Financial Trading using Reinforcement Learning

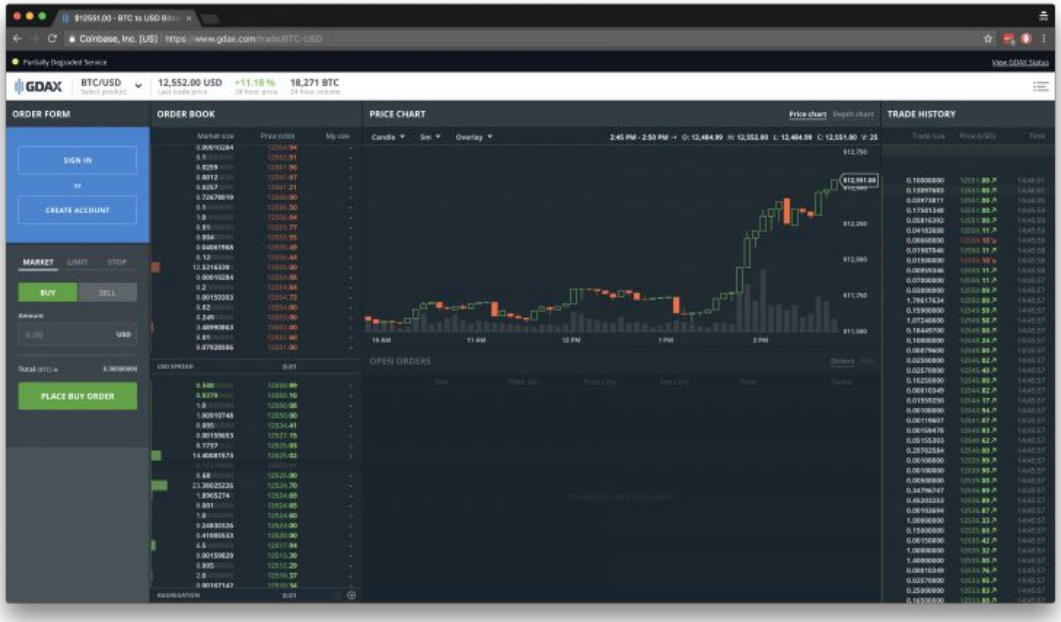
A Major Project Presentation.

Aditya Pareek
(2K15/MC/006)

Shikhar Bhardwaj
(2K15/MC/078)

Tushar Prasad
(2K15/MC/093)

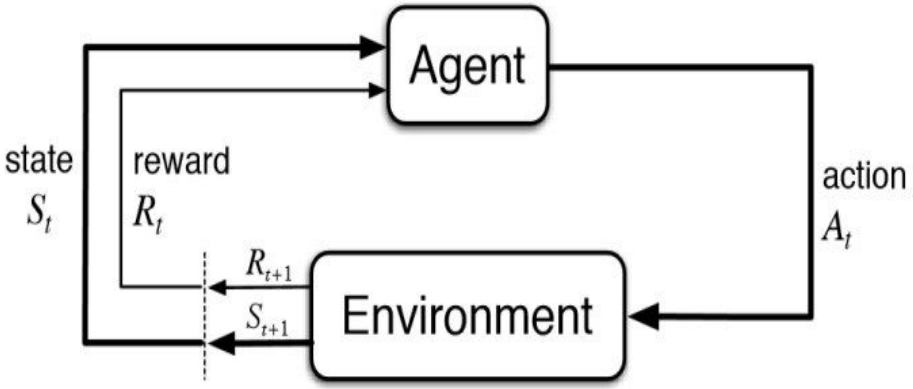
Financial Markets and Trading



- Price Chart
- OHLC
- Volume
- Trade History
- Order Book
- Spread

This is GDAX, one the most popular U.S based exchanges. Given graph represents the BTC-USD trade exchange.

Reinforcement Learning



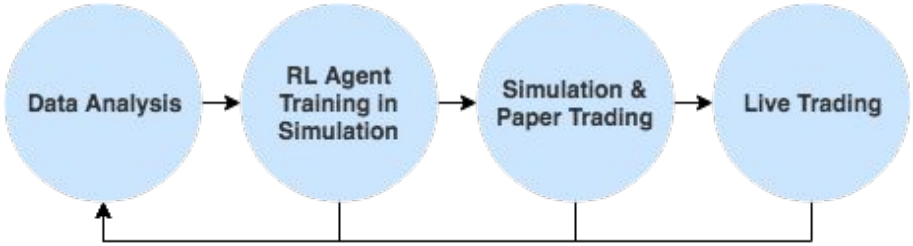
Reinforcement learning is an approach to machine learning that is inspired by behaviorist psychology. It is similar to how a child learns to perform a new task.

Reinforcement learning contrasts with other machine learning approaches in that the algorithm is not explicitly told how to perform a task, but works through the problem on its own.

An agent interacts with its environment, receives a reward/penalty depending on how it performs.The agent over time makes decisions to maximize its reward and minimize its penalty

Reinforcement Learning in a Trade Setting

Why can Reinforcement Learning be a good algorithm in a financial trade domain?



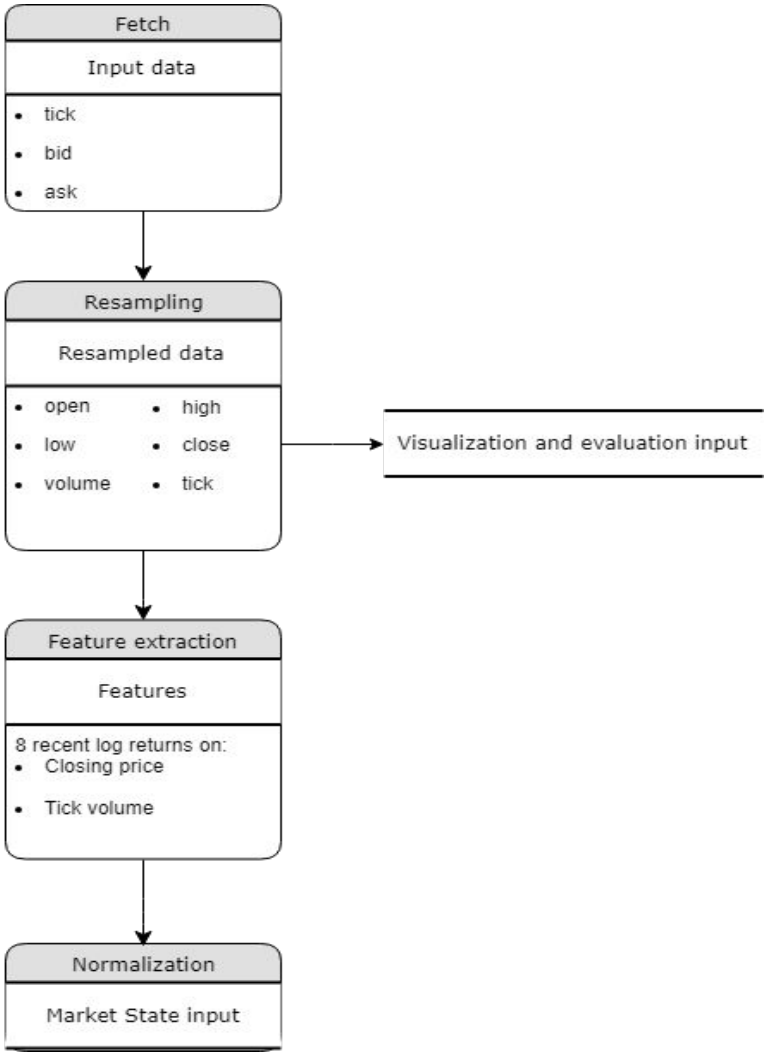
- End-to-end Optimization of what we care about.
- Learned Policies
- Trained directly in Simulation Environments
- Learning to adapt to market conditions
- Ability to model other agents

Problem Statement

To model an automated agent that seeks to generate constant profit from the financial market, by leveraging the recents advancements in Deep Reinforcement Learning

The work is inspired from the paper: “Financial Trading as a Game: A Deep Reinforcement Learning Approach” by Chien-Yi Huang [2018].

Implementation



Data

- Top-of-the book forex data for 15 major currency pairs, with fractional pip spreads in millisecond detail.
- The entire dataset comprises of 1080 archives for each month for each currency pairs, 26 GB in total.
- We need some data preprocessing and feature extraction to make training feasible on this large dataset.

Environment

State Space	Action Space	Reward Function
1. Periodic time features Minute, hour, day of the week and month of the year modelled as - $f_t = \sin\left(\frac{2\pi t}{T}\right)$ **2. Market history features** 7 most recent log returns on closing price and tick volume. 14 features per currency pair. **3. Trading position features** One-hot encoding of the most recent action - buy, sell or hold. 3 features per currency pair	Actions are encoded as integers • Hold (**0**) • Buy **(1)** • Sell **(2)** The position held on the asset currently is denoted by a one-hot encoding of this index.	The reward function is the logarithmic portfolio returns for the time step at which the action was taken $r_t = \log\left(\frac{v_t}{v_{t-1}}\right)$ The value of the portfolio is calculated as follows $v_t = v_{t-1} + a_t . c . (c_t - o_t)$

Our Approach

Algorithm 1 Financial DRQN Algorithm

Initialize $T \in \mathbb{N}$, recurrent Q-network Q_θ, target network Q_{θ^-} with $\theta^- = \theta$, dataset $\mathcal{D}$ and environment E, $steps = 1$
Simulate env E from dataset $\mathcal{D}$
Observe initial state s from env E
for each step **do**
  $steps \leftarrow steps + 1$
  Select greedy action w.r.t. $Q_\theta(s, a)$ and apply to env E
  Receive reward r and next state s' from env E
  Augment actions to form $\mathcal{T} = (s, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s'})$ and store $\mathcal{T}$ to memory $\mathcal{D}$
  if $\mathcal{D}$ is filled and $steps \mod T = 0$ **then**
    Sample a sequence of length T from $\mathcal{D}$
    Train network Q_θ with equation (4) and (5)
  end if
  Soft update target network $\theta^- \leftarrow (1-\tau)\theta^- + \tau\theta$
end for

$$L(\theta) = E_{(s,a,r,s`)\sim D}[|| \, r + \gamma . Q_{\theta^-}(s`, argmax_{a`}\{Q_\theta(s`, a`)\}) - Q_\theta(s, a) \, ||^2]$$

Training and Evaluation

Expectations:

- We expect an agent to learn about the basic rule of financial trading (buy low and sell high) within the first few hundred hours of experience with the market state.
- We evaluate this hypothesis over 3 models of varying architecture to see which agent is able to learn the basic rules of the task with minimal training time and experience.
- The training is performed on a single currency pair market state data for the EURUSD pair during the year 2012.
- Each episode consists of 200 data points (15-minute intervals). The agent is trained over 10 episodes for a total experience of around 500 hours.

Results

- Agent 1
  - 2 Fully connected Dense hidden layers of 256 units each
- Agent 2
  - 2 Fully connected Dense hidden layers of 64 units each

Training output:

Even after a cumulative experience exceeding 1500 hours, both these models failed to exploit the environment during the evaluation setting, acting only to hold the initial position of no-risk.

This behaviour can be explained by the lack of state memory in this agent's model, which causes it to not learn about experiences involving some actions taken in the recent past. Thus it cannot relate past actions to current rewards and Chooses to keep the risk-free position.

Results

- Agent 3
  - 2 Fully connected dense hidden layers of 32 units each, followed by an LSTM layer of 64 units.

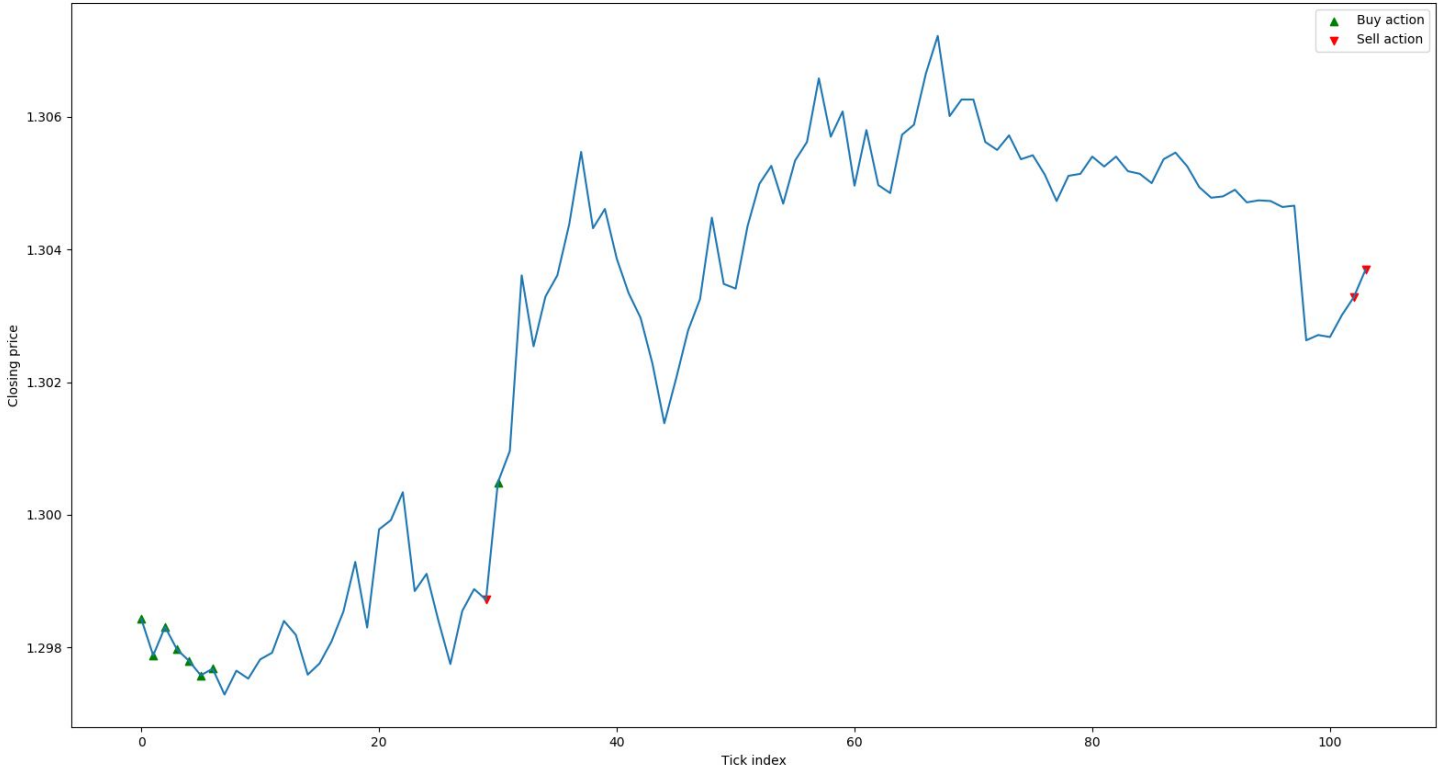
Training output:

After training the model for a cumulative experience of 500 hours, it is able to recognise positions of profit and take actions to exploit them.

Evaluation:

The model is evaluated on a sample of 400 points (100 hours) from the year 2013.

Results


Buy action
Sell action
1.306
1.304
1.302
1.300
1.298
Closing price
0
20
40
60
80
100
Tick index

Results

```
λ tradegame/src ∴ python3 evaluate.py EURUSD_2012_norm /working_2012/model_ep10
Using TensorFlow backend.
2018-12-07 07:20:04.982210: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:964] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2018-12-07 07:20:04.982613: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: Tesla K80 major: 3 minor: 7 memoryClockRate(GHz): 0.8235
pciBusID: 0000:00:04.0
totalMemory: 11.17GiB freeMemory: 11.10GiB
2018-12-07 07:20:04.982645: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2018-12-07 07:20:05.292523: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor
with strength 1 edge matrix:
2018-12-07 07:20:05.292588: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2018-12-07 07:20:05.292597: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2018-12-07 07:20:05.292881: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:lo
calhost/replica:0/task:0/device:GPU:0 with 10759 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 00
00:00:04.0, compute capability: 3.7)
Evaluating the model on sampled data points.
----------------------
Evaluation parameters
----------------------
Start cash : 10000
Trade size :  1000
----------------------
Buy: Rate = $1.2984300 | Cost =  1298.43
Buy: Rate = $1.2978800 | Cost =  1297.8799999999999
Buy: Rate = $1.2983100 | Cost =  1298.3100000000002
Buy: Rate = $1.2979700 | Cost =  1297.97
Buy: Rate = $1.2978000 | Cost =  1297.8
Buy: Rate = $1.2975800 | Cost =  1297.58
Buy: Rate = $1.2976800 | Cost =  1297.6799999999998
Sell: $1.2987200 | Profit:  $0.2900000
Buy: Rate = $1.3004900 | Cost =  1300.49
Sell: $1.3032800 | Profit:  $5.4000000
Sell: $1.3037000 | Profit:  $5.3900000
--------------------------------
Closing all open positions to compute portfolio value
Total Profit: $38.0600000
Final portfolio worth: $10038.0600000
--------------------------------
```

thank you, next.

P.S. It was god’s plan all the time.