshmallow test

What would you do here?

What's the optimal strategy?



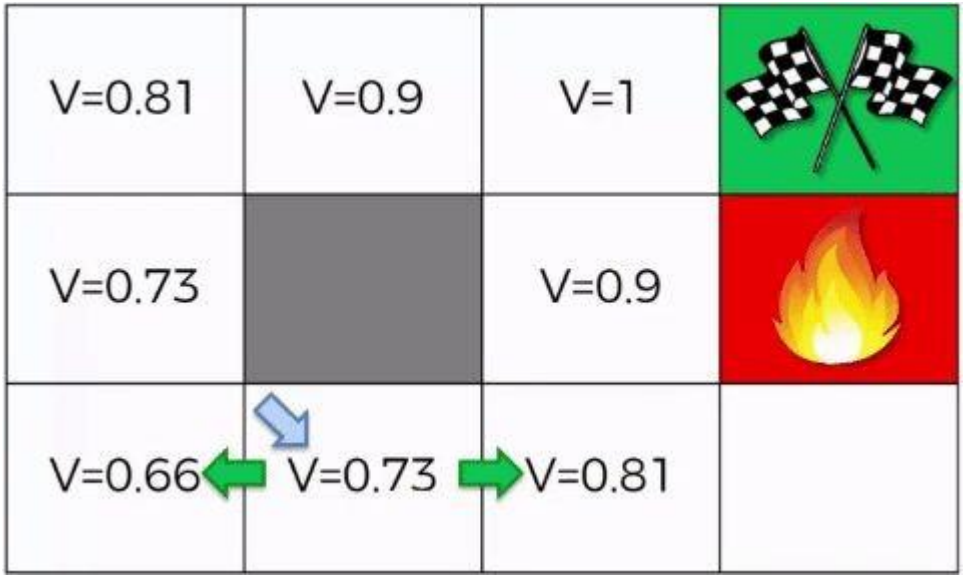
At T = 500, should the business make a $6MM investment?

The answer is NOT clear-cut as decisions now have repercussions some time in the future after a number of unpredictable events.

Business A could get close to bankruptcy by making that investment but is saved from inevitable bankruptcy later on because of it.

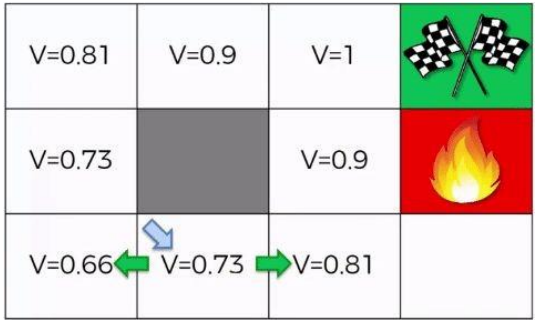
Optimal control theory

V(s,a) = r(s,a) + gamma * max{ V(s',a) }

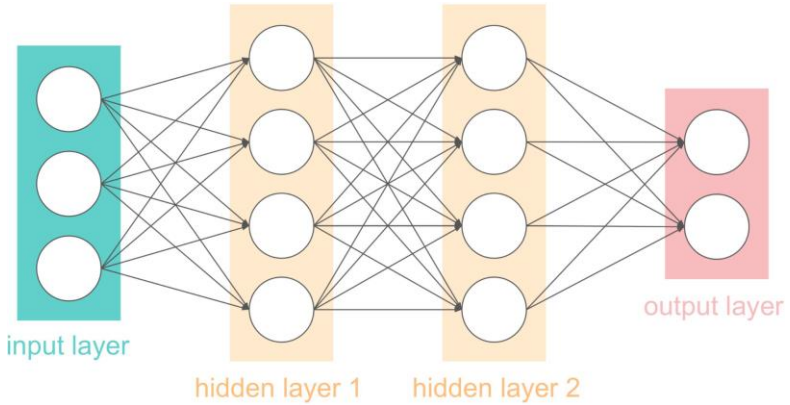


V(s,a), is equal to the maximum of different Rewards you can get from that state by performing any of the allowed actions, r(s, a) and the 'discounted' value of new state where you will land upon by taking that particular action 'a'.

Deep RL replaces Q-tables with neural networks

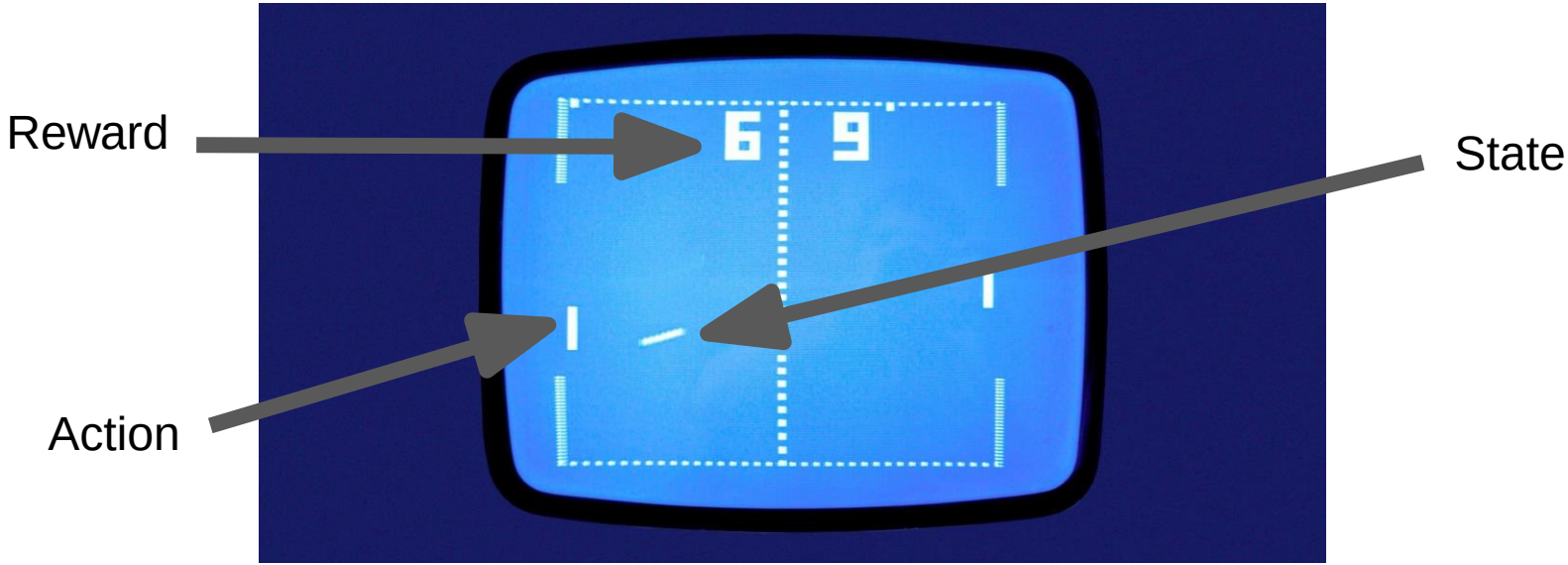


With a large number of probably states Q-tables become increasingly complex. ANNs can approximate their functionality.



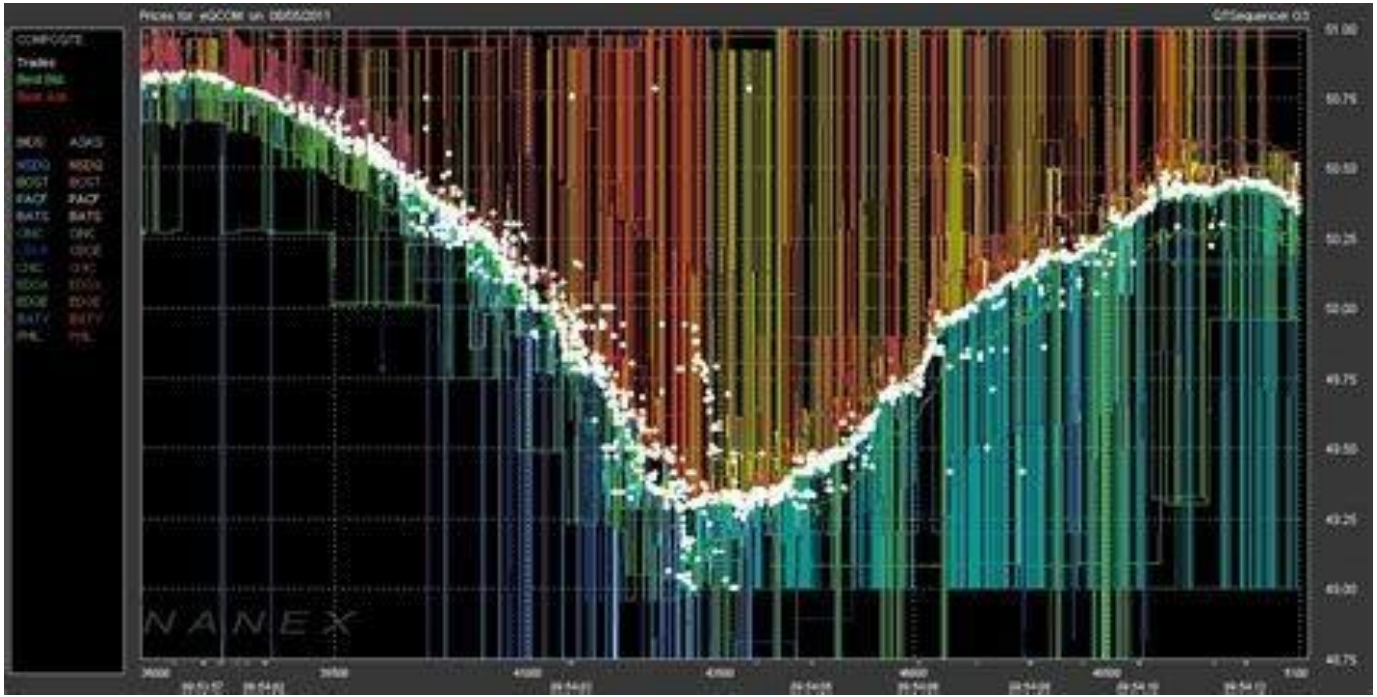
Playing Games

Each action is assigned an implied reward. Actions with higher probabilities of winning receive higher implied rewards.

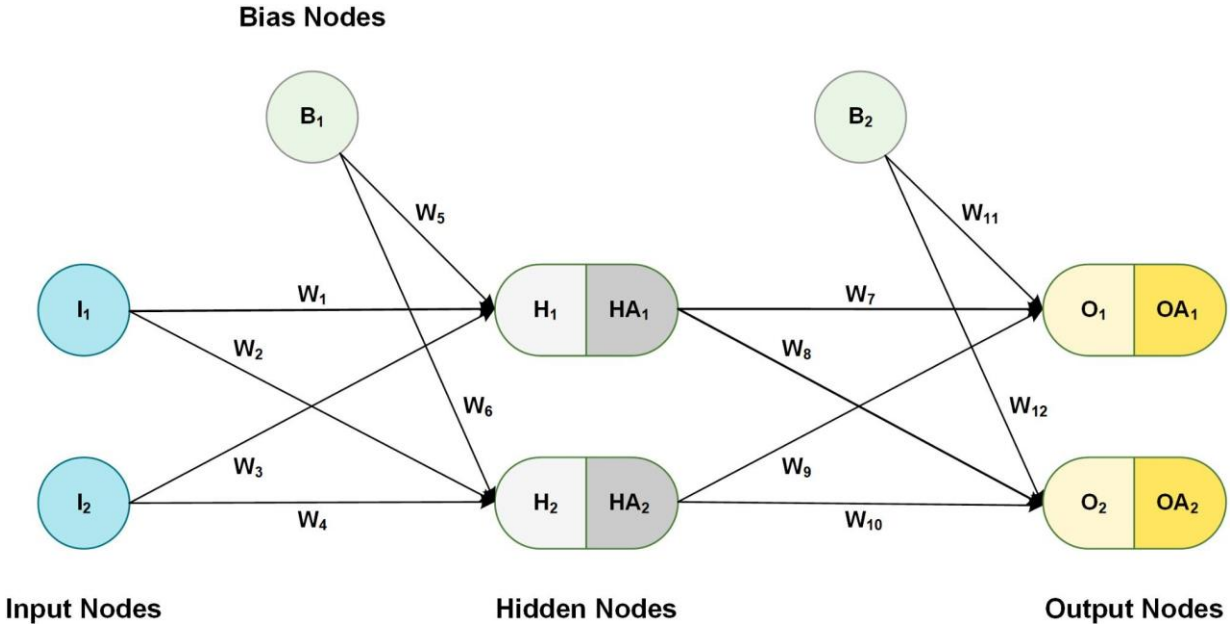


Other uses of backward induction

This method is used very successfully in execution algorithms and market making

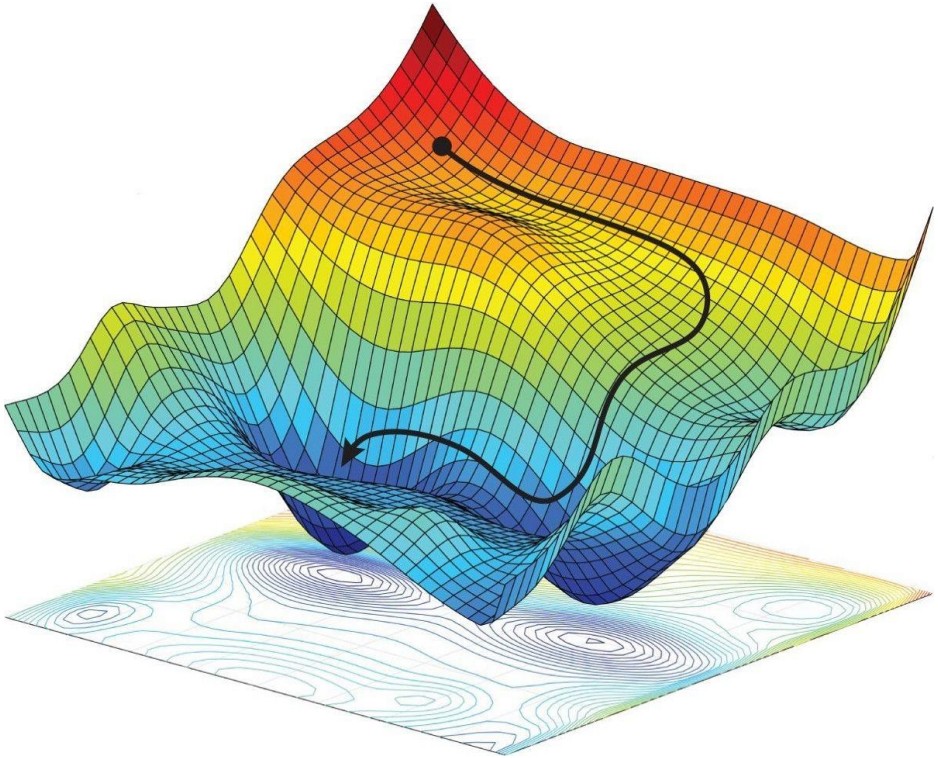


How does a neural network operate?



The weights and values applied to each node are transformed into a new value by the activation function.

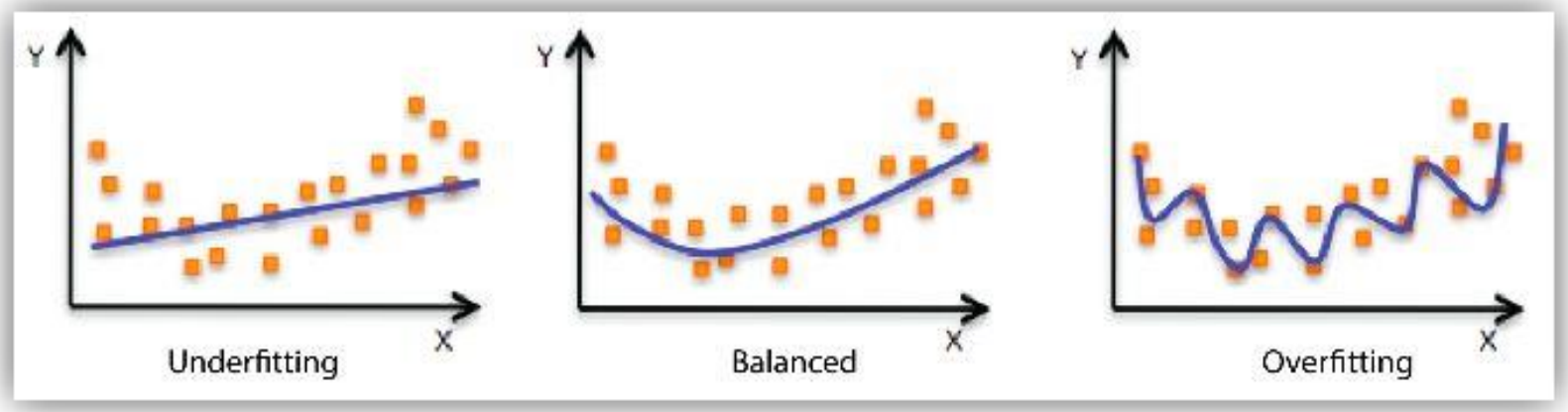
Optimization



The difference between the output and the actual training data is call the “Loss function” and it is minimized by changing the weights between the nodes.

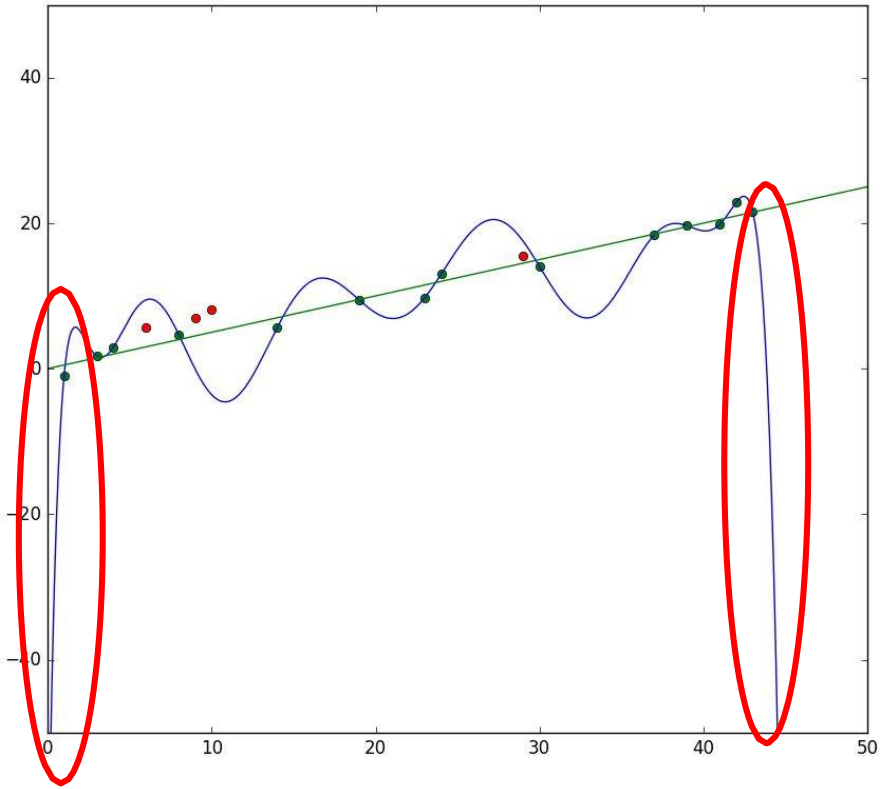
A popular optimization algorithm is “stochastic gradient descent” (SGD).

Caveat: fitting a function



ANNs can fit practically any function, but which one should they fit in any given case?

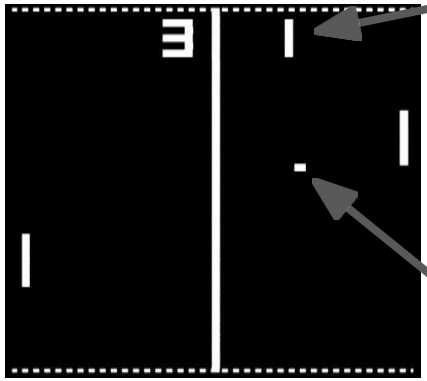
Typical out-of-sample performance for overfitting



Smoothing the noise

Technical indicators can smooth noise and provide additional state information but they also have a lag and may introduce non-existing autocorrelations.

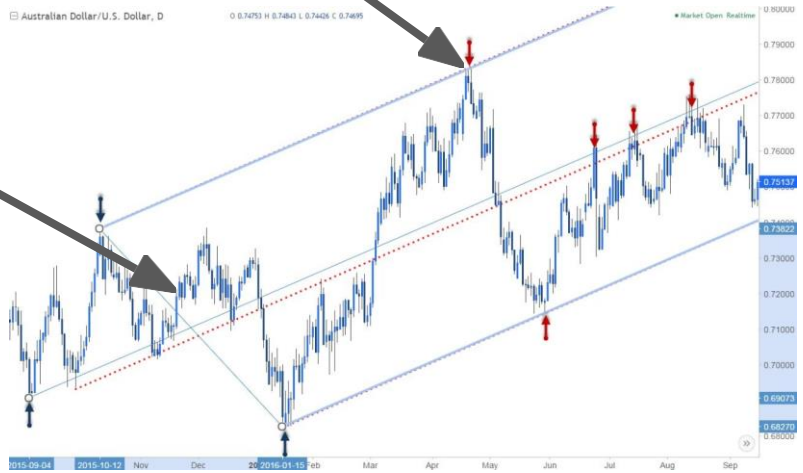
Gamification of trading



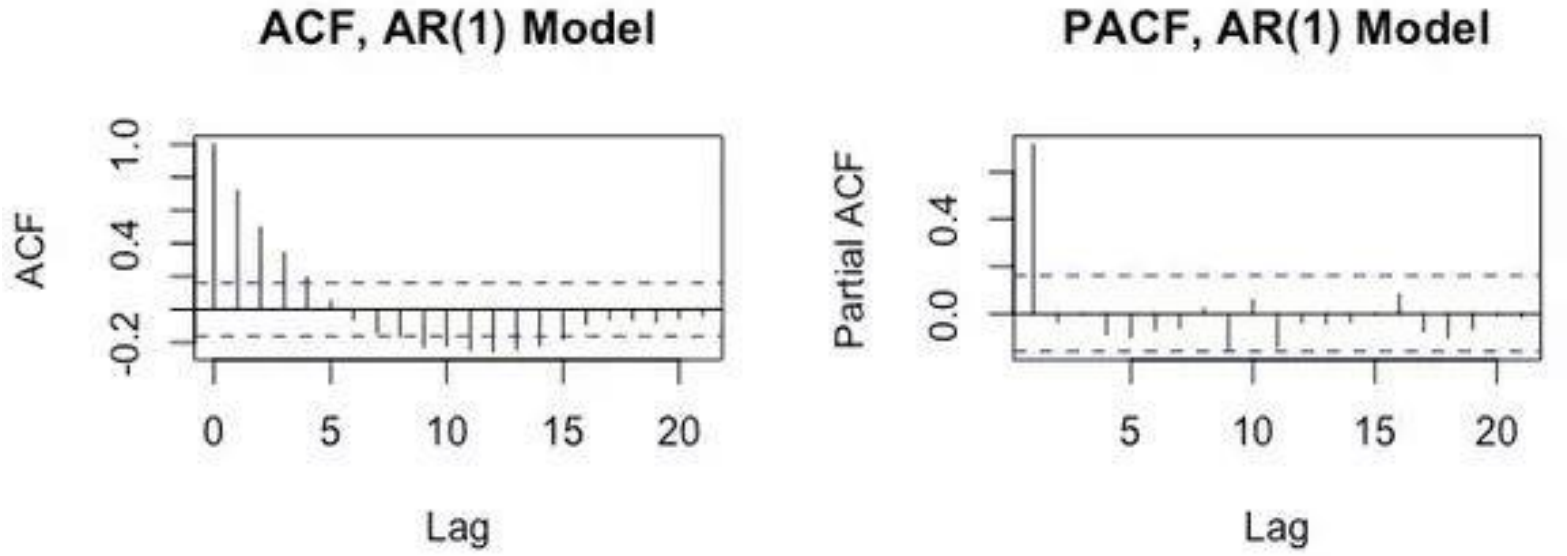
PnL

Buy, sell, hold

State

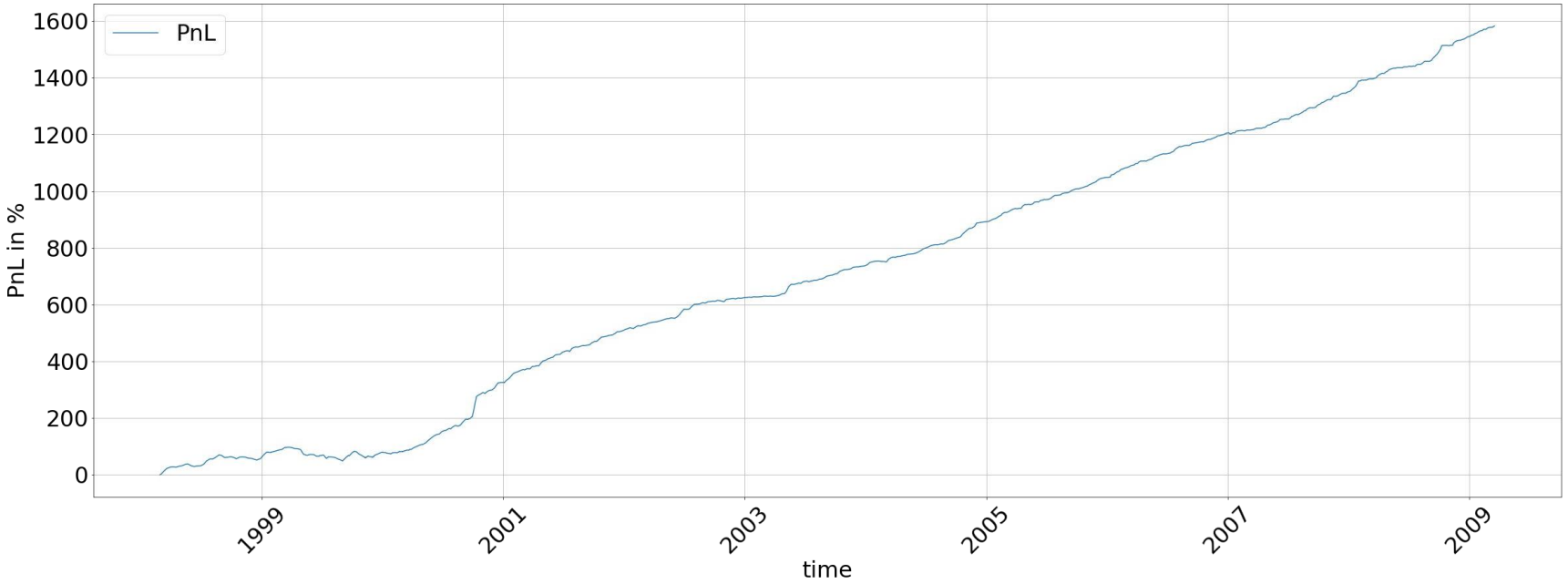


Prerequisite: time series needs to be forcastable.

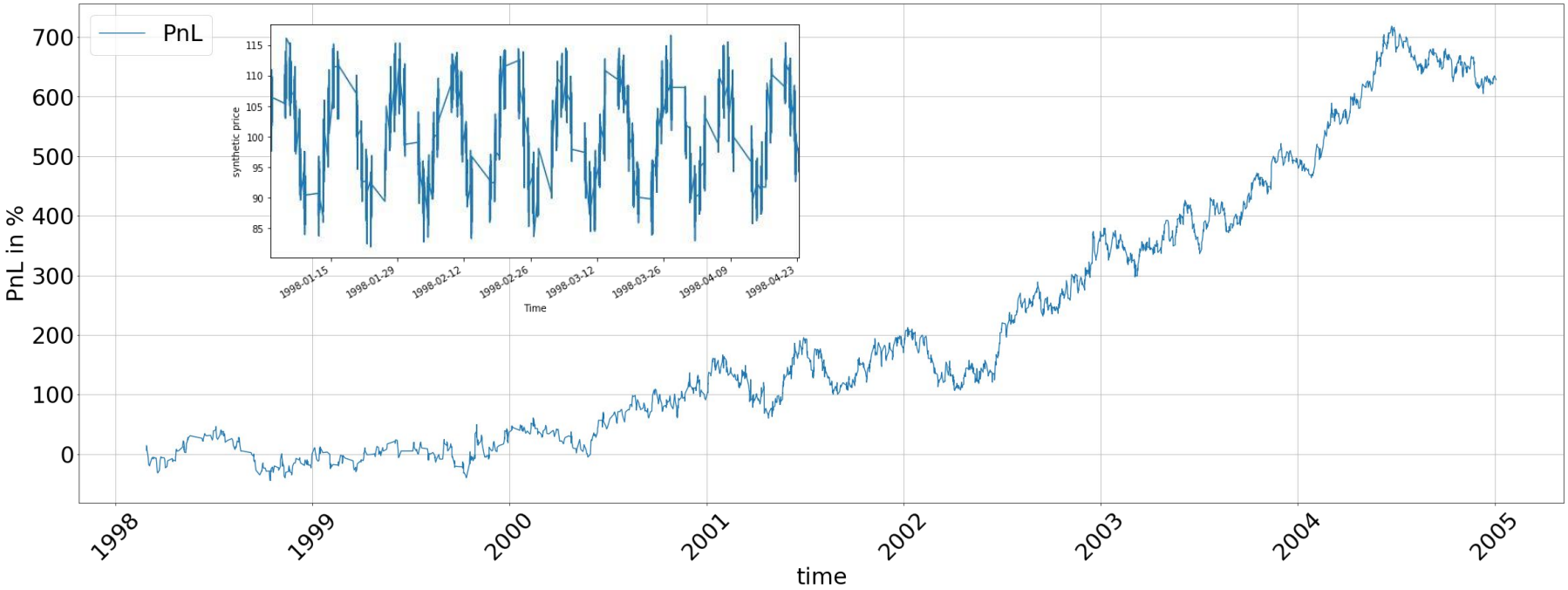


This is an inherent assumption we make if we apply RL to trading without any further testing.

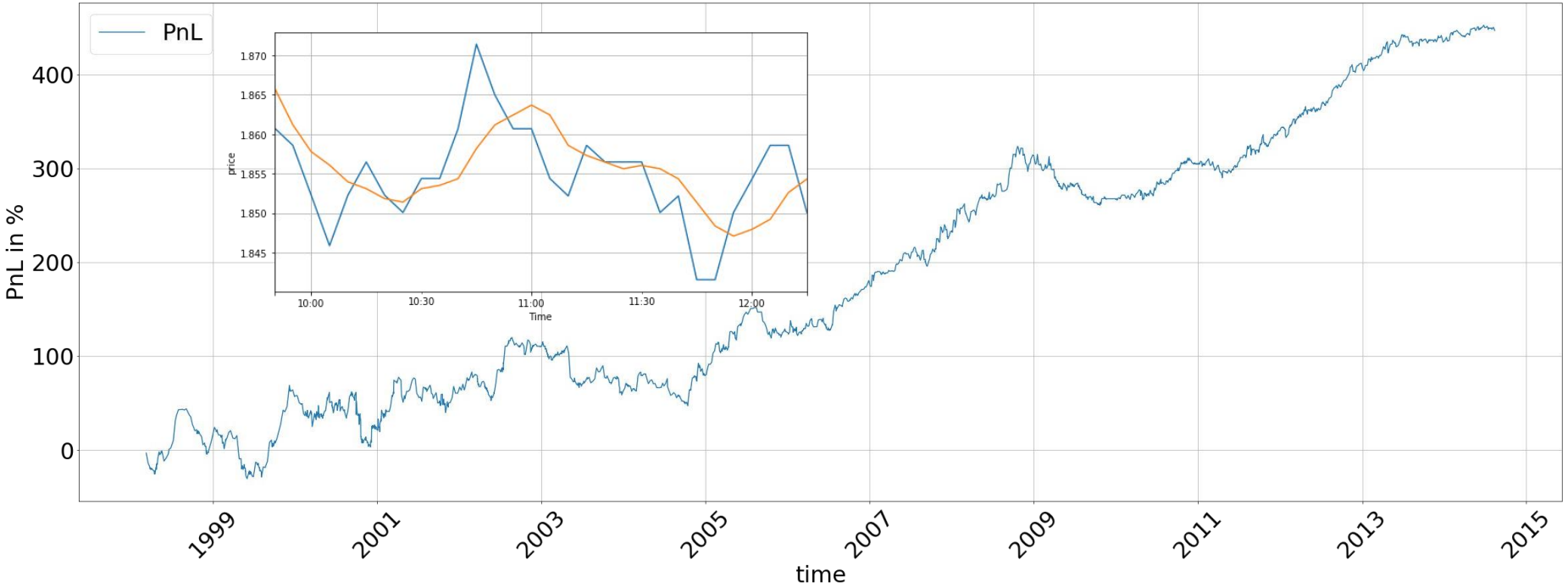
Testing the RL-trader: sine wave



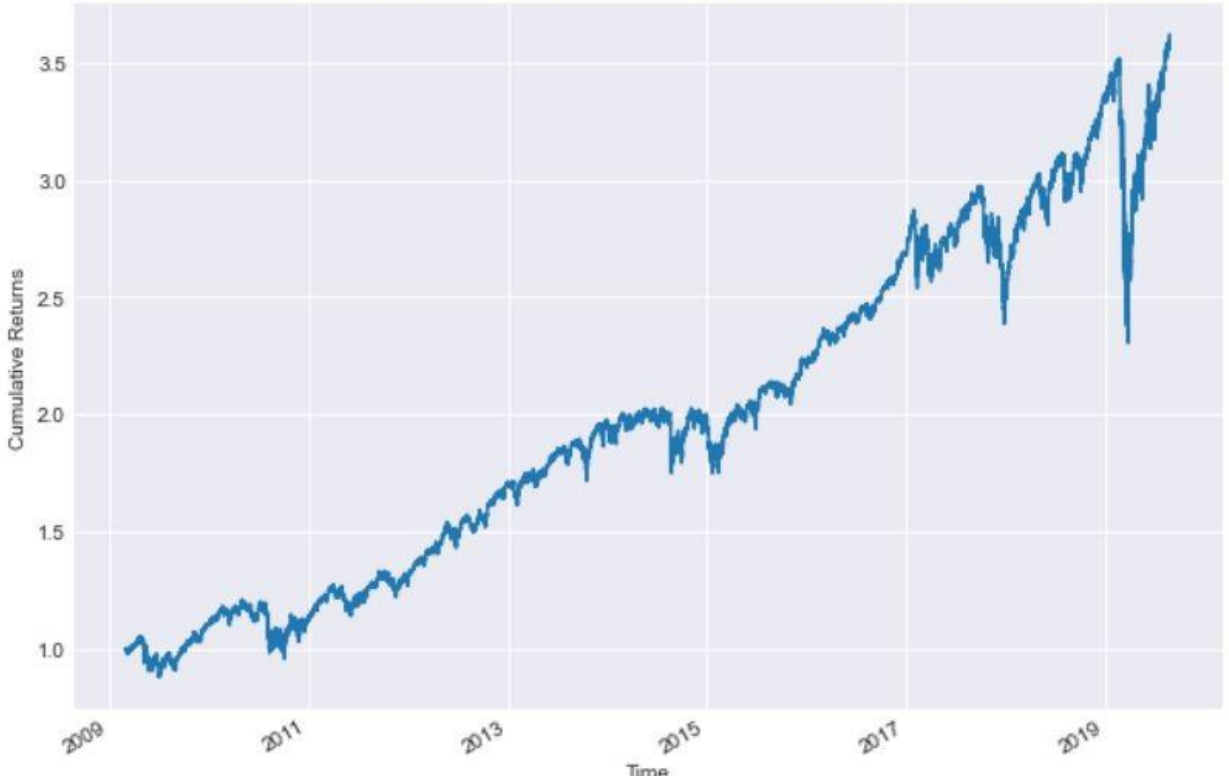
Testing the RL-trader: noisy sine wave



Performance SPY with 5-day MA



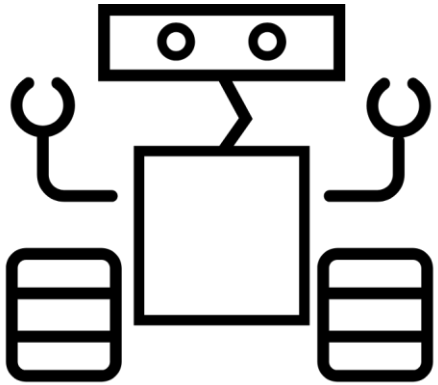
SPY



Algorithm converges to a buy-and-hold pattern as it finds that this is the optimal strategy - say hello to Warren Buffet.

Better reward function design?

- Pure PnL on exit
- Profit per tick
- Sharpe ratio
- Punishment for long hold times
- Punishment for drawdowns
- Binary win/loss
- Signed categorical
- Unsigned categorical (using exponential)



State features

- OHLCV
- Technical indicators
- Time of day, day of week, time of year
- Different time granularity
- Other instruments
- Alternative data

Lessons learned

- RL overfits very easily
- Mostly learns very basic market patterns
- Complex “Zoo” of hyper-parameters
- Reward function design is very challenging
- RL is not a “silver bullet”
- Market experience is highly advisable

However:

- It can and will come up with strategies that are hardly conceivable to us that may actually be optimal in the long run.