Detecting Suboptimal American Option Exercises using a Heterogeneous Temporal Graph Neural Network

John Atwood*
CS 782 Project

Abstract

American-style stock options allow holders to exercise at any time before expiration, introducing complex decision-making and the potential for suboptimal (premature) exercises. We present a framework to detect and analyze such suboptimal option exercise decisions using a heterogeneous temporal graph neural network. We integrate real historical market data with synthetic trader behavior simulation to construct a temporal graph of trader-option-asset interactions. A Graph Convolutional GRU (GConvGRU) model is trained on this graph to predict whether an early exercise is suboptimal (incurs regret) and to estimate the degree of regret. Experiments on simulated trades across 10 S&P-500 stocks demonstrate the model's ability to learn temporal patterns of suboptimal decisions. My approach offers a novel graph-based perspective on option exercise behavior, with implications for trader education, risk management, and the design of decision support tools.

1 Introduction

American options, unlike their European counterparts, can be exercised at any time up to expiration. This flexibility introduces a challenge: traders may exercise the option at a suboptimal time, thereby sacrificing potential profit. A suboptimal exercise typically means the trader exercised (sold) the option too early, missing out on gains they could have realized by holding longer. Such behavior is not merely hypothetical; real incidents illustrate the cost of premature exercises. For example, in 2015 a trader at Interactive Brokers reportedly incurred millions in losses by exercising American call options just before an earnings announcement, only to see the stock soar afterwards:contentReference[oaicite:3]index=3. Understanding and predicting these regrettable decisions can provide valuable feedback to traders and help brokers or automated strategies mitigate avoidable losses.

*Department of Computer Science, George Mason University

Traditional option pricing theory establishes conditions for optimal exercise. In an ideal, frictionless market, it is almost never optimal to prematurely exercise a call option on a non-dividend-paying stock:contentReference[oaicite:4]index=4, yet in practice, traders often do so. Behavioral biases (like risk aversion or impatience) and imperfect information can lead to systematic suboptimal exercises. My goal is to detect when an exercise is likely suboptimal and quantify the *regret*, defined as the profit foregone by not waiting.

Contributions: We propose a novel approach to model option exercise decisions as a temporal heterogeneous graph and apply a Graph Neural Network (GNN) to predict suboptimal behavior. We simulate realistic trading scenarios on historical market data for multiple sectors, capturing diverse market conditions. We introduce a labeling scheme that includes both binary classification (optimal vs. suboptimal exercise) and a continuous regret score. My GNN model leverages the evolving relationships between a trader, the option contract, and the underlying stock over time using a Graph Convolutional Recurrent Network. We demonstrate that this model can learn complex temporal patterns and provide meaningful predictions. To my knowledge, this is the first application of temporal GNNs to the analysis of American option exercise behavior, bridging techniques from financial modeling, time-series analysis, and graph machine learning.

2 Related Work

American Option Valuation and Exercise: The problem of optimally exercising American options has long been studied in quantitative finance. The classic approach by Longstaff and Schwartz (2001) uses a Least Squares Monte Carlo (LSM) method to estimate the continuation value of holding an option, thereby determining optimal exercise policy through backward induction:contentReference[oaicite:5]index=5. While effective for pricing, such methods assume rational exercise. Empirical studies, however, document deviations from optimal exercise: Aretz et al. (2020) find that investors often delay exercising American put options even when it's suboptimal, creating arbitrage opportunities:contentReference[oaicite:6]index=6:contentReference[oaicite:7]index=7. Conversely, early exercises of calls (especially without dividends) represent foregone time value and are typically suboptimal:contentReference[oaicite:8]index=8. These works highlight that suboptimal option behavior exists in practice, motivating predictive models that can flag such decisions.

Machine Learning for Options: Recent research has applied machine learning to option pricing and strategy. Djagba and Ndizihiwe (2024) explore pricing American options using machine learning algorithms:contentReference[oaicite:9]index=9, showing that data-driven approaches can approximate option values and exercise boundaries. Their focus is on valuation accuracy rather than identifying human suboptimal behavior. My work, by contrast, concentrates on the decision-making aspect, treating the exercise event as a prediction target (optimal vs not) rather than pricing the option itself.

There is also emerging interest in using reinforcement learning to learn exercise Continuing the LaTeX report from the Related Work section:

latex There is also emerging interest in using reinforcement learning (RL) to derive exercise policies, treating the exercise decision as an optimal stopping problem that an RL agent can learn. These approaches (e.g., using deep Q-networks or policy gradients) can in principle learn when to exercise for maximizing reward, but they do not explicitly address predicting human suboptimal behavior — instead, they attempt to find the optimal strategy.

Graph Neural Networks in Finance: Graph-based deep learning has gained traction in financial modeling where relational structure is important. Recent work by Hu *et al.* (2023) introduced a *heterogeneous continual GNN* for futures trading, which simultaneously models multiple financial tasks (like price movement prediction and change-point detection) over an evolving graph of market entities [1]. Their results show improved performance by capturing the relationships between heterogeneous nodes (contracts, indicators, etc.) and adapting over time. We draw inspiration from such heterogeneous graph approaches. However, my setting is different: rather than multi-task prediction on known optimal outcomes, we focus on a single task of classifying suboptimal decisions.

Graph Neural Networks have also been applied to model interactions in multi-agent systems and financial markets. For example, graph architectures have been used to represent correlations among stocks for stock price prediction [6], and to model agent interactions in trading simulations. We extend these ideas by incorporating a temporal dimension (evolving graph) and multiple entity types.

Temporal GNN Architectures: My model is based on the concept of graph convolutional recurrent networks by Seo *et al.* (2018) [5]. In their framework, a recurrent neural network (like an LSTM or GRU) is combined with graph convolution operations to handle structured sequence data. This approach has proven effective in domains such as traffic flow forecasting, where nodes (sensors) form a graph and readings evolve over time. I employ a similar idea using the GConvGRU cell (a gated recurrent unit with graph convolutions). Temporal extensions of PyTorch Geometric, as described by Rozemberczki *et al.* (2021) [6], provide convenient building blocks for such models. My work applies these temporal GNN techniques in a novel context of option exercise decisions.

3 Methodology

My framework consists of: (1) Data collection and feature engineering from real market data from Kaggle (specifically, the JacksonCrow Stock Market dataset), (2) Synthetic trader behavior simulation and labeling, (3) Graph construction for each time snapshot, and (4) GNN model architecture and training procedure.

3.1 Data and Feature Engineering

I selected 10 stocks from different sectors of the S&P 500 to ensure diversity: AAPL (Tech), JNJ (Health), JPM (Financial), AMZN (Consumer Discr.), KO (Consumer Staples), XOM (Energy), BA (Industrial), NEE (Utilities), GOOG (Comm. Services), and LIN (Materials). For each, I obtained daily historical prices (Open-High-Low-Close-Volume) for the period 2022-2023 using Yahoo Finance. I also fetched the CBOE VIX index as a measure of overall market volatility when I could, but Yahoo Finance keeps rate limiting with any decent size of data I try to fetch.

From the price data, I engineered technical indicators commonly used by traders:

- **Relative Strength Index (RSI):** a momentum oscillator indicating overbought/oversold conditions (I use 14-day RSI).
- **Moving Average Convergence Divergence (MACD):** a trend-following indicator (difference of 12-day and 26-day EMAs) and its 9-day signal average.
- **Bollinger Bands:** 20-day moving average plus/minus 2 standard deviations, capturing recent volatility.
- **Price Returns and Volume:** daily percentage returns and trading volume.
- **Volatility Index (VIX):** I align the daily VIX value with each stock's data as a feature, to indicate market turmoil.

These features form the attributes of each *asset (stock) node* in my graph at a given time.

For the option contracts, I used Yahoo Finance option chain data to get representative strike prices and expiration dates. However, historical option prices were not directly available, so instead of using option prices, I derive labels from underlying prices (explained below) and use option characteristics (e.g., strike, time-to-expiry) as features. Each trade involves one option, characterized by:

- **Option type:** call or put
- **Strike price relative to underlying (moneyness):** e.g., moneyness = S/K for calls, where S is underlying price at trade time, K is strike.
- **Time to expiration:** in days, from exercise (or purchase) date to contract expiration.
- **Implied volatility:** I use either the option's quoted IV or estimate volatility from historical stock data (e.g., recent realized vol).
- **Option greeks (optional):** such as delta or gamma, which I can approximate via Black-Scholes if needed.

These form the features of an *option node*.

The *trader node* can have features describing the trader's profile or current strategy. In my simulation, I tag whether the trader is acting randomly or using

momentum in that trade, and I can include a bias term or their past success rate as features.

3.2 Synthetic Trader Simulation and Regret Labeling

Because public datasets of individual option exercise decisions (especially suboptimal ones) are not readily available, I generate synthetic examples. Each month in 2023 is treated as a decision opportunity where my simulated trader might buy an option and decide when to exercise.

I implemented two trading strategies: **(i) Random:** The trader picks a random day before expiration to exercise. **(ii) Momentum-based:** The trader monitors the underlying's momentum. If a short-term downtrend is detected (e.g., the 5-day moving average crossing below the 20-day, or price breaking below the lower Bollinger Band), the trader exercises immediately to avoid losing profits. If no negative signal occurs, the trader holds until expiration (or the last day).

For each trade, I determine the regret after the fact by looking at the underlying stock's subsequent prices. Specifically, if the trader exercised on day t_{exercise} at price S_t, and the maximum price up to expiration (day T) was $\max_{t<u\leq T} S_u$, then:

$$\text{Regret} = \max\{0, \max_{u>t} S_u - S_t\}.$$

This represents the extra profit per share the trader could have gained by waiting for the optimal peak (assuming a call option). I also define a normalized Regret-Based Opportunity Score (RBOS) as:

$$\text{RBOS} = \frac{\max_{u>t} S_u - S_t}{S_t},$$

which is essentially the percent increase the stock achieved after the sale. A nonzero regret indicates a suboptimal exercise.

I label each trade in two ways: - Binary label: $\textit{suboptimal} = 1$ if RBOS $> \tau$, else 0, where τ is a threshold (I used $\tau = 0$ or a small positive value like 1% to filter out negligible misses). - Regression target: the RBOS value (or absolute \$ regret) itself, to quantify degree of suboptimality.

My simulation produced a dataset of $\sim$100 trade instances (roughly 1 per stock per month). While not large, it is enough to train a proof-of-concept model. Each instance includes the trade execution time, strategy type, and outcome (regret).

3.3 Graph Construction

I model the state of the system at each month's end (or at each exercise decision point) as a graph $G_t = (V, E_t)$. The node set V includes:

- One **Trader** node (single agent making all decisions in this simplified setup).

- Ten **Asset** nodes (one per stock, representing underlying equities).
- An **Option** node for the active option contract in that month.

Edges E_t at time t include:

- Trader–Option edge (undirected, or two directed edges): connects the trader to the option they hold.
- Option–Asset edge: connects the option to its underlying stock's node.

I do not directly connect the trader to the asset node, to force information flow through the option node which represents the transaction relationship. (One could add Trader–Asset edges to indicate the trader's general interest in certain stocks, but I omitted that for clarity.)

Over time, as t progresses month to month, the graph changes: the Option node from month t might be considered to be removed or replaced by a new node in month $t+1$ (for a new option). For simplicity, I reused a single Option node identity, updating its features and changing its connections to the new asset in focus each month. This yields a temporally evolving graph where the set of nodes is fixed, but their features and the edge set E_t change with t. An alternative would be to instantiate a new option node for each trade and keep old ones in the graph (with perhaps no edges once their trade is done). I chose the former approach (node reuse) to keep the state space small, though it means the Option node's hidden state carries over and must be interpreted carefully (I reset part of its features but allow the model to carry some memory, effectively giving the GNN a way to remember the trader's past option outcomes).

Each Asset node's features are updated to the values of technical indicators at the current time step t. The Trader node's features can also be updated; e.g., I include a feature for the strategy used in that period (random or momentum) which can inform the model of the decision context. All node features are numeric and scaled appropriately (standardizing price-based features, etc., to assist training).

This forms a heterogeneous graph (different node types) at each snapshot. I did not use type-specific convolution due to the modest graph size; instead, the GNN learns a unified representation, relying on feature vectors to carry type information (I include one-hot encodings or specific feature slots for type if needed). This approach is akin to treating the graph as homogeneous but with rich node feature descriptors that indicate their type.

3.4 GNN Model Architecture

To model the temporal graph sequence $\{G_1, G_2, \ldots, G_T\}$, I employ a recurrent graph neural network. My model uses a Graph Convolutional GRU (GConvGRU) layer [5], which performs graph convolutions at each time step and retains a hidden state for each node across steps (similar to how a standard GRU retains a hidden vector for each sequence element). The graph convolution used is the Chebyshev spectral convolution (of order K) as implemented in

torch_geometric_temporal [6]. In my experiments, I set $K = 2$ (each convolution covers up to 2-hop neighborhoods).

Model details: I denote the input feature matrix at time t as $X_t \in \mathbb{R}^{N \times d}$ (where $N = |V| = 12$ nodes in my case, and d is feature dim). The GConvGRU takes X_t, the graph's edge index (connectivity), and the previous hidden state $H_{t-1} \in \mathbb{R}^{N \times h}$ (with h the hidden dimension) to produce an updated H_t. Internally, it computes a graph convolution $Z_t = \text{GraphConv}(X_t, A_t)$ (where A_t is adjacency) as part of its gating mechanism. Conceptually, each node's hidden state is influenced by its neighbors' previous hidden states and current features.

On top of the recurrent layer, I add a simple feedforward classifier/regressor: After obtaining H_t, I focus on the hidden state of the Option node (since that corresponds to the decision/trade at time t). Let $h_t^{(opt)}$ be the hidden vector for the option node. I feed $h_t^{(opt)}$ into a fully-connected layer to predict the outcome:

$$\hat{y}_t = w^T h_t^{(opt)} + b,$$

where $\hat{y}_t$ is a scalar. In classification mode, $\hat{y}_t$ is a logit (I apply a sigmoid to get probability of suboptimal exercise); in regression mode $\hat{y}_t$ is the predicted regret value.

The model is trained through time: at each time step t, I compute a loss comparing $\hat{y}_t$ with the true label y_t. I sum (or average) losses over the sequence and backpropagate through the entire unrolled sequence (BPTT). In my training, I used binary cross-entropy loss for classification and mean squared error for regression. I also experimented with a combined loss (treating it as a multi-task learning problem where the model outputs both a classification and a regression), but found focusing on classification was more straightforward.

I implemented the model using PyTorch Geometric Temporal library [6]. The GConvGRU cell and data iterator for temporal graphs simplified the process. The hidden dimension h was set to 32, which was sufficient for the small graph size.

3.5 Training Procedure

I trained the model for 100 epochs using Adam optimizer (learning rate 0.01). During each epoch, I iterated through the time snapshots in chronological order. The hidden state was initialized as zero at the start of each sequence (January 2023) and carried through to December. I logged the training loss at each epoch. Because of the limited data, I used the training sequence itself to evaluate performance (reporting training fit); ideally, one would generate additional synthetic sequences for testing or use part of the sequence for validation (e.g., train on first 8 months, test on last 4). Given the sequential nature, I opted not to shuffle the time order. In a multi-sequence scenario, one could shuffle the sequences but not the internal temporal order.

To prevent overfitting on this small dataset, I kept the model simple. I also experimented with dropping some features (to test robustness) and found the model still learned the main patterns, suggesting it was not just memorizing

specific feature values but capturing structural cues (like "option was exercised while RSI was high and an uptrend continued, hence regret").

4 Experiments and Results

I evaluated the model's performance on identifying suboptimal exercises and estimating regret. Despite the small scale, the results are illustrative of the model's capabilities.

Classification Performance: Varying behavior was observed from experimentation; only a few results will be discussed, due to space limitations. Out of a run for ∼100 simulated trades, about 60% were labeled suboptimal (the others had little or no regret, often because the stock did not rise after exercise). The GNN model achieved an overall accuracy of about 80% in classifying in some of these cases. Table 1 shows the confusion matrix. The precision for identifying suboptimal exercises was high (the model rarely false-flagged an optimal exercise as suboptimal), possibly because genuine suboptimal cases had strong signals (e.g., stock price surged after exercise) that the model picked up on. Recall was moderate; a few suboptimal cases were missed, often those with very borderline regret or noisy indicator behavior.

Table 1: Confusion Matrix for Suboptimal Exercise Classification (on simulation data).

	Predicted Optimal	**Predicted Suboptimal**
Actual Optimal	16	4
Actual Suboptimal	7	19

The model's errors made intuitive sense: for instance, some false negatives (missed suboptimal) occurred in cases where the stock's post-exercise rally was short-lived or marginal, and the indicators at exercise time were not strongly bullish, making it hard even for an expert to predict the rally. False positives (flagging optimal exercises as suboptimal) sometimes occurred for trades where the stock was volatile but ultimately did not exceed the sell price — the model anticipated a rise that didn't materialize. Such cases might be mitigated with more training data or by incorporating information about upcoming events (earnings, etc., which in reality influence decisions).

Regression Performance: When training the model to predict the RBOS value, I found it could approximate the magnitude of regret with an R^2 of about 0.5 on the training data — not very high, but indicating some correlation. Extremely high regret cases (where the stock skyrocketed after sale) were usually predicted as high as well, but smaller gradations were harder to get exact. This is expected given the limited data. However, even a coarse regression prediction can be useful to rank decisions by how much regret they might incur.

Ablation and Insights: I probed which features and connections were most important by training variants of the model: - Removing the trader node or

its features (treating it as a constant) had little effect on performance, which suggests the strategy type (random vs momentum) was not a dominant factor for the model. The asset's state and outcome were more crucial. This makes sense: suboptimality is more about what the market did after the decision than the decision rule itself. - If I disconnect the graph (no edges, meaning the model cannot relate the trader and asset via the option), the performance drops significantly (to near 65% accuracy). This confirms that the relational inductive bias — treating the trade as a link between trader and asset — is beneficial. The GNN essentially learns a pattern like: "Trader exercised on asset X; asset X's features and subsequent movement indicate if that was too early." Without the graph structure, the model would have to rely on a flat sequence of features, losing the clear association of which asset was involved in each trade.

Qualitative Analysis: I also visualized the learned hidden state trajectories. The option node's hidden state tends to cluster in two regions corresponding to the two classes, suggesting the GConvGRU learned a representation that separates regretful vs non-regretful outcomes. In one case, for a momentum strategy trade on Amazon (AMZN) in July, the model correctly predicted it as suboptimal. The indicators at exercise showed a mild uptrend, but nothing obvious; however, the hidden state incorporated the broader context (AMZN node had very low RSI a month prior followed by rising momentum) which hinted that the stock had more room to rise — indeed it jumped after the trader exited. The GNN, by capturing temporal dependencies, potentially recognized this pattern.

5 Discussion

Myresults, while based on a simulated dataset, demonstrate the feasibility of using temporal GNNs to detect suboptimal option exercises. The model effectively integrates information about the trader's action, the specific option's attributes, and the underlying asset's state. There are several points worth discussing:

Generality and Extension to Multiple Traders: I simplified to one trader to concentrate on the core decision modeling. In reality, different traders have different propensities for premature exercise. Myframework could be extended to multiple trader nodes, each with their own series of decisions. The heterogeneous graph would then have edges mapping each trader to the options they trade. This would increase graph size and complexity, but the GNN approach should scale given appropriate modifications (perhaps using mini-batches of subgraphs). It would be interesting to see if a model could learn trader-specific factors (some traders consistently exercise too early due to personal bias, etc.).

Real Data Application: Applying this model to real trading data is challenging but conceivable. Brokerage data could provide timestamps of option exercises along with the relevant market data. One hurdle is obtaining the ground truth of "what if the trader waited?" — I approximated it with hindsight maximum price, which is valid for evaluating regret. In a live setting, one could use this retrospectively to label past decisions. Another hurdle is that

real decisions are influenced by factors like risk management, capital needs, or portfolio considerations which my simulation did not capture. Nonetheless, the model might still flag decisions that appear suboptimal from a market perspective.

Regret as a Continuous Signal: Beyond classification, predicting the magnitude of regret can be valuable. A trader education tool could say, "Selling this call now is likely suboptimal; you might be leaving approximately 5% of additional profit on the table if historical patterns repeat." My regression results are a first step. With more data, this could be refined. Moreover, alternative definitions of regret (time-decayed regret, or regret adjusted for risk) could be explored. For instance, a large regret right after exercise might be less "blameworthy" if only a few days were left to expiration versus if a month remained.

Limitation – Option Pricing Factors: I did not explicitly model option premiums or the Greeks in depth. In reality, exercising early also means giving up remaining time value, which is rational only if that time value is less than the immediate intrinsic value gained. My model implicitly learns some of this (since deep in-the-money options nearing expiration are more likely to be exercised optimally, e.g., an American put deep ITM might be correctly exercised early to earn interest on proceeds). Incorporating option pricing formulas or Greeks into the feature set could improve the model's understanding of what an optimal exercise boundary looks like, thereby highlighting deviations.

Future Work: A promising extension is to incorporate more market context, such as corporate events (dividends, earnings releases) which strongly affect exercise decisions. Dividends can make early exercise of calls optimal (to capture the dividend), and my model could be expanded to learn those conditions too. Another direction is using the model for what-if analysis: simulate how a different decision (holding longer) might pan out. Since my GNN has learned a representation of the system's dynamics, one could potentially intervene on the graph (e.g., keep the option node active longer) and see how the model's predictions change, essentially using the model as a proxy for scenario analysis.

6 Conclusion

I presented a novel approach to identify suboptimal exercise decisions of American options by modeling the trading system as a temporal heterogeneous graph and applying a graph neural network. In a world where hindsight is 20/20, my model tries to bring some of that hindsight into foresight — predicting regret before it happens. The integration of trader behavior with market state in a GNN framework enables capturing complex patterns that rule-based approaches might miss. While my results are based on simulations, they lay the groundwork for more sophisticated applications. As data availability improves, such models could become part of smart trading platforms, alerting traders when they're about to "take money off the table" too soon, or helping risk managers understand the behavioral risks in options portfolios.

Overall, this work demonstrates the synergy of financial expertise (knowing

what constitutes a bad decision in options trading) with advanced machine learning (temporal GNNs) to tackle a problem at the intersection of behavioral finance and computational finance. I hope it spurs further exploration into graph-based representations of trading behavior and the development of decision support tools that can reduce costly mistakes in the markets.

References

[1] Hu, M., Tan, Z., Liu, B., & Yin, G. (2024). *Futures Quantitative Investment with Heterogeneous Continual Graph Neural Network*. arXiv:2303.16532.

[2] Djagba, P., & Ndizihiwe, C. (2024). *Pricing American Options using Machine Learning Algorithms*. arXiv:2409.03204.

[3] Aretz, K., Garrett, I., & Gazi, A. (2020). *Taking Money Off the Table: Suboptimal Early Exercises, Risky Arbitrage, and American Put Returns*. SSRN Working Paper 3677041.

[4] Longstaff, F. A., & Schwartz, E. S. (2001). *Valuing American Options by Simulation: A Simple Least-Squares Approach*. *Journal of Finance*, 56(5), 1647–1672.

[5] Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. (2018). *Structured Sequence Modeling with Graph Convolutional Recurrent Networks*. arXiv:1612.07659.

[6] Rozemberczki, B., *et al.* (2021). *PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models*. In *Proc. of CIKM '21*.