Research Plan for CSE3000 Research Project

The Use of Reinforcement Learning in Algorithmic Trading: Evaluating Function Approximation Methods

Robert Mertens

April 27, 2025

Background of the research

Algorithmic trading in financial markets has increasingly adopted artificial intelligence techniques, including reinforcement learning (RL), to develop adaptive trading strategies. RL-based systems enable agents to make sequential decisions under uncertainty by interacting with dynamic environments. Recent literature has explored deep RL models, yet challenges remain in understanding how different function approximation methods influence trading performance, stability, and generalization across market regimes [1, 3, 2].

Function approximation plays a pivotal role in RL when dealing with high-dimensional or continuous state spaces, where tabular methods become infeasible. Various approaches such as linear function approximation, non-linear approximators (e.g., kernel methods), and neural networks (deep RL) have been proposed. While deep RL has shown promise, issues such as overfitting, sample inefficiency, and instability under noisy financial data persist [4]. There is a lack of comparative analysis under controlled trading environments that evaluates these approximators using unified baselines and metrics.

This project aims to fill that gap by comparing function approximation methods in the context of Forex trading using a shared RL framework and dataset, contributing empirical insights into their respective trade-offs.

Research Question

The main research question for the project is: **"How do different function approximation methods impact the performance of a Reinforcement Learning model trading in the Forex Market?"** We divide into the following sub-questions:

- How do function approximation methods
- Under what market conditions do simpler models (e.g., linear approximators) outperform more complex ones?
- How does model complexity affect convergence speed, stability, and sensitivity to hyperparameters?
- What is the comparative performance in terms of risk-adjusted return metrics (Sharpe Ratio, Sortino Ratio)?
- How does the sample efficiency vary across function approximators given the same training data?

After answering the aforementioned questions we expect to end up with a detailed comparison of the performance of function approximation methods, using standardized data and metrics, visualized through plots such as performance curves and Sharpe ratio histograms.

Method

Software and Libraries

The core implementation will utilize Python 3. Key libraries include:

- **FinRL**: An open-source library tailored for applying Reinforcement Learning to financial trading. Its modular structure enables easy customization of environments, algorithms, and backtesting workflows. FinRL will be adapted to allow consistent experimentation across different function approximation methods.
- **Stable-Baselines3**: Provides reliable implementations of common RL algorithms (e.g., DQN, PPO), facilitating integration with FinRL.
- **Pandas**: For data manipulation and preparation.
- **NumPy**: For efficient numerical computation.
- **Matplotlib/Seaborn**: For visualization of training progress and trading results.
- **TA-Lib (or similar)**: For generating technical indicators as additional features, if necessary.
- **Gymnasium**: For environment standardization and RL training interfaces.

Data Collection and Preparation

- **Source**: Historical Forex data will be collected via the OANDA API or downloaded from other publicly available sources.
- **Instrument**: Focus will be on EUR/USD, aligning with common practices in existing literature.
- **Granularity**: 15-minute (M15) candlesticks will be used, maintaining a low-frequency trading context.
- **Time Period**: Data will span from 2023-01-01 to 2025-01-01.
- **Splitting**: The dataset will be divided chronologically into training, validation (optional, for tuning), and testing sets to evaluate generalization.

Baseline RL Model Configuration

To focus on the impact of different function approximation methods, a standardized RL setup will be defined:

- **Algorithm**: A stable off-policy algorithm such as DQN will be used as the base model.
- **Trading Environment**: The trading environment (action space, reward structure, state features) will be kept constant across experiments.
- **Hyperparameters**: Initial hyperparameters will be kept fixed, based on validated defaults from previous financial RL studies, unless tuning is deemed necessary for stability.

Experiments

The experiments will compare function approximation methods across different algorithms, changing the trading environment, action space, reward function, and hyperparameters only for as much as necessary. Each method will involve training the agent on the training set and evaluating performance on an unseen test set. This means comparing the rows from following table:

Function Approximation	**Value-Based (Q-learning)**	**Policy-Based (PPO)**	**Actor-Critic (A2C, A3C, SAC, DDPG, TD3)**
Tabular	Tabular Q-learning: table of Q-values	N/A	N/A
Linear Function Approximation	Linear Q-learning: $Q(s,a) = w^T \phi(s,a)$	Linear Policy Gradient: $\pi(a\|s) = \text{softmax}(w^T \phi(s))$	Linear Actor-Critic: linear value and policy functions
Non-linear Function Approximation	Non-linear Q-learning: e.g., decision trees, kernels	Non-linear Policy Optimization: e.g., kernelized policies	Non-linear Actor-Critic: non-linear value and policy functions
Deep Reinforcement Learning	DQN: Deep Q-Network (deep neural net for Q-values)	PPO: Proximal Policy Optimization (deep neural net for policy)	A2C, A3C, SAC, DDPG, TD3: Deep Actor-Critic algorithms

Table 1: Combination of Function Approximation Methods and Reinforcement Learning Algorithms

Performance Metrics

Agent performance will be assessed using standard financial and statistical metrics:

- **Sharpe Ratio**: Primary measure of risk-adjusted return.
- **Cumulative Returns**: Total profitability over the test set.
- **Sortino Ratio**: Focused on downside volatility.
- **Maximum Drawdown**: Largest loss from a peak, indicating risk.
- **Win Rate and Average Risk-Reward Ratio**: To better understand trade behavior quality.
- **Permutation Tests**: To determine if observed differences are statistically significant rather than random.

Analysis

The analysis will proceed as follows:

- **Comparative Tables**: Tabulating key metrics across all function approximation methods.
- **Visualizations**: Plotting equity curves, cumulative returns, and metric comparisons across different methods.
- **Statistical Testing**: If multiple training runs are feasible, statistical tests (e.g., permutation tests) will be used to verify the significance of performance differences.
- **Discussion**: Results will be interpreted to evaluate which function approximation techniques are most effective for financial trading tasks. Observed trade-offs between model complexity, stability, and performance will be discussed, along with limitations and directions for future research.

Tools and Collaboration

Git will be used for version control and collaboration. Tasks are divided into data collection, environment setup, model implementation, training, evaluation, and analysis. Task completion will be evaluated based on predefined criteria (e.g., successful training convergence, statistically valid evaluation results).

Planning of the Research Project

Deliverables & Course Activities

1. **Research Plan (Draft)** – 22 April 23:59
2. **Research Plan (Text)** – 27 April 23:59
3. **Research Plan (Presentation Slides)** – 27 April 23:59
4. **Midterm Presentation** – 19 May
5. **Midterm Poster** – 20 May 23:59
6. **Final Paper (Draft v1)** – 02 June 23:59
7. **Final Paper (Draft v2)** – 10 June 23:59
8. **Final Paper** – 22 June 23:59
9. **Final Poster** – 23 June 23:59

Meetings

1. 23 April 12:00–13:00 (13:30) – Supervisor Meeting – **IN PERSON** – Room-34.C3.170
2. 30 April 12:00–13:00 (13:30) – Supervisor Meeting – **IN PERSON** – Room-34.C3.170
3. 9 May 12:00–13:00 – Supervisor Meeting – **IN PERSON** – TBD
4. 14 May 12:00–13:00 (13:30) – Supervisor Meeting – **IN PERSON** – Room-34.C3.170
5. 21 May 12:00–13:00 (13:30) – Supervisor Meeting – **IN PERSON** – Room-34.C3.170
6. 28 May 13:00–14:00 – Supervisor Meeting – **IN PERSON** – TBD
7. 4 June 12:00–13:00 (13:30) – Supervisor Meeting – **IN PERSON** – Room-34.C3.170
8. 11 June 12:00–13:00 (13:30) – Supervisor Meeting – **IN PERSON** – Room-34.C3.170
9. 18 June 12:00–13:00 – Supervisor Meeting – **ONLINE** – Microsoft Teams

Estimated Weekly Planning

1. **Week 1**: Familiarize with the topic, the field, the project, and create the research plan.
2. **Week 2**: Analyze past research to decide on a baseline RL Model, decide on some experiments, and set up the codebase.
3. **Week 3**: Run experiments, do analysis, design new experiments, and analyze additional previous research.
4. **Week 4**: Run additional experiments, do additional analysis, draw further conclusions, design further experiments, and analyze more research. Also, prepare the midterm poster summarizing methods, experiments, and findings up to this point.

5. **Week 5**: Present findings so far and perform additional experiments based on feedback received during the midterm presentations.
6. **Week 6**: Run final experiments, complete final analysis, make conclusions, and write the draft paper.
7. **Week 7**: Review a peer's paper and incorporate peer feedback into the final paper. Additional analysis and results can still be added.
8. **Week 8**: Finalize the paper and incorporate any received feedback.
9. **Week 9**: Continue finalizing the paper and incorporate any further feedback.
10. **Week 10**: Conduct the final presentations.

Each week should also include summarizing findings and data, and incorporating this into the paper.

A Terminology

- **Linear Function Approximation**: An RL method where Q-values (or value functions) are approximated as a weighted sum of features. Scales better than tabular methods, but may lack expressiveness in complex environments.
- **Non-linear Function Approximation**: Uses non-linear models (e.g., decision trees, kernel methods) to approximate value or policy functions. Captures more complex relationships but can be harder to train.
- **Deep Reinforcement Learning** (Deep RL): Combines deep learning (typically neural networks) with RL algorithms. Effective in large, high-dimensional state spaces (e.g., images, financial data).
- **Policy**: mapping of states to actions
- **Deterministic**: always selects the same action in a state
- **Stochastic**: samples actions from a probability distribution
- **Value-based methods**: we compute the value of each state-action combination
- **Policy-based methods**: learn a policy directly, rather than indirectly via a value function
- **Policy-gradient methods**: subset of policy-based methods that use gradient descent on the expected return to
- **Actor-critic methods**: combines value and policy based methods
- **Model-free vs. model-based**: Model-free methods learn directly from interactions without understanding the environment's dynamics, while model-based methods build a model of the environment to plan ahead.
- **On-policy vs. off-policy**: On-policy methods learn from the actions taken by the current policy, while off-policy methods learn from actions taken by a different policy (e.g., a past or exploratory one).
- **Algorithm**: the overall procedure used to learn how to behave optimally. It defines how the agent updates its policy or value estimates based on experience.
- **Function approximation method**: refers to how the agent represents and estimates complex functions like the value function, action-value function, or policy.
- **Tabular Q-learning**: A basic RL approach where each state-action pair has an explicitly stored Q-value in a table. Suitable for small, discrete state-action spaces.

- **PPO**: Proximal Policy Optimization, policy gradient method that clips the policy updates to increase stability.
- **DQN**: Deep Q-network, Q-learning method that replaces the Q-table with a neural network. Value-based, Good for discrete action spaces (Buy/Sell/(Hold)), Sample efficient: Deep Q-Learning
- **A2C**: Actor-Critic, Learns both the policy (actor) and value function (critic).
- **A3C**: async version of A2C. Works well with continuous input
- **SAC**: Actor-Critic. Maximizes both reward and entropy(keeps exploring). Highly stable, handles continuous action.
- **DDPG**: Actor-Critic. Like Q-learning for continuous actions. Works with continuous action space. Very sensitive to hyperparameters. Hard to train. Lowkey outdated.
- **TD3**: Actor-Critic. Improvement upon the DDPG model.
- **State Representation**: How the environment's current situation is encoded for the agent. A good state representation captures all relevant information needed for decision-making.
- **Dimensionality Reduction**: Techniques like PCA or autoencoders used to reduce state complexity while preserving essential information.
- **Feature Engineering**: The process of selecting or transforming raw data into informative inputs for the RL model (e.g., technical indicators, volatility, price momentum).
- **Timeframe**: the duration that each data point (or candlestick) represents on a chart. It determines the granularity of the data the RL agent learns from
- **Candlestick**: a visual representation of price movement over a specific timeframe. It provides four key pieces of information: Open: Price at the beginning of the timeframe. High: Highest price during the timeframe. Low: Lowest price during the timeframe. Close: Price at the end of the timeframe. Volume: Total transaction amount during the timeframe. (sometimes included in the data)
- **Reward Shaping**: Modifying the reward function to make learning more efficient (e.g., adding penalties for high drawdown or excessive trading).
- **Sparse vs. Dense Rewards**: Sparse: Rewards occur infrequently (e.g., only at episode end). Dense: Frequent feedback (e.g., after every action), often improves learning speed.
- **Risk-sensitive Reward Functions**: Incorporate metrics like Sharpe Ratio, Sortino Ratio, or drawdown into rewards to align learning with risk-adjusted return goals.
- **Epsilon-Greedy**: With probability ϵ, the agent explores (random action), otherwise exploits (best-known action). Simple but effective.
- **Boltzmann (Softmax) Exploration**: Chooses actions probabilistically based on their Q-values, with higher-valued actions more likely. Controlled by a temperature parameter.
- **Upper Confidence Bound (UCB)**: Balances exploration and exploitation using confidence intervals; actions with uncertain (but potentially high) value are explored more.
- **Thompson Sampling**: Probabilistic exploration using a Bayesian approach to sample from the posterior of action-value distributions.
- **Transfer Learning**: Leveraging knowledge (e.g., learned policies, Q-values, or features) from one task (e.g., a currency pair or market regime) to improve learning in another.
- **Policy Transfer**: Using a policy trained in one environment as a starting point in another.
- **Q-table Transfer**: Reusing Q-values for similar state-action pairs in a new task.
- **Domain Adaptation**: Adjusting the model to handle differences between source and target environments (e.g., volatility, spread behavior)

References

[1] Ben Hambly, Ruochen Xu, and Haoyang Yang. “Recent advances in reinforcement learning in finance”. In: *Mathematical Finance* 33.3 (2023), 437â503. DOI: `10.1111/mafi.12382`.

[2] Chien-Yi Huang. “Financial Trading as a Game: A deep reinforcement learning approach”. In: *arXiv preprint arXiv:1807.02787* (2018). URL: `https://arxiv.org/abs/1807.02787`.

[3] Tudor Pricope. “Deep Reinforcement Learning in Quantitative Algorithmic Trading: a review”. In: *arXiv preprint arXiv:2106.00123* (2021). URL: `https://arxiv.org/abs/2106.00123`.

[4] Thomas ThÃ©ate and Damien Ernst. “An application of deep reinforcement learning to algorithmic trading”. In: *Expert Systems with Applications* 173 (2021), p. 114632. DOI: `10.1016/j.eswa.2021.114632`.