Dive Deeper in Finance

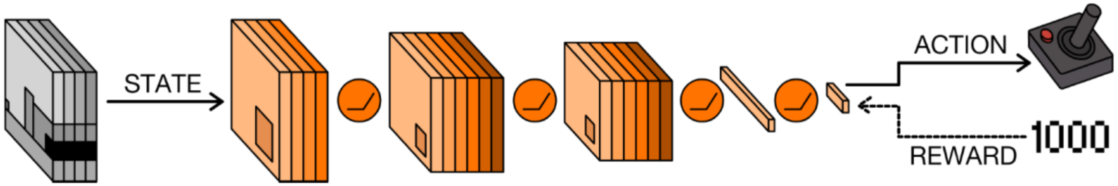
GTC Inspired Deep Learning Event, Munich

Daniel Egloff
Dr. sc. math.
Managing Director QuantAlea
October 10, 2017

Reinforcement Learning

Reinforcement Learning

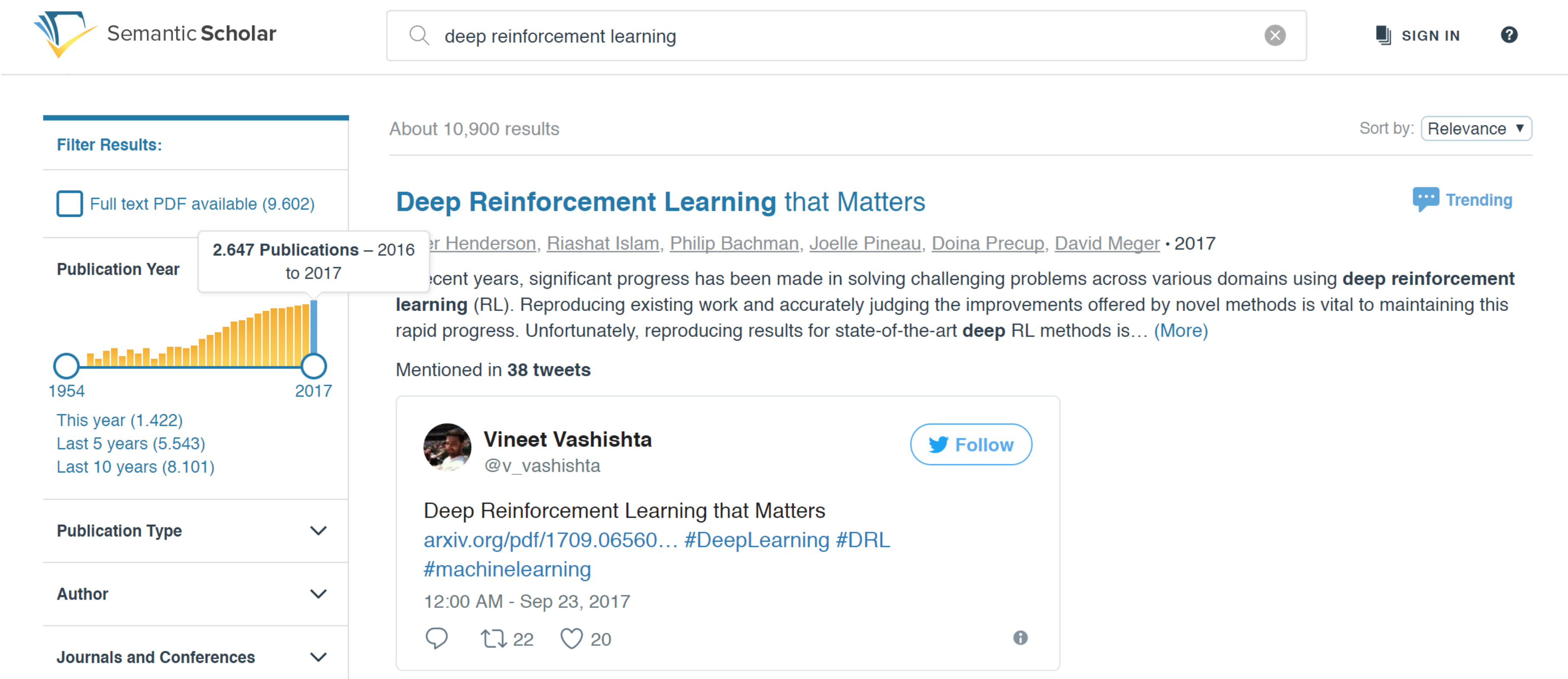
- Emerged from two streams of research
  - Bellman's 1957 work on optimal control
  - Animal learning with trial and error
- Deep Learning allows RL to scale
- Recent applications
  - Learning to play Atari games
  - Google AlphaGo



Deep RL is Hot Topic in AI

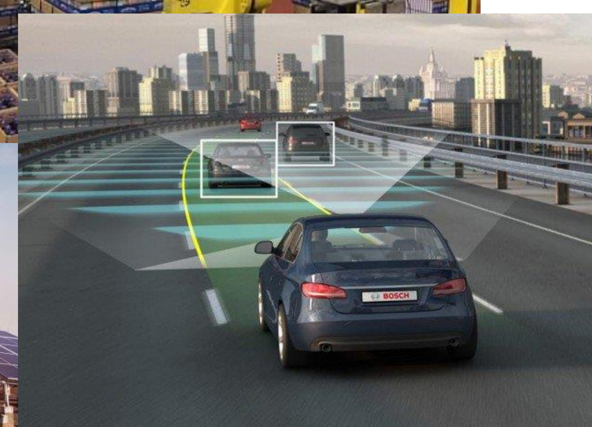
QuantAlea

From **1603** in 2016 to **2647** publications in first 3 quarters of 2017



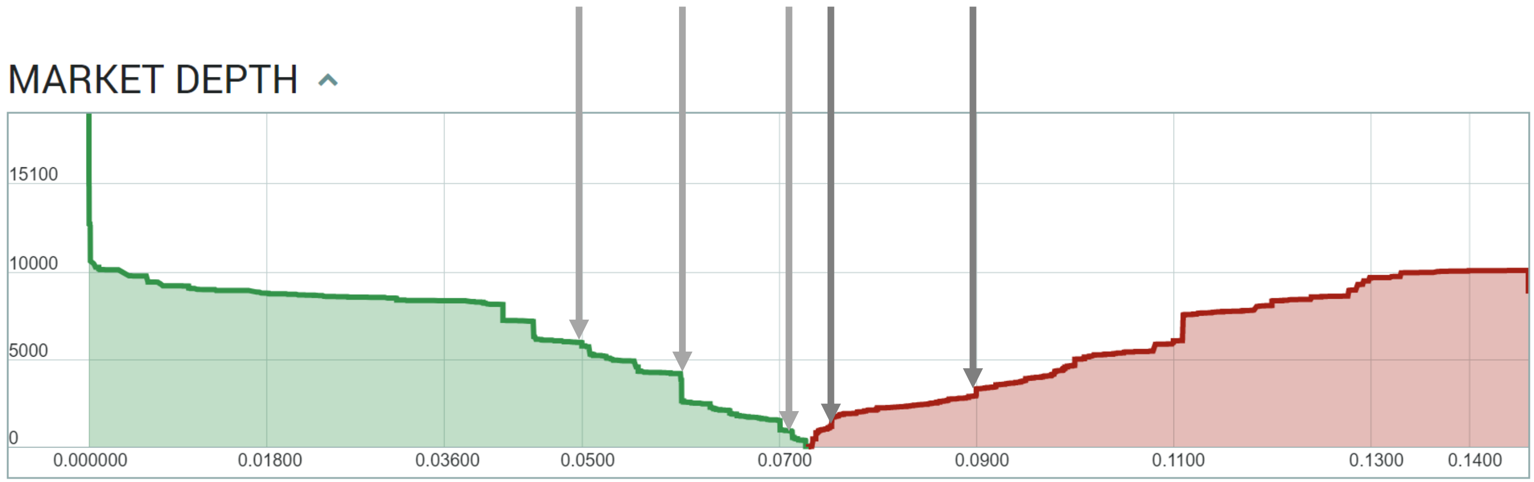
RL in Industrial Applications

- Smart robots for manufacturing, …
- Inventory management
- Supply chain optimization
- Self driving cars
- Autonomous vehicles and drones
- Power grid management



RL in Finance?

- Optimize order execution
  - Learning to place limit orders to reach a position goal
  - Optimally distribute volume over time
  - Incorporate market impact in decision process

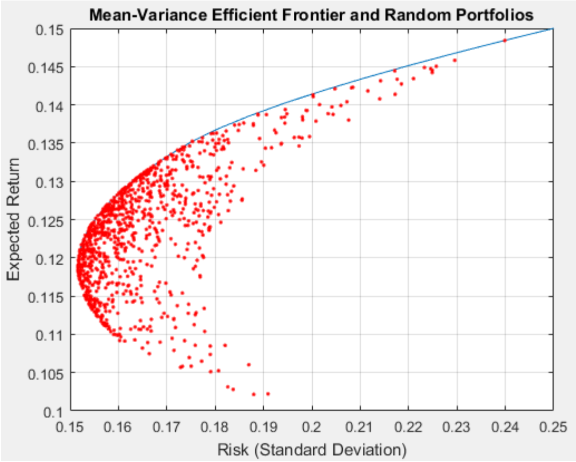


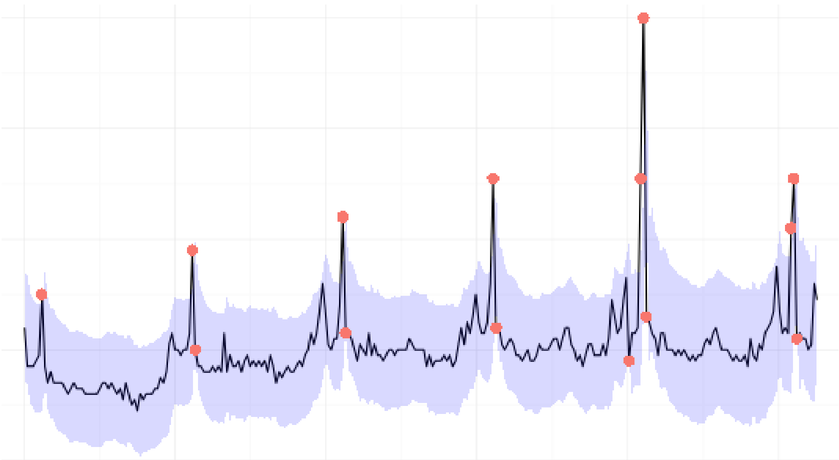
RL in Finance?

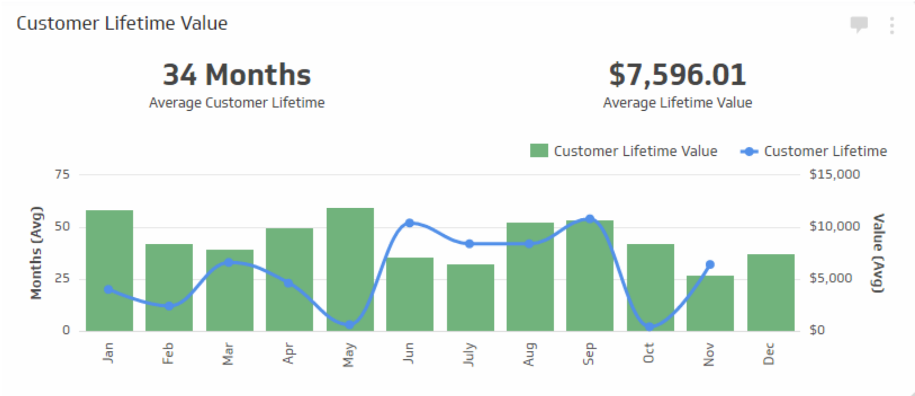
- Learning to trade
  - Risk and reward based trading strategy
  - Learned end-to-end from historical data
  - Online adaption to new market information
  - Consistently incorporate different information sources

More RL Applications in Finance

- Portfolio construction
- Manage customer lifetime value
- Transaction anomaly detection
- Funding and capital optimization
- ….





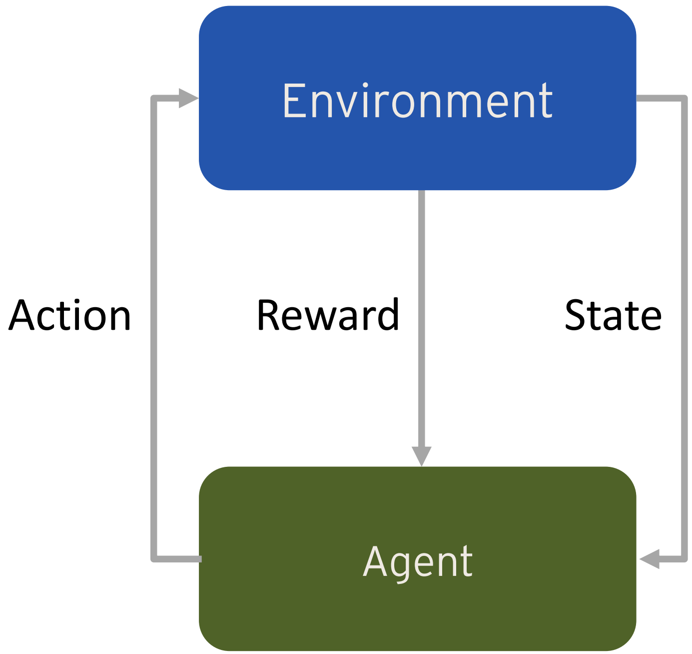


RL Mechanics

RL Setup

QuantAlea

- Learning a behavioral strategy which maximizes long term sum of rewards by a direct interaction with an unknown and uncertain environment



While not terminal do:

Agent perceives state s_t

Agent performs action a_t

Agent receives reward r_t

Environment evolves to state s_{t+1}

Markov Decision Process

- RL is a Markov Decision Process
  - Set of states $\mathcal{S}$
  - Set of actions $\mathcal{A}$
  - State transitions $p(s_{t+1}|s_t, a_t)$
  - Instantaneous reward function r_t
  - Policy $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$ maps states to probabilities over actions
  - Trajectory

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots, a_t, r_t, s_{t+1}, \ldots$$

$$a_t \sim \pi(s_t)$$

$$s_{t+1} \sim p(.|s_t, a_t)$$

Markov Decision Process

- Episodic MDP
  - State resets after T steps
  - Cumulative discounted reward $R = \sum_{t=0}^{T-1} \gamma^t r_t$
  - Optimal policy

$$\pi^* = \mathrm{argmax}_\pi \mathbb{E}[R \mid \pi]$$

- Non-episodic MDP
- Average return

Markov Decision Process

QuantAlea

- State value function
  - Expected return when starting in a state and following a policy

$$V^\pi(s) = \mathbb{E}[R \mid s, \pi]$$

- State action function
  - Expected return when starting in a state, taking an action and thereafter following a policy

$$Q^\pi(s, a) = \mathbb{E}[R \mid s, a, \pi]$$

$$V^\pi(s) = \max_a Q^\pi(s, a)$$

Markov Decision Process

- Dynamic programming
  - Bellman's equation

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma Q^{\pi}\big(s_{t+1}, \pi(s_{t+1})\big)\right]$$

  - Optimal Bellman equation

$$Q^{*}(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a'} \; Q^{*}(s_{t+1}, a')\right]$$

RL Challenges

QuantAlea

- Optimal policy must be inferred by trial and error interaction with environment
- Next observation depends on action of agent
- Observations may show strong temporal correlations
- Action often manifests itself after many state transitions
- Rewards may be sparse

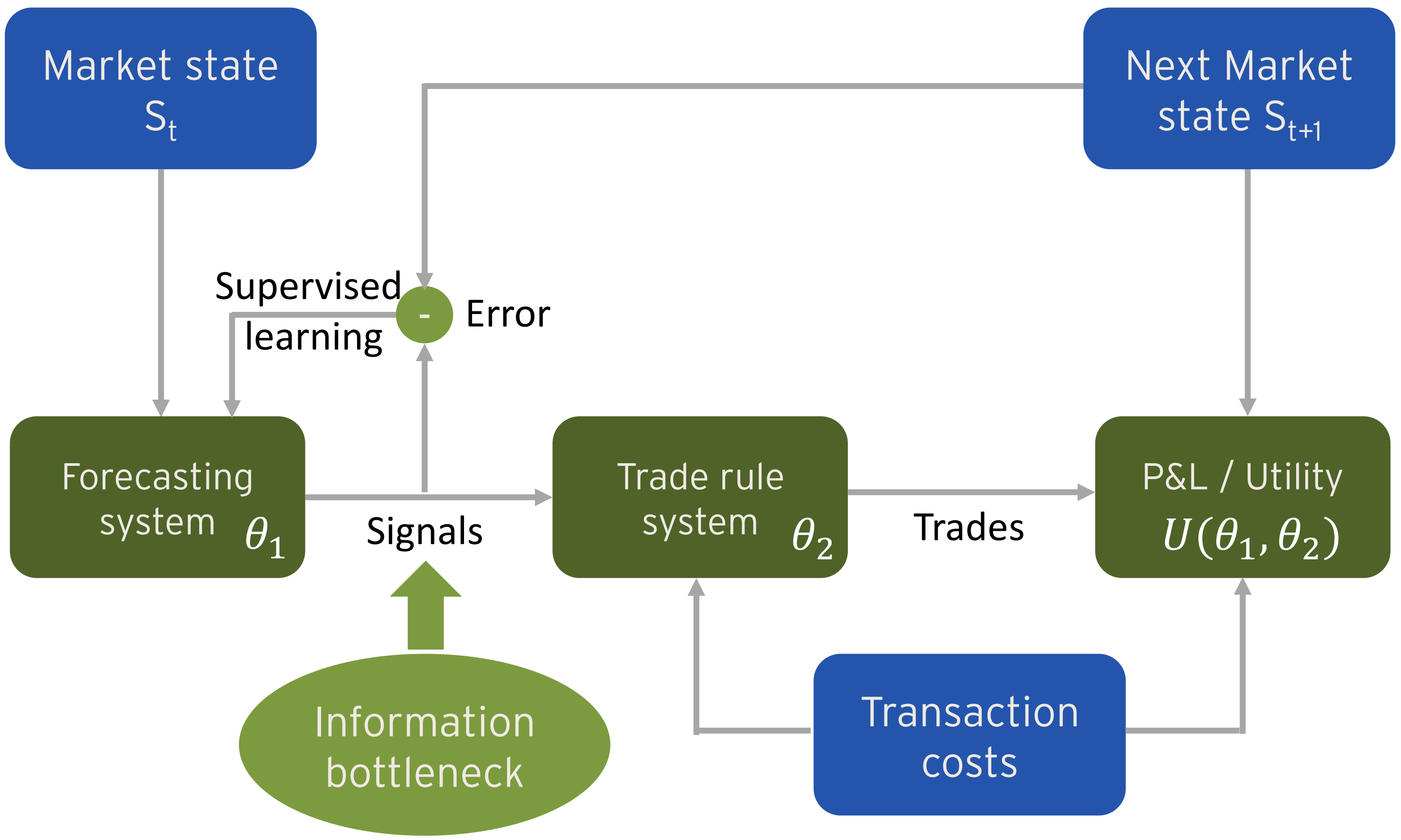
RL Algorithm Zoo

- Critic only
  - Q-learning
  - SARSA
- Actor only
  - Policy gradient
  - SRV
  - REINFORCE
- Actor-critic
  - TD(λ)
  - A2C, A3C
  - TRPO
  - PPO
  - DPPO

RL for Trading

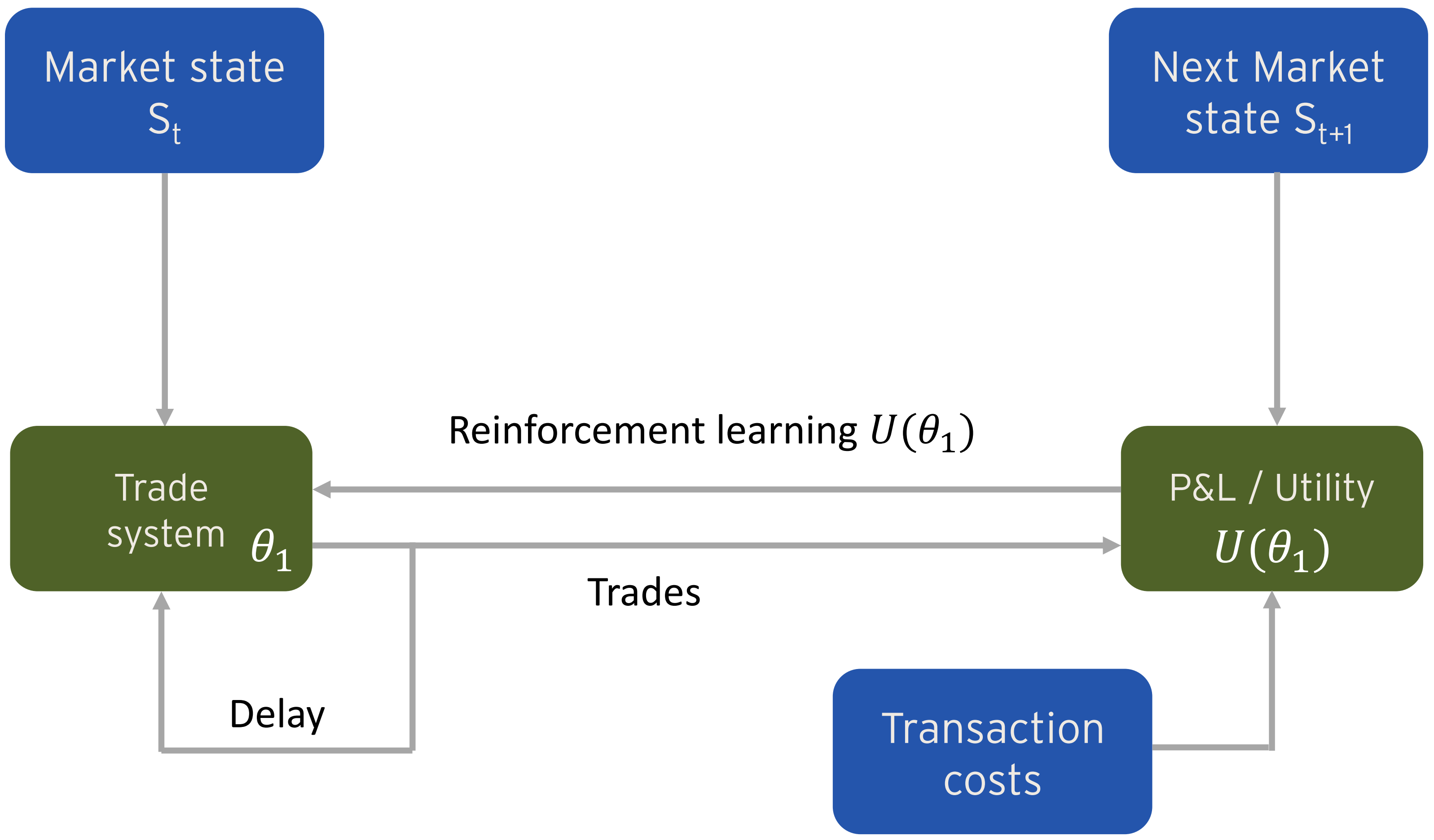
Why RL for Trading?

Signal based trading system



Why RL for Trading?

Reward driven trading system with reinforcement learning



Comparison

QuantAlea

Reinforcement learning based trading

- Single set of parameters
- Single utility function
- Utility includes transaction cost
- Direct mapping from market data to trades
- Easier to implement online updating
- Expensive model training

Signal based trading

- Parsimonious
- Two disconnected sets of parameters
- Supervised learning for signal forecast error is not utility
- Forecast system does not include transaction cost
- Two stage process causes information bottleneck

Challenges

- How to model the state of the market and a trading system?
- How to handle time series data, tick data, order book data, …?
- What additional time series data should be added?
- How to represent the policy? Discrete, continuous, hybrid…
- How to define the reward? Average, discounted
- How to include risk? Sharpe ratio, ….
- How to not forget catastrophic market events?
- How to incorporate additional constraints such as risk limits, …?
- Which algorithms to use for model training?

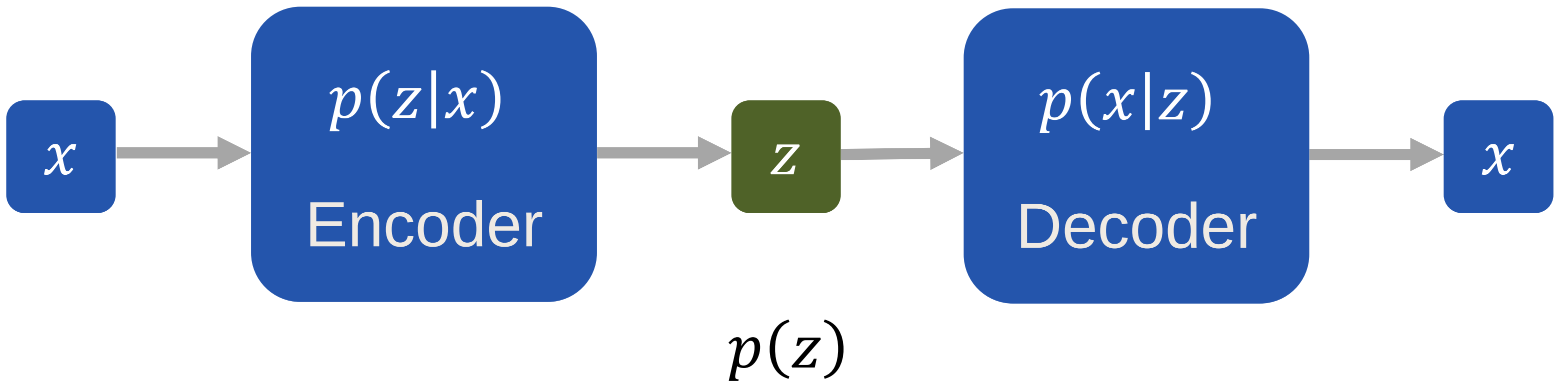
Generative Models

Generative Models

- Why generative models?
  - What I cannot create, I do not understand (Richard Feynman)
- Two main classes
  - GANs – Generative adversarial networks
  - VAE – Variational autoencoders

Latent Variable Model

- Latent variable z can be thought of a encoded representation of x
- Likelihood serves as decoder
- Posterior provides encoder



Fit with Maximum Likelihood?

QuantAlea

- Maximum likelihood standard model fitting approach

$$p(x) = \int p(x|z)\, p(z) dz \;\; \rightarrow \;\; \max$$

- **Problem**: marginal $p(x)$ and therefore also posterior

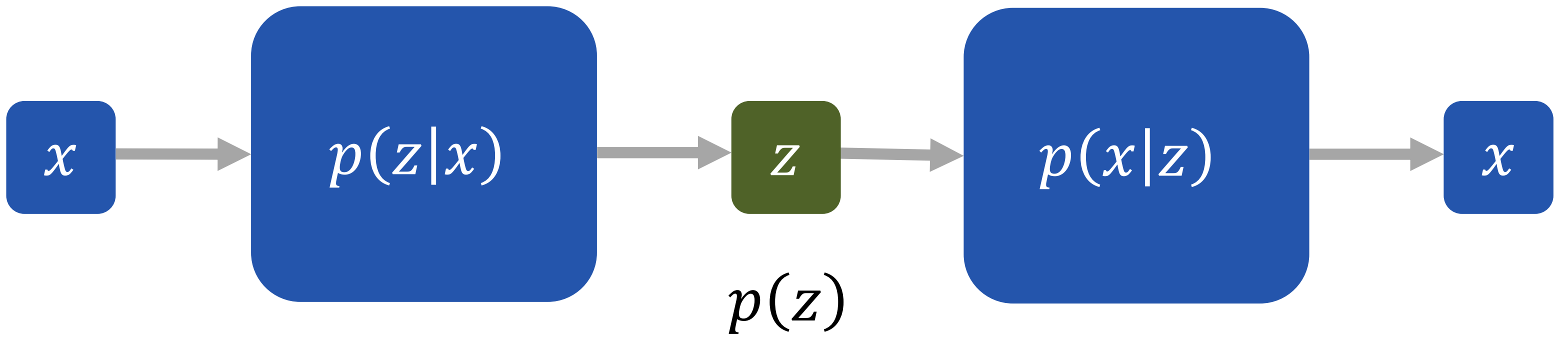
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

  are **intractable** and their calculation suffers from **exponential complexity**

- Solutions
  - Markov Chain MC, Hamiltonian MC
  - Approximation and variational inference

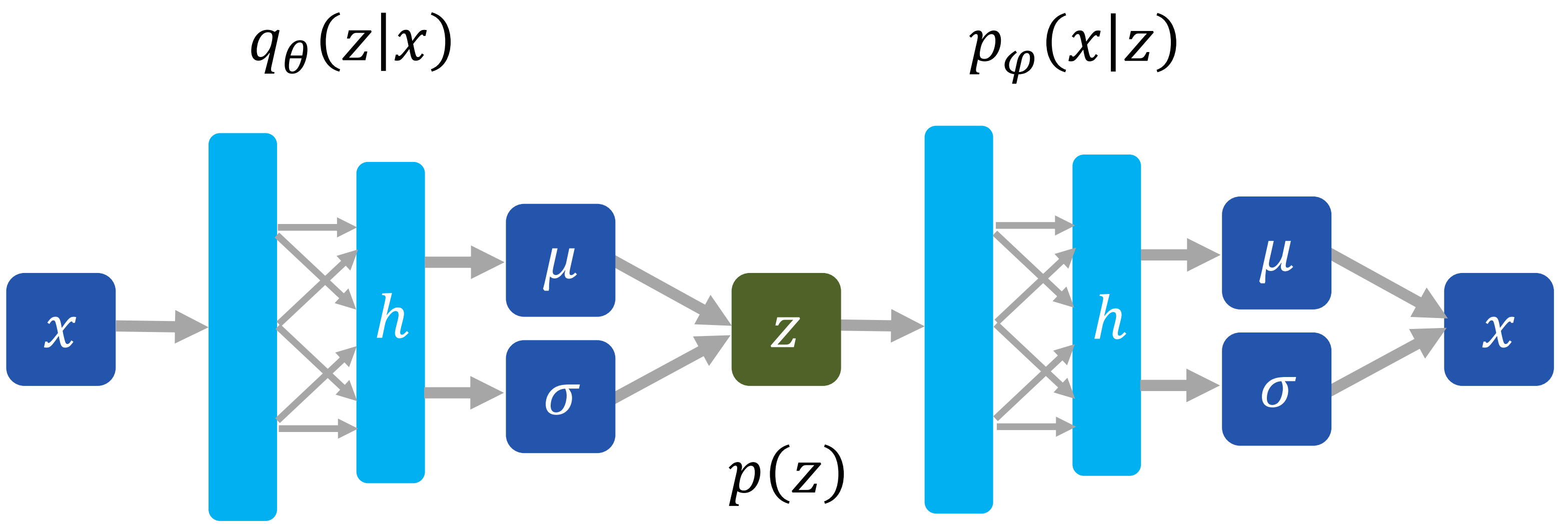
Variational Autoencoders

- Assume latent variable model with prior $p(z)$



Deep Variational Autoencoders

- Parameterize likelihood $p(x|z)$ with a deep neural network
- Approximate intractable posterior $p(z|x)$ with a deep neural network
- Learn the parameters θ and φ with backpropagation



Variational Inference

QuantAlea

- Which loss to optimize?
- It turns out that there is a computable lower bound for the log of the marginal:

$$\underbrace{E_{q_\theta(z|x)}\left[\log p_\varphi(x,z)\right] - E_{q_\theta(z|x)}\left[\log q_\theta(z|x)\right]}_{ELBO(\theta,\varphi)} \leq \log p_\varphi(x)$$

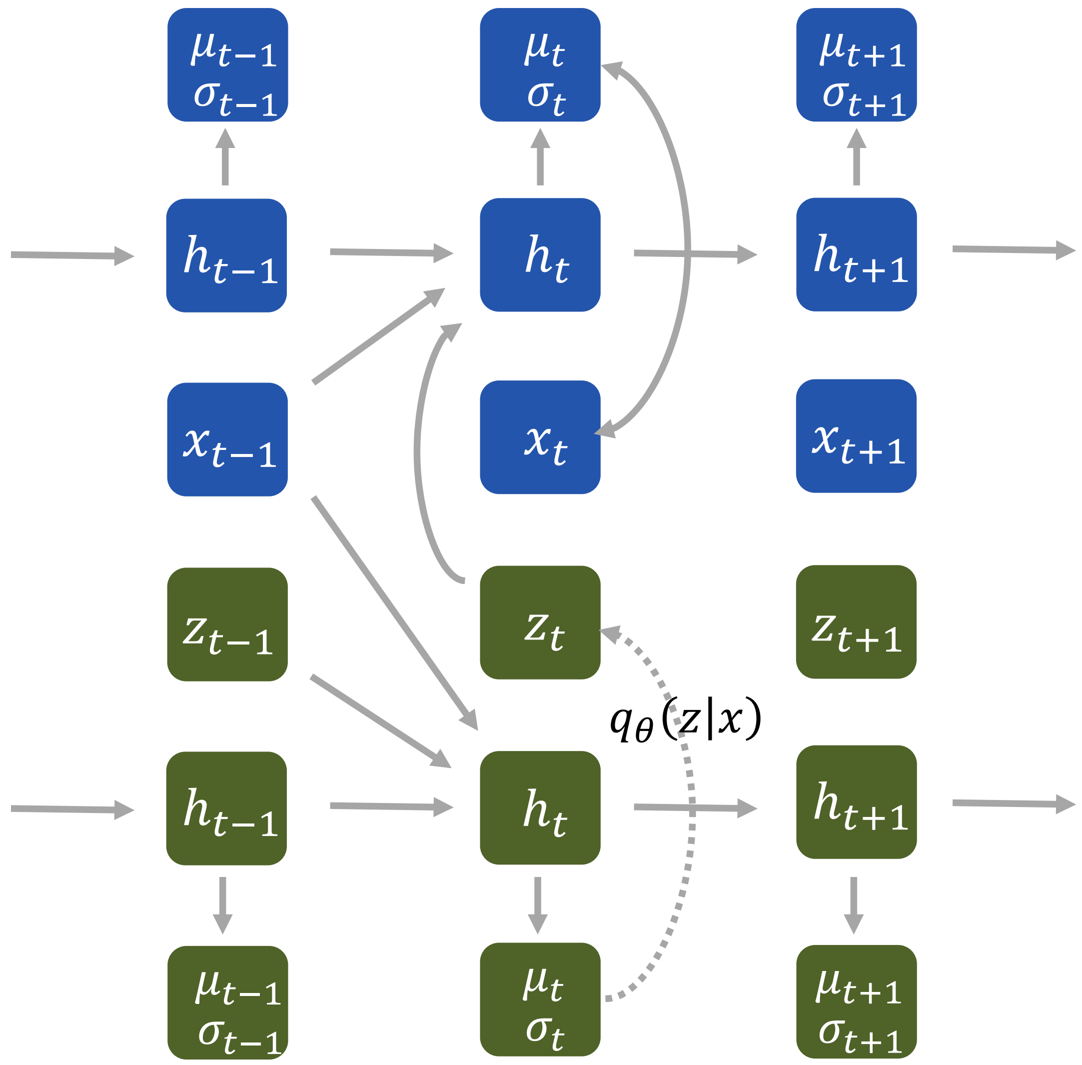
- Training criterion: maximize the evidence lower bound

$$ELBO(\theta,\varphi) \leq \log p_\varphi(x)$$

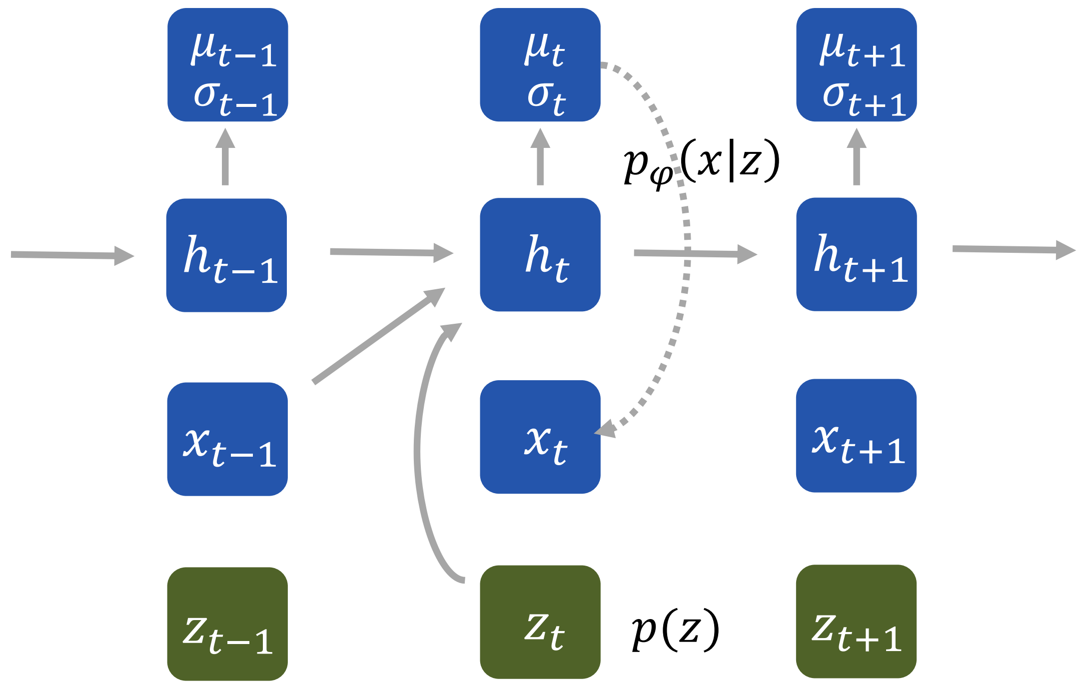
Applications to Time Series

- Sequence structure for observable and latent factor
- Model setup
  - Gaussian distributions with parameters calculated from deep recurrent neural network
  - Prior standard Gaussian
  - Model training with variational inference

Inference and Training

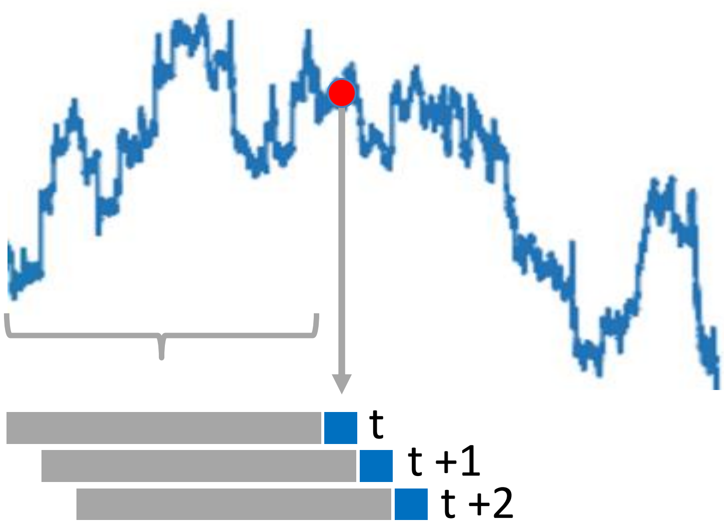


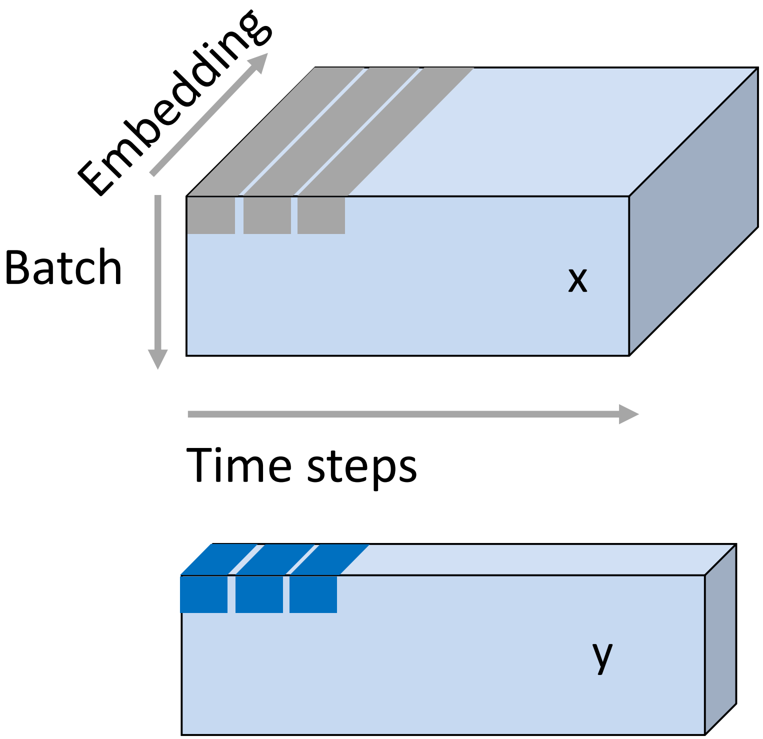
Generation


μ_{t-1} σ_{t-1}
μ_t σ_t
μ_{t+1} σ_{t+1}
$p_\varphi(x|z)$
h_{t-1}
h_t
h_{t+1}
x_{t-1}
x_t
x_{t+1}
z_{t-1}
z_t
$p(z)$
z_{t+1}

Time Series Embedding

- Single historical value not predictive enough
- Embedding
  - Use lag of 20 to 60 historical observations at every time step





TensorFlow Dynamic RNN

- Unrolling rnn with tf.nn.dynamic_rnn
  - Simple to use
  - Can handle variable sequence length
- Not flexible enough for generative networks

```
B = 3
D = 4
T = 5
PKEEP = 0.9

tf.reset_default_graph()

x = tf.placeholder(shape=[T, B, D], dtype=tf.float32)

with tf.variable_scope("RNN"):
    cell = tf.contrib.rnn.GRUCell(num_units = D)
    cell = tf.contrib.rnn.DropoutWrapper(cell, output_keep_prob = PKEEP)
    cells = tf.contrib.rnn.MultiRNNCell([cell])

    h, states = tf.nn.dynamic_rnn(cells, inputs = x, time_major=True, dtype=tf.float32)
```

TensorFlow Control Structures

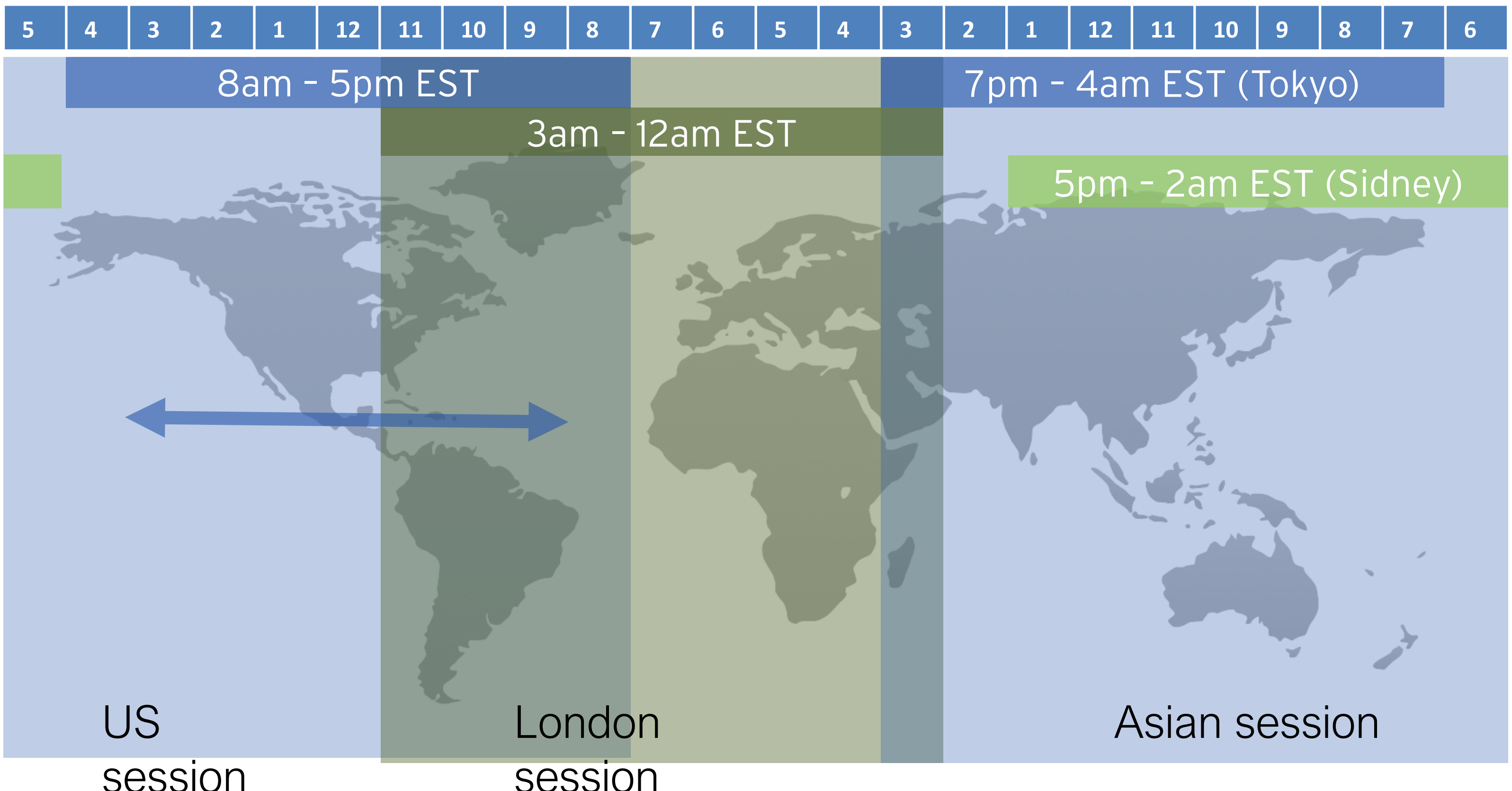
- Using tf.while_loop
  - More to program, need to understand control structures in more detail
  - Much more flexible

```
x = tf.placeholder(shape=[T, B], dtype=tf.float32)

output_ta = tf.TensorArray(size=T, dtype=tf.float32)
input_ta = tf.TensorArray(size=T, dtype=tf.float32)
input_ta = input_ta.unstack(x)

def body(time, output_ta):
    xt = input_ta.read(time)
    output_ta = output_ta.write(time, tf.reduce_sum(xt**2))
    return (time+1, output_ta)

time_final, output_ta_final = tf.while_loop(
      cond=lambda time, *_ : time < T,
      body=body,
      loop_vars=(time, output_ta))

output_final = output_ta_final.stack()
```

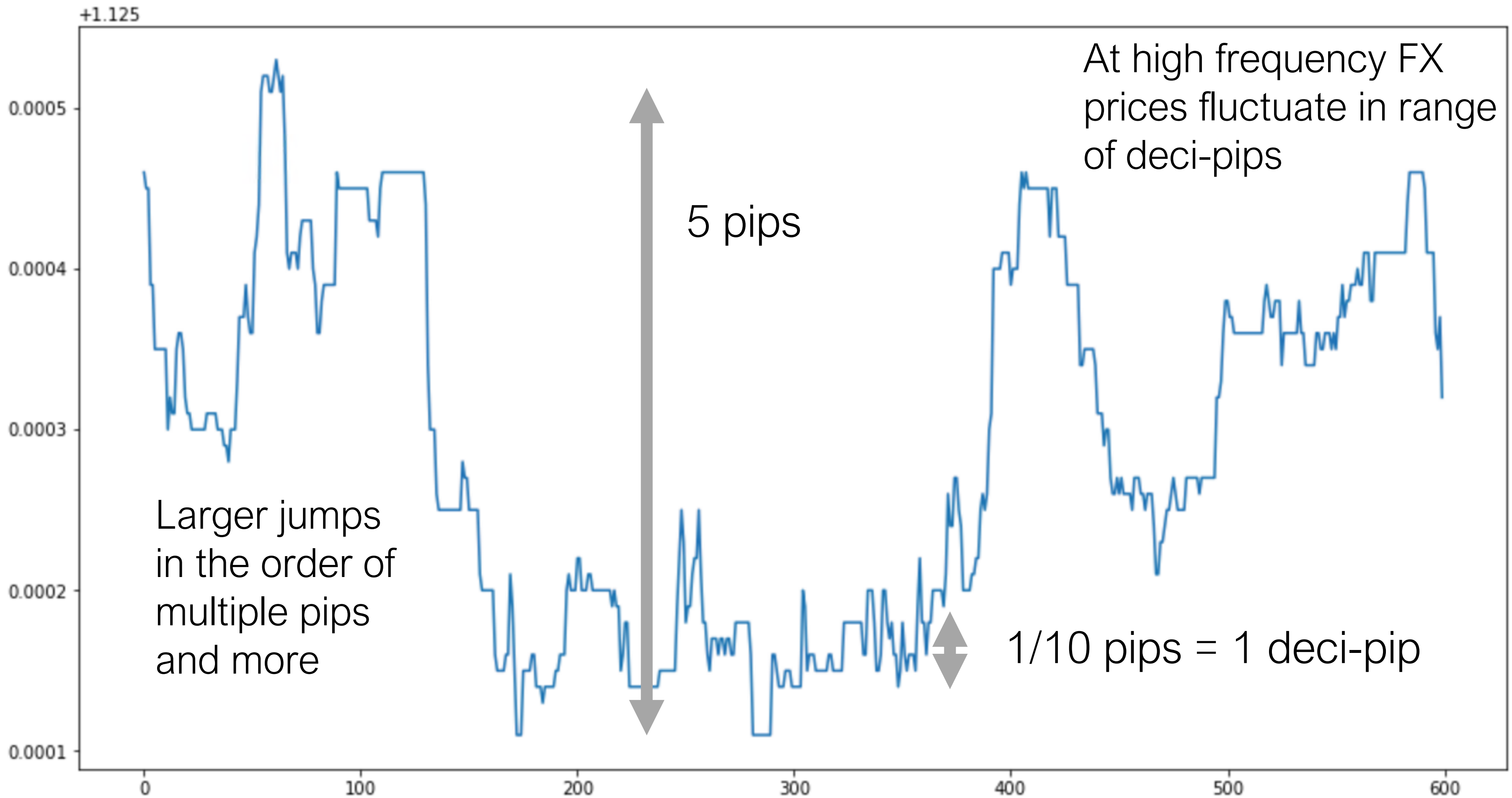
Application to FX Data

FX Data

- Collect tick data from major liquidity provider e.g. LMAX
- Aggregation to OHLC bars (1s, 10s, …)
- Focus on US trading session

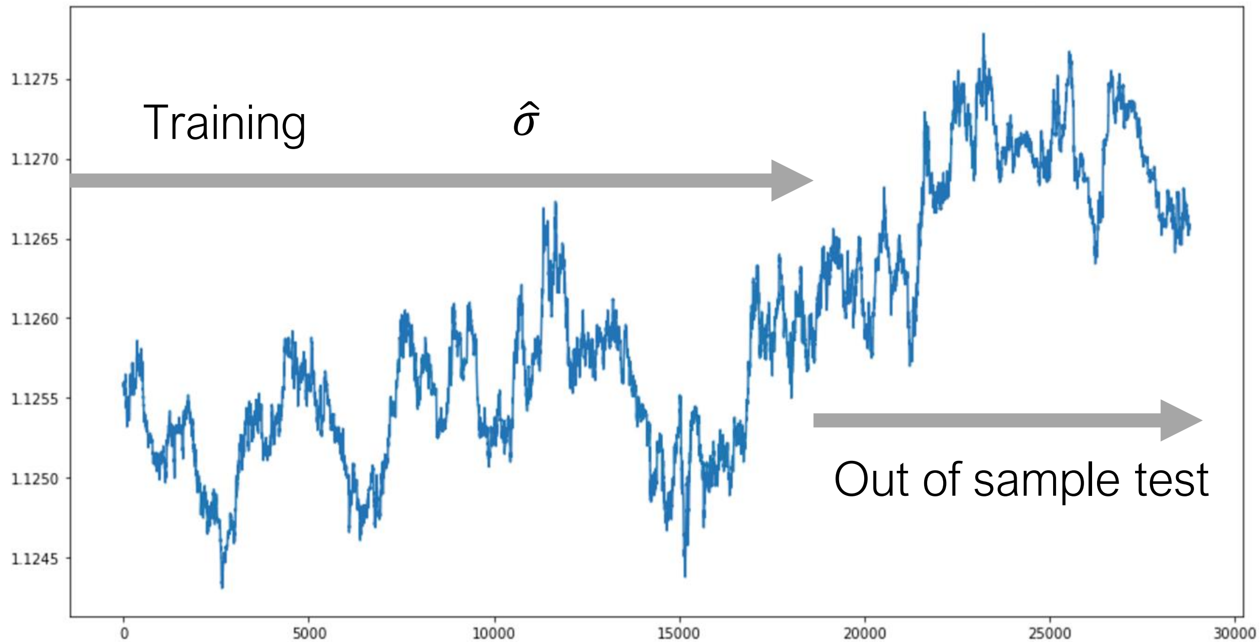


10 Min Sampled at 1s

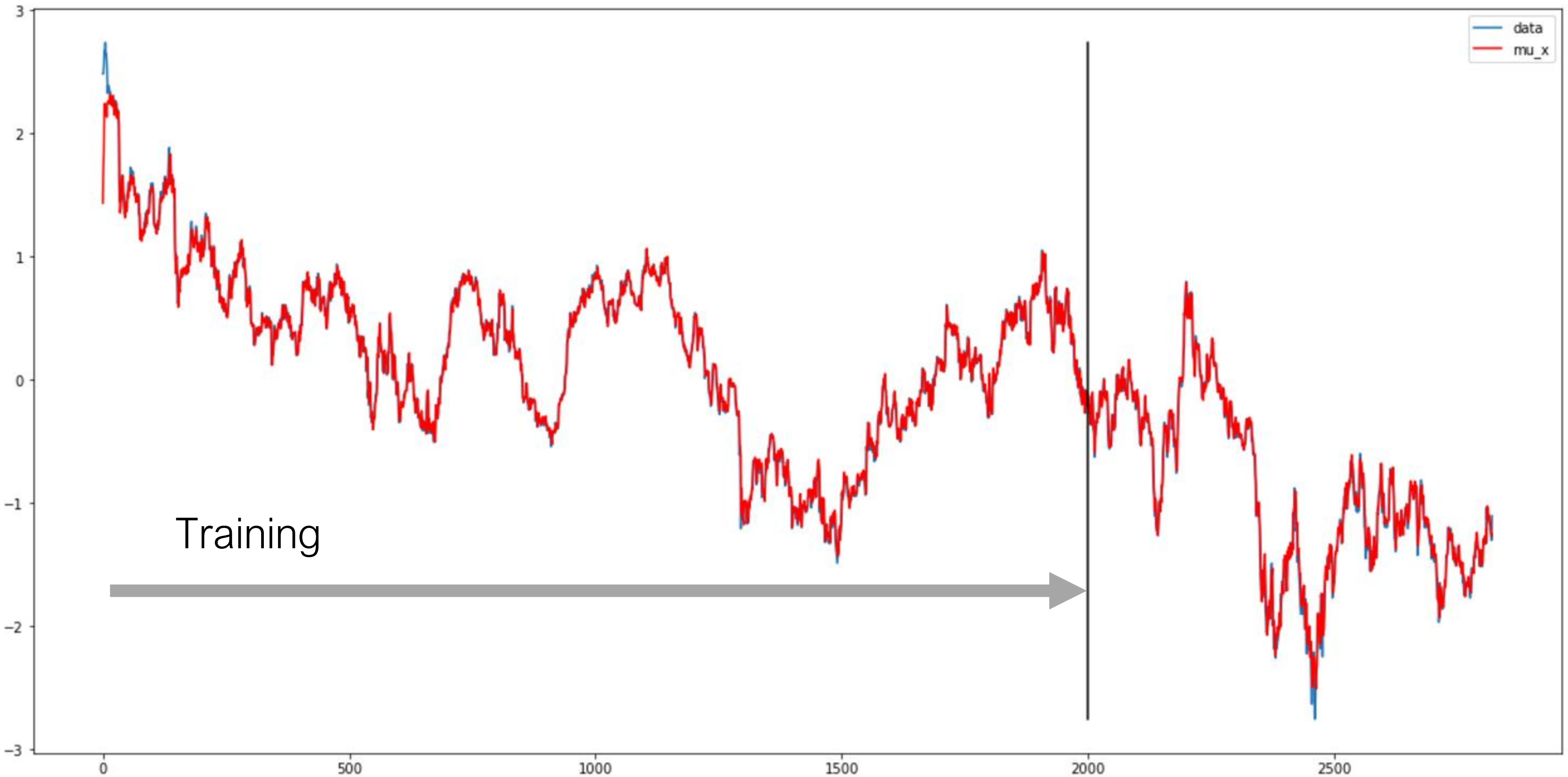


Setup

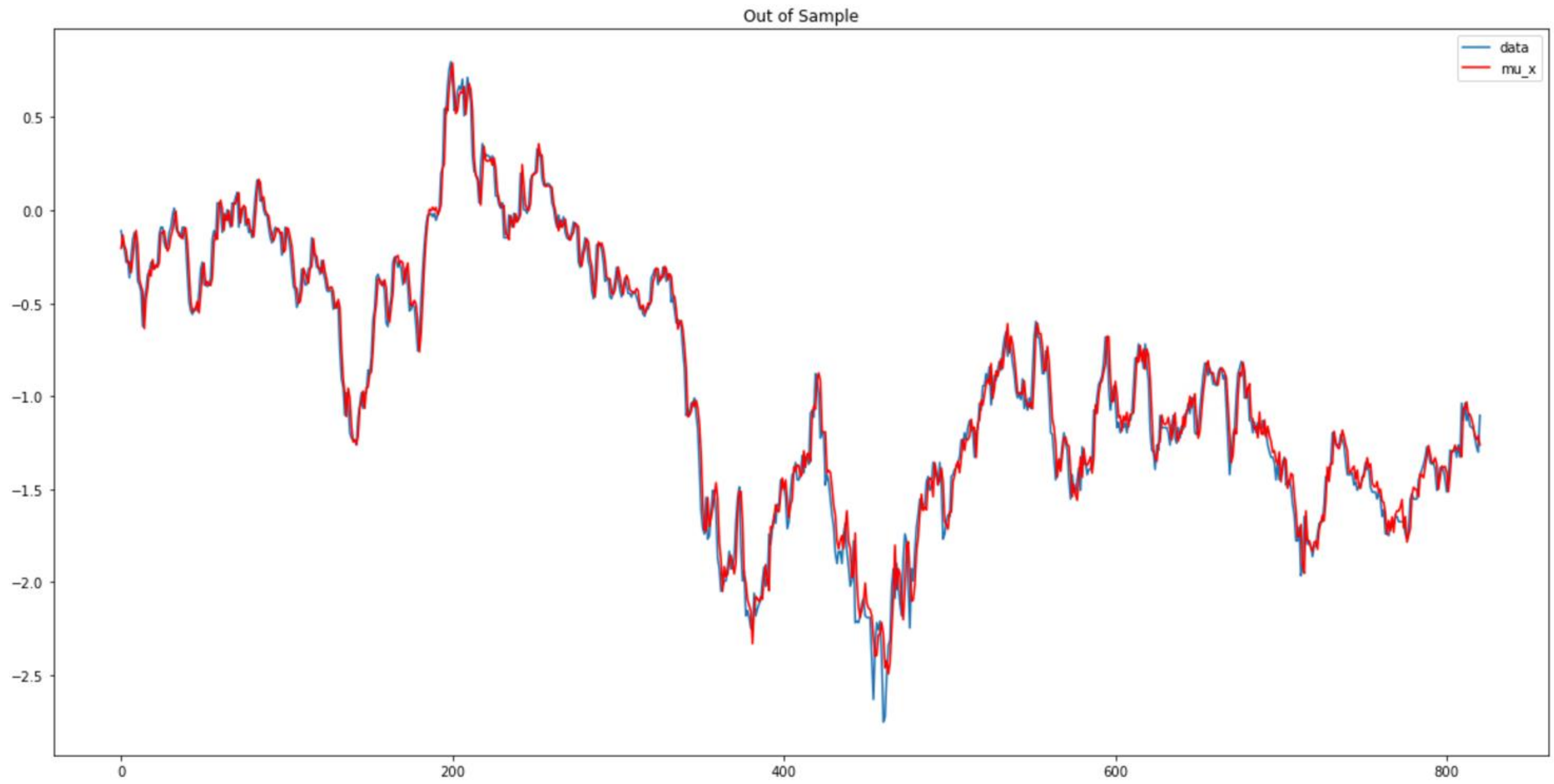
- Normalize data with std deviation $\hat{\sigma}$ over training interval
- 260 trading days in 2016, one model per day
- 60 dim embedding, 2 dim latent space



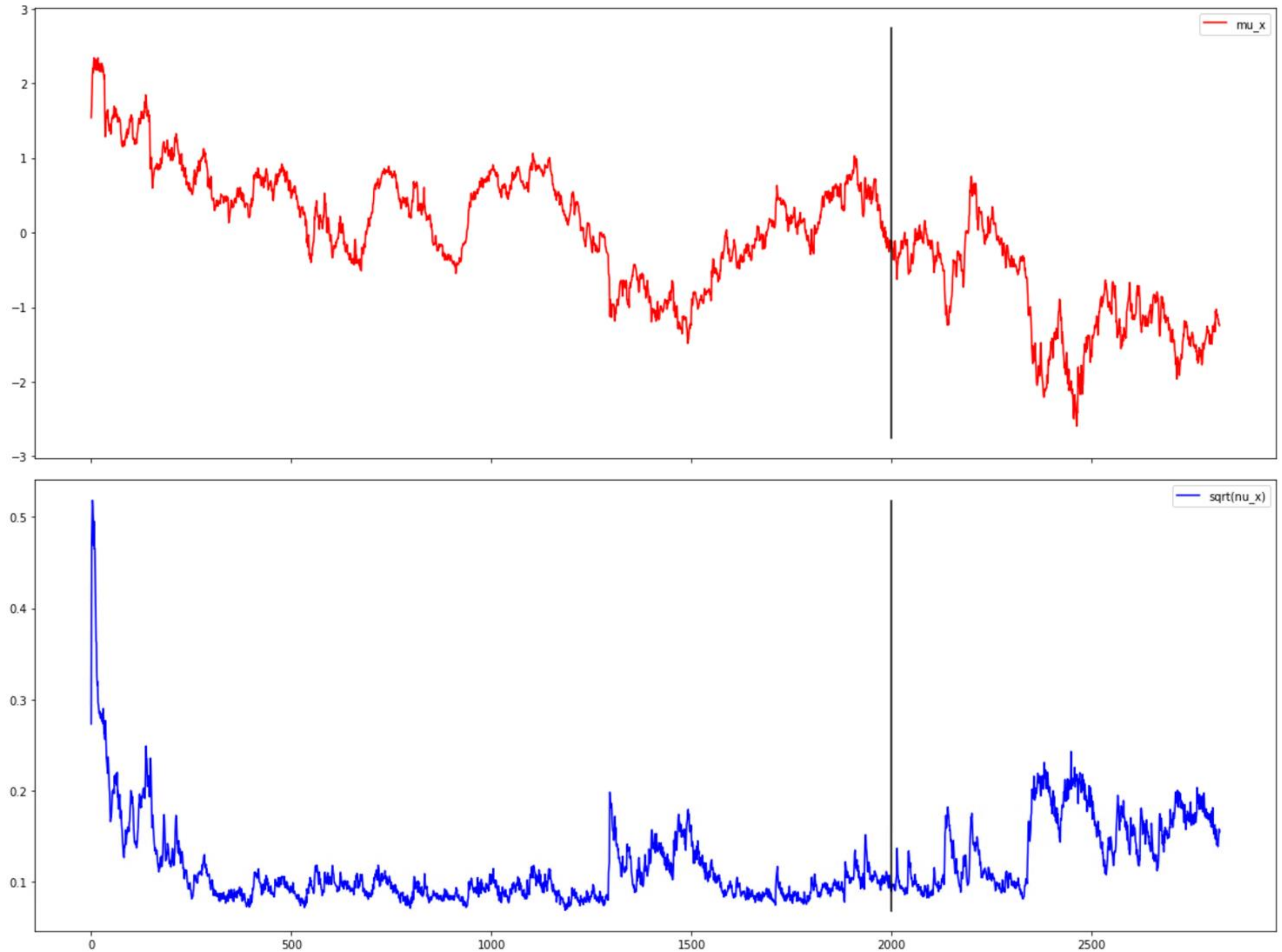
Results



Out of Sample



Volatility of Prediction



@EgloffDaniel
@QuantAlea

Daniel Egloff
Dr. sc. math.
Phone: +41 79 430 03 61
daniel.egloff@quantalea.net