Key Mathematical Foundations for RL and LLMs

Michal Andrzejewski

Core Concepts in RL

Agent: A decision-making entity that interacts with an environment to maximize cumulative rewards. Formally, the agent selects actions a_t based on a policy $\pi(a|s)$.

Environment: The external system with which the agent interacts, typically modeled as a Markov Decision Process (MDP). The environment provides states s_t and rewards r_t in response to actions a_t.

State (s): A representation of the environment at a given time step. States can be fully observable, partially observable, or encoded as embeddings.

Action (a): A decision or move made by the agent. Actions can be discrete (e.g., move left/right) or continuous (e.g., adjust a control parameter).

Reward (r): A scalar signal providing feedback on the agent's performance. The agent's goal is to maximize the expected cumulative reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{1}$$

where γ is the discount factor.

Trajectory (τ): A sequence of states, actions, and rewards: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$.

Policy ($\pi(a|s)$): A mapping from states to action probabilities. Policies can be deterministic ($a = \pi(s)$) or stochastic ($a \sim \pi(a|s)$).

Value Function ($V(s)$): The expected cumulative reward starting from state s:

$$V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right]. \tag{2}$$

Action-Value Function ($Q(s,a)$): The expected cumulative reward starting from state s and taking action a:

$$Q(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right]. \tag{3}$$

Advantage Function ($A(s,a)$): Measures how much better an action is compared to the expected value of a state:

$$A(s,a) = Q(s,a) - V(s). \tag{4}$$

Markov Decision Process (MDP)

An MDP is defined as a tuple (S, A, P, R, γ):

- S: Set of possible states.
- A: Set of possible actions.
- $P(s'|s,a)$: Transition probability function.
- $R(s,a)$: Reward function.
- γ: Discount factor ($0 \leq \gamma < 1$).

The Markov property ensures that the future state depends only on the current state and action:

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots). \tag{5}$$

Bellman Equations

State-Value Function:

$$V(s) = \mathbb{E}[r + \gamma V(s')|s]. \tag{6}$$

Action-Value Function:

$$Q(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q(s', a')|s, a]. \tag{7}$$

These equations form the basis of dynamic programming and temporal difference learning.

Deep Learning Foundations

Neural Networks

A neural network is a function approximator composed of layers of neurons. For a feedforward network with L layers:

$$\mathbf{h}_l = \sigma(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l), \tag{8}$$

where $\mathbf{h}_l$ is the output of layer l, $\mathbf{W}_l$ and $\mathbf{b}_l$ are weights and biases, and σ is a nonlinear activation function (e.g., ReLU, tanh).

Convolutional Neural Networks (CNNs)

CNNs are specialized for grid-like data (e.g., images). A convolution operation applies a filter $\mathbf{K}$ to an input $\mathbf{X}$:

$$(\mathbf{X} * \mathbf{K})_{i,j} = \sum_m \sum_n \mathbf{X}_{i-m,j-n} \mathbf{K}_{m,n}. \tag{9}$$

Key components:

- **Convolutional Layers**: Extract spatial features.
- **Pooling Layers**: Reduce spatial dimensions (e.g., max pooling).
- **Fully Connected Layers**: Combine features for final output.

Graph Neural Networks (GNNs)

GNNs operate on graph-structured data. For a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, a GNN updates node embeddings $\mathbf{h}_v$ as:

$$\mathbf{h}_v^{(l+1)} = \sigma \left(\mathbf{W}_l \mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}(v)} \mathbf{W}'_l \mathbf{h}_u^{(l)} \right), \tag{10}$$

where $\mathcal{N}(v)$ is the set of neighbors of node v.

Transformers

Transformers use self-attention to process sequences. For an input sequence $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, the self-attention mechanism computes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{11}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are learned linear transformations of $\mathbf{X}$.

Advanced RL Topics

Model-Free vs. Model-Based RL

- **Model-Free RL**: Learns a policy or value function without explicitly modeling the environment (e.g., Q-learning, PPO).
- **Model-Based RL**: Uses a learned model of the environment dynamics to plan and optimize decisions (e.g., Dyna-Q, MuZero).

Hierarchical RL

Options Framework: Breaks tasks into subtasks with temporal abstraction.

$$\pi(a|s) = \sum_o P(o|s)\pi(a|o), \tag{12}$$

where o is a subtask.

Deep Learning for RL

- **Transformers in RL**: Used in sequence modeling (GATO, Decision Transformer).
- **Graph Neural Networks (GNNs)**: Improve coordination in multi-agent RL by learning spatial-temporal relationships.
- **Neural PDEs for Continuous Spaces**: Models RL problems in continuous state/action spaces.

Multi-Agent RL (MARL)

Extends RL to scenarios with multiple interacting agents. Key concepts:

- **Centralized Training, Decentralized Execution (CTDE)**: Agents share information during training but act independently during execution.
- **Cooperative, Competitive, or Mixed-Sum Games**: Agents may have aligned, conflicting, or mixed objectives.
- **Credit Assignment**: Difficulty in attributing success or failure to individual agents.

LLM-Guided RL

LLMs assist in decision-making within RL systems:

- **Curriculum Learning**: LLMs design learning sequences for agents.
- **Prompt-Based Action Selection**: LLMs suggest actions given state descriptions.
- **Language-Augmented Policies**: Policies integrate linguistic instructions.

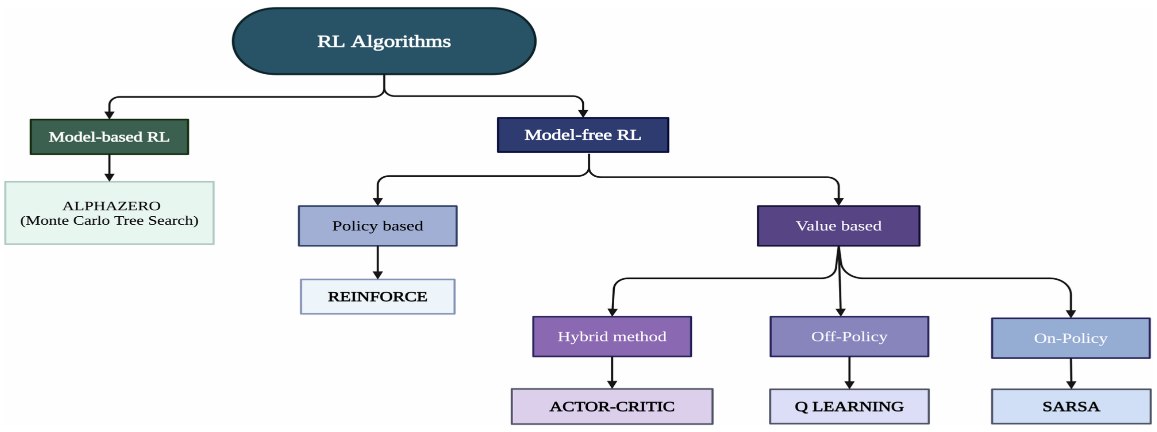

Figure 1: Key RL Categories. Source: Nature

References: Sutton & Barto (2018), Silver et al. (2016), OpenAI PPO (2017), Brown et al. (2020)

Reinforcement Learning Cheatsheet

Michal Andrzejewski

Core Concepts

Agent: Learns and interacts with the environment.

Environment: The external system with which the agent interacts.

State (s): The current representation of the environment.

Action (a): A decision made by the agent.

Reward (r): Feedback signal indicating success.

Policy (π): The strategy used by the agent to choose actions.

Value Function ($V(s)$): Expected cumulative reward from state s.

Q-Function ($Q(s, a)$): Expected cumulative reward from state s after taking action a.

Markov Decision Process (MDP): A tuple (S, A, P, R, γ):

- S: Set of states
- A: Set of actions
- $P(s'|s, a)$: Transition probability
- $R(s, a)$: Reward function
- γ: Discount factor ($0 \leq \gamma < 1$)

Bellman Equations

State-Value Function:

$$V(s) = \mathbb{E}[r + \gamma V(s')|s] \tag{13}$$

Action-Value Function:

$$Q(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q(s', a')|s, a] \tag{14}$$

Key RL Algorithms

Q-Learning (Off-Policy, Model-Free)

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{15}$$

Deep Q-Networks (DQN)

- Uses a neural network to approximate Q-values.
- Experience Replay: Stores past transitions and samples randomly.
- Target Networks: A separate network stabilizes training.

Policy Gradient Methods

$$\nabla J(\theta) = \mathbb{E}[\nabla \log \pi(a|s; \theta) Q(s, a)] \tag{16}$$

Actor-Critic Methods

- Actor: Learns policy $\pi(a|s)$.
- Critic: Learns value function $V(s)$.

Proximal Policy Optimization (PPO)

$$L(\theta) = \min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t\right) \tag{17}$$

Soft Actor-Critic (SAC)

$$J(\pi) = \mathbb{E}\left[\sum_t \gamma^t (r_t + \alpha H(\pi(\cdot|s_t)))\right] \tag{18}$$

Exploration vs. Exploitation

ε-Greedy: With probability ε, take a random action; otherwise, take the best action.

Boltzmann Exploration: Select actions using softmax probabilities:

$$P(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}} \tag{19}$$

UCB (Upper Confidence Bound): Balances exploration-exploitation tradeoff.

$$A_t = \arg\max_a \left(Q(s, a) + c\sqrt{\frac{\ln t}{N(s, a)}}\right) \tag{20}$$

Discounting and Horizon

Discount Factor (γ):

- Closer to 1 $\Rightarrow$ Long-term rewards matter more.
- Closer to 0 $\Rightarrow$ Focuses on immediate rewards.

Finite vs. Infinite Horizon:

- **Finite Horizon**: Optimization over a fixed number of steps.
- **Infinite Horizon**: Optimization over an unbounded future.

Practical Considerations

- **Reward Shaping**: Helps speed up learning but can introduce bias.
- **Batch vs. Online Learning**: Batch updates stabilize learning.
- **Hyperparameter Tuning**: Learning rate, batch size, discount factor impact performance.

References: Sutton & Barto (2018), Silver et al. (2016), OpenAI PPO (2017)

Multi-Agent Reinforcement Learning (MARL) Cheatsheet

Michal Andrzejewski

Core Concepts in MARL

Multi-Agent System (MAS): Multiple agents interact in a shared environment. Each agent has its own policy, observations, and rewards, but their actions collectively influence the environment.

State (s): The global state of the environment, which may be fully observable, partially observable, or only locally observable by individual agents.

Action (a): A joint action vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, where a_i is the action taken by agent i. The joint action space grows exponentially with the number of agents.

Reward (r): Each agent may receive an individual reward r_i or a shared reward r. Reward structures can be cooperative (aligned), competitive (conflicting), or mixed.

Policy (π_i): The strategy used by agent i to select actions. Policies can be deterministic or stochastic.

Value Function ($V_i(s)$): The expected cumulative reward for agent i starting from state s and following policy π_i.

Q-Function ($Q_i(s, \mathbf{a})$): The expected cumulative reward for agent i starting from state s, taking joint action $\mathbf{a}$, and following policy π_i.

Markov Game: A generalization of MDPs to multi-agent settings. Defined as a tuple $(S, A_1, \ldots, A_n, P, R_1, \ldots, R_n, \gamma)$:

- S: Set of states.
- A_i: Set of actions for agent i.
- $P(s'|s, \mathbf{a})$: Transition probability to state s' given joint action $\mathbf{a}$.
- $R_i(s, \mathbf{a})$: Reward function for agent i.
- γ: Discount factor ($0 \leq \gamma < 1$).

Key Challenges in MARL

Non-Stationarity:

- The environment dynamics change as other agents learn, violating the Markov property for individual agents.
- Requires adaptive learning algorithms to handle evolving strategies of other agents.

Credit Assignment:

- In cooperative settings, it is challenging to attribute global success or failure to individual agents.
- Solutions include difference rewards, counterfactual reasoning, and reward shaping.

Coordination vs. Competition:

- **Coordination**: Agents must align their actions to achieve a common goal (e.g., multi-robot teams).
- **Competition**: Agents have conflicting objectives (e.g., adversarial games like chess or poker).
- **Mixed Settings**: Agents may have both cooperative and competitive goals (e.g., trading markets).

Communication:

- **Explicit Communication**: Agents share information via predefined protocols or learned communication channels.
- **Implicit Communication**: Agents infer information from others' actions or environmental changes.
- **Learning to Communicate**: Agents develop communication protocols during training (e.g., using differentiable communication channels).

MARL Algorithms

Independent Q-Learning (IQL):

- Each agent learns its own Q-function independently, treating other agents as part of the environment.
- Pros: Simple and scalable.
- Cons: Suffers from non-stationarity and lacks coordination.

Multi-Agent Deep Q-Networks (MADQN):

- Extends DQN to multi-agent settings by using separate networks for each agent.
- Centralized Training with Decentralized Execution (CTDE): Agents share experiences during training but act independently during execution.
- Pros: Handles high-dimensional state spaces and improves coordination.
- Cons: Computationally expensive and requires careful tuning.

Policy Gradient Methods in MARL:

$$\nabla J_i(\theta_i) = \mathbb{E}[\nabla \log \pi_i(a_i|s; \theta_i) Q_i(s, \mathbf{a})] \tag{21}$$

- Each agent optimizes its policy using gradient ascent.
- Can incorporate centralized critics or shared value functions.

Actor-Critic Methods in MARL:

- **Actor**: Learns the policy $\pi_i(a_i|s; \theta_i)$.
- **Critic**: Learns the value function $V_i(s)$ or Q-function $Q_i(s, \mathbf{a})$.
- **Centralized Critics**: Use global information during training to improve coordination.

- **Decentralized Critics**: Each agent learns its own critic using local information.

Multi-Agent PPO (MAPPO):

$$L_i(\theta_i) = \min\left(r_{i,t}(\theta_i)A_{i,t}, \text{clip}(r_{i,t}(\theta_i), 1-\epsilon, 1+\epsilon)A_{i,t}\right) \quad (22)$$

- Extends PPO to multi-agent settings with shared or separate networks.
- Pros: Stable and sample-efficient.
- Cons: Requires careful hyperparameter tuning.

Mean Field MARL:

- Approximates the effect of other agents as a mean field, reducing complexity.
- Suitable for large-scale systems with many agents.
- Pros: Scalable and computationally efficient.
- Cons: Assumes homogeneity among agents.

Communication in MARL

Explicit Communication:

- Agents exchange messages to share information.
- Protocols can be predefined or learned.
- Examples: Differentiable inter-agent learning (DIAL), CommNet.

Implicit Communication:

- Agents infer information from others' actions or environmental changes.
- Examples: Stigmergy in swarm robotics.

Learning to Communicate:

- Agents develop communication protocols during training.
- Examples: Reinforcement learning with emergent communication (RL-EC).

Cooperative vs. Competitive MARL

Cooperative MARL:

- Agents share a common goal and must coordinate their actions.
- Examples: Multi-robot coordination, team sports, collaborative tasks.

Competitive MARL:

- Agents have conflicting objectives and compete against each other.
- Examples: Adversarial games (e.g., chess, poker), auctions.

Mixed MARL:

- Agents have both cooperative and competitive objectives.
- Examples: Trading markets, negotiation scenarios, mixed-motive games.

Practical Considerations in MARL

- **Scaling**: MARL becomes computationally expensive as the number of agents increases. Techniques like mean field approximation and decentralized execution help mitigate this.
- **Exploration**: Agents must explore both their own policies and the behaviors of other agents. Techniques like intrinsic motivation and curiosity-driven exploration can help.
- **Stability**: Training multiple agents simultaneously can lead to instability. Techniques like experience replay, target networks, and regularization improve stability.

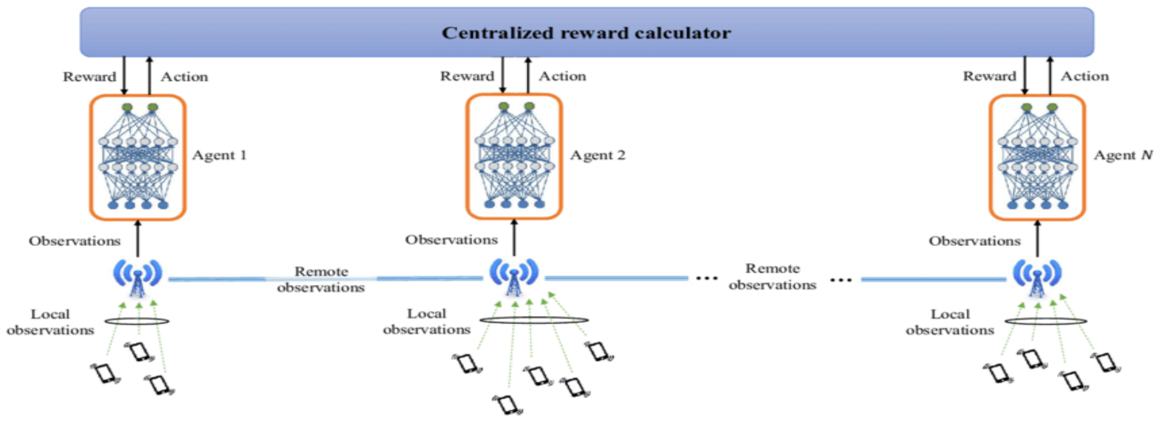

Figure 2: Multi-agent deep reinforcement learning diagram, where the agents receive local observations and select actions accordingly. Source: Multi-agent deep reinforcement learning diagram.

Game Theory & Decision-Making Under Constraints

Michal Andrzejewski

Core Concepts in Game Theory

Game: A formal framework for modeling interactions between rational agents, where each agent selects strategies to maximize its utility.

Players: The decision-making entities in a game, corresponding to agents in MARL.

Strategies: The set of possible actions or policies an agent can adopt. Strategies can be pure (deterministic) or mixed (probabilistic distributions over actions).

Payoff Function: Maps each strategy profile to numerical rewards, representing agent objectives.

Utility Maximization: Each agent seeks to maximize its expected utility given its knowledge of the game and other agents' behavior.

Types of Games

Normal-Form Game: Represented as (N, S, U) where:

- N is the set of players.
- $S = S_1 \times S_2 \times ... \times S_n$ is the set of strategy profiles.
- $U = (U_1, U_2, ..., U_n)$ where $U_i : S \to \mathbb{R}$ assigns payoffs.

Extensive-Form Game: A sequential decision-making game represented as a tree where agents act in turns.

Cooperative vs. Non-Cooperative: Cooperative games allow binding agreements, while non-cooperative games focus on self-interested decision-making.

Zero-Sum vs. General-Sum:

- **Zero-Sum**: One player's gain is another's loss ($\sum U_i = 0$).
- **General-Sum**: Players' payoffs are not necessarily opposing.

Equilibrium Concepts

Nash Equilibrium (NE): A strategy profile $(s_1^*, s_2^*, ..., s_n^*)$ where no player can unilaterally improve their payoff:

$$U_i(s_i^*, s_{-i}^*) \geq U_i(s_i, s_{-i}^*) \quad \forall s_i \in S_i, \forall i. \tag{23}$$

Correlated Equilibrium (CE): A probability distribution over strategy profiles where agents follow recommendations to maximize expected utility.

Minimax Strategy: Used in adversarial settings (e.g., AlphaZero), where an agent minimizes the opponent's maximum possible gain:

$$\pi^* = \arg\min_{\pi} \max_{\pi'} U(\pi, \pi'). \tag{24}$$

Game Theory in MARL

Multi-Agent Learning: Agents adapt policies in dynamic, interactive environments.

Decentralized vs. Centralized Learning:

- **Centralized Training, Decentralized Execution (CTDE)**: Policies are trained with shared information but executed independently.
- **Independent Learning**: Each agent learns without direct knowledge of others.

Markov Games: The MARL extension of MDPs where state transitions depend on joint actions:

$$P(s'|s, a_1, ..., a_n). \tag{25}$$

Credit Assignment Problem: Determining individual agent contributions to team success in cooperative MARL.

Equilibrium Computation in MARL: Algorithms such as Nash Q-learning and Fictitious Play are used to find equilibria in multi-agent environments.

Exploration vs. Exploitation in MARL: Agents must balance learning new strategies (exploration) with leveraging known good strategies (exploitation).

Applications in Constraint-Aware AI

- **Resource Allocation**: Optimizing shared resources in multi-agent decision systems.
- **Autonomous Trading**: Game-theoretic RL strategies for market making and portfolio optimization.
- **Robust AI Systems**: Ensuring AI decisions remain optimal under constraints and adversarial conditions.

References: Osborne & Rubinstein (1994), Shoham et al. (2007), Sutton & Barto (2018), Silver et al. (2016)

AI-Assisted Human-Guided RL Cheatsheet

Michal Andrzejewski

Core Concepts

Human-in-the-Loop RL (HITL-RL): A framework where human feedback is integrated into the RL loop to guide agent behavior. Humans provide demonstrations, preferences, or corrections, which are used to shape the reward function or policy. Formally, the agent interacts with both the environment and a human teacher, who provides feedback $\mathcal{F}$ to improve the agent's policy π.

Reinforcement Learning with Human Feedback (RLHF): A specific approach where human feedback is used to define or refine the reward function, often through preference learning or ranking-based methods. The goal is to align the agent's behavior with human values. Formally, given human feedback $\mathcal{F}$, the agent learns a reward function $R(s, a)$ that maximizes the likelihood of $\mathcal{F}$.

LLM-Enhanced Decision-Making: Leveraging large language models (LLMs) to assist in decision-making by generating synthetic feedback, interpreting human preferences, or providing explainable insights into agent behavior. LLMs can act as proxies for human feedback, reducing the need for extensive human involvement.

Reward Modeling: Learning a reward function $R(s, a)$ from human feedback. This can be done via:

- **Inverse Reinforcement Learning (IRL)**: Inferring $R(s, a)$ from demonstrations.
- **Preference Learning**: Inferring $R(s, a)$ from pairwise comparisons of trajectories.

Preference Learning: A technique where humans rank or compare trajectories, and the agent learns a reward function that aligns with these preferences. Given trajectories τ_1 and τ_2, the agent learns R such that $R(\tau_1) > R(\tau_2)$ if τ_1 is preferred. Formally:

$$P(\tau_1 \succ \tau_2 | R) = \frac{\exp(R(\tau_1))}{\exp(R(\tau_1)) + \exp(R(\tau_2))}. \tag{26}$$

Key Challenges

Human Feedback Quality:

- Human feedback is often noisy, inconsistent, or biased.
- Techniques like robust optimization and noise-tolerant reward modeling are used to mitigate this.
- Example: Use a Gaussian process to model uncertainty in human feedback.

Scalability:

- Collecting human feedback at scale is expensive and time-consuming.
- Solutions include:
  - **Active Learning**: The agent queries humans for feedback on the most informative states or actions.
  - **Synthetic Feedback**: Use LLMs or other models to generate feedback.

Alignment:

- Ensuring the agent's behavior aligns with human values and intentions.
- Techniques include:
  - **Cooperative Inverse Reinforcement Learning (CIRL)**: The human and agent cooperate to infer the reward function.
  - **Reward Shaping**: Modify the reward function to better align with human preferences.

Interpretability:

- Humans need to understand and trust the agent's decisions.
- Techniques include:
  - **SHAP (SHapley Additive exPlanations)**: Assigns importance values to features for a given prediction.
  - **LIME (Local Interpretable Model-agnostic Explanations)**: Approximates the model locally to provide interpretable explanations.
  - **LLM-Generated Explanations**: Use LLMs to provide natural language explanations of agent decisions.

Technical Methods

Inverse Reinforcement Learning (IRL):

- Given demonstrations $\mathcal{D} = \{(s_1, a_1), \ldots, (s_N, a_N)\}$, IRL infers a reward function $R(s, a)$ that explains the demonstrated behavior.
- Formally, find R such that the demonstrated policy π^* is optimal under R:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right]. \tag{27}$$

- Common algorithms include Maximum Entropy IRL and Generative Adversarial Imitation Learning (GAIL).

Preference-Based RL:

- Humans provide pairwise comparisons of trajectories, $\tau_1 \succ \tau_2$.
- The agent learns a reward function R that maximizes the likelihood of the observed preferences:

$$\max_{R} \mathbb{E}_{\tau_1, \tau_2}[\log P(\tau_1 \succ \tau_2 | R)], \tag{28}$$

  where $P(\tau_1 \succ \tau_2 | R) = \frac{\exp(R(\tau_1))}{\exp(R(\tau_1)) + \exp(R(\tau_2))}$.
- Example algorithms: T-REX, D-REX.

Active Learning:

- The agent selects states or actions where human feedback would be most informative.
- Formally, maximize the expected information gain:

$$s^* = \arg\max_{s\in\mathcal{S}} H(\pi(a|s)) - \mathbb{E}_{a\sim\pi(a|s)}[H(\pi(a|s, \text{feedback}))], \tag{29}$$

  where H is the entropy.
- Example: Uncertainty sampling, where the agent queries feedback for states with high predictive uncertainty.

RLHF with LLMs:

- LLMs are used to generate synthetic feedback or interpret human preferences.
- For example, given a trajectory τ, an LLM can generate a natural language explanation of why τ is preferred.
- LLMs can also be used to simulate human feedback, reducing the need for real human input.

Imitation Learning:

- The agent learns a policy $\pi(a|s)$ by imitating human demonstrations.
- Formally, minimize the divergence between the agent's policy π and the human's policy π_H:

$$\min_{\pi} D(\pi, \pi_H), \tag{30}$$

  where D is a divergence measure (e.g., KL divergence).
- Often combined with RL to fine-tune policies using a reward function learned from demonstrations.

Applications in Finance

Portfolio Optimization:

- Human experts provide feedback on trading strategies, which is used to refine the reward function.
- RLHF can align trading agents with risk preferences and ethical constraints.
- Example: Use preference learning to rank portfolios based on risk-return tradeoffs.

Fraud Detection:

- Human feedback is used to label fraudulent transactions, which is incorporated into the reward function.
- Active learning can prioritize high-risk transactions for human review.
- Example: Use IRL to infer a reward function that maximizes fraud detection accuracy.

Applications in Drug Discovery

Molecule Generation:

- Human chemists provide feedback on generated molecules, which is used to refine the reward function.
- RLHF can align molecule generation with desired properties like efficacy and safety.
- Example: Use preference learning to rank molecules based on binding affinity and toxicity.

Clinical Trial Design:

- Human experts provide feedback on trial designs, which is used to optimize the reward function.
- RLHF can balance exploration of new treatments with exploitation of known effective treatments.
- Example: Use active learning to query feedback on trial designs with the highest uncertainty.

Practical Considerations

- **Feedback Efficiency**: Use active learning or synthetic feedback to minimize human involvement.
- **Bias Mitigation**: Address biases in human feedback through diverse data collection and debiasing techniques.
- **Scalability**: Leverage LLMs or crowdsourcing platforms to scale feedback collection.
- **Interpretability**: Ensure agent decisions are interpretable using techniques like SHAP, LIME, or LLM-generated explanations.

References: Christiano et al. (2017), OpenAI (2022), Stiennon et al. (2020), Brown et al. (2020)

LLM-Guided RL and Strategy Optimization

Michal Andrzejewski

Core Concepts

LLM-Guided RL: A paradigm where large language models (LLMs) act as advisors or decision-support systems in reinforcement learning (RL). LLMs provide interpretable guidance, generate synthetic feedback, or assist in reward shaping to improve the agent's policy π.

Strategy Optimization: The process of optimizing decision-making strategies in dynamic environments, such as algorithmic trading or portfolio management. LLMs can assist by providing high-level strategic advice, interpreting market conditions, or generating synthetic data for training.

LLM as an Advisor: LLMs can act as advisors in RL by:

- Providing natural language explanations of agent decisions.
- Generating synthetic feedback or preferences for reward modeling.
- Suggesting high-level strategies or actions based on domain knowledge.

Interpretable Decision-Making: LLMs enhance interpretability by translating complex agent decisions into human-readable explanations. This is critical in high-stakes domains like finance, where transparency is essential.

Key Components

LLM-Guided Reward Shaping:

- LLMs generate synthetic rewards or preferences to guide the agent's learning process.
- Formally, given a state s, the LLM generates a reward $R_{\text{LLM}}(s,a)$ that is combined with the environment reward $R_{\text{env}}(s,a)$:

$$R(s,a) = \alpha R_{\text{env}}(s,a) + (1-\alpha)R_{\text{LLM}}(s,a), \tag{31}$$

where α controls the tradeoff between environment and LLM guidance.

LLM-Generated Synthetic Feedback:

- LLMs generate synthetic feedback (e.g., rankings or preferences) to augment human feedback.
- This reduces the need for extensive human involvement and scales reward modeling.

LLM-Enhanced Exploration:

- LLMs suggest exploratory actions or strategies based on domain knowledge.
- Example: In algorithmic trading, an LLM might suggest exploring a new trading strategy based on market trends.

LLM as a Policy Critic:

- LLMs critique the agent's policy π by identifying suboptimal decisions or suggesting improvements.
- This can be formalized as a multi-agent system where the LLM acts as a critic agent.

Technical Methods

LLM-Guided Preference Learning:

- LLMs generate pairwise preferences over trajectories, which are used to learn a reward function.
- Formally, given trajectories τ_1 and τ_2, the LLM generates a preference $\tau_1 \succ \tau_2$, and the agent learns R such that:

$$P(\tau_1 \succ \tau_2 | R) = \frac{\exp(R(\tau_1))}{\exp(R(\tau_1)) + \exp(R(\tau_2))}. \tag{32}$$

LLM-Assisted Reward Modeling:

- LLMs generate synthetic rewards $R_{\text{LLM}}(s,a)$ based on domain knowledge.
- These rewards are combined with environment rewards to guide the agent's learning:

$$R(s,a) = \alpha R_{\text{env}}(s,a) + (1-\alpha)R_{\text{LLM}}(s,a). \tag{33}$$

LLM-Enhanced Exploration Strategies:

- LLMs suggest exploratory actions or strategies to improve the agent's exploration.
- Example: In algorithmic trading, the LLM might suggest exploring a new asset class based on market trends.

LLM as a Policy Critic:

- The LLM critiques the agent's policy π by identifying suboptimal decisions or suggesting improvements.
- Formally, the LLM provides a critique $C(s,a)$ that is used to update the policy:

$$\pi'(a|s) = \pi(a|s) + \eta C(s,a), \tag{34}$$

where η is a learning rate.

Applications in Finance

Algorithmic Trading:

- LLMs provide high-level trading strategies or interpretable explanations of market conditions.
- Example: An LLM might suggest a momentum-based trading strategy during a bull market.

Portfolio Optimization:

- LLMs assist in optimizing portfolios by suggesting asset allocations based on market trends.

- Example: An LLM might recommend increasing exposure to tech stocks during a tech boom.

Risk Management:

- LLMs provide interpretable explanations of risk factors and suggest risk mitigation strategies.
- Example: An LLM might recommend hedging strategies during periods of high volatility.

Recent Research Developments

LLM-Guided Reward Shaping:

- The FinCon framework introduces an LLM-based multi-agent system for financial tasks, incorporating self-critiquing mechanisms to update investment beliefs and shape rewards based on conceptual verbal reinforcement.
- Recent work in FinRL-DeepSeek integrates risk assessments and trading recommendations from financial news, allowing for adaptive reward shaping in RL frameworks.

LLM-Generated Synthetic Feedback:

- The FinRL-DeepSeek framework extends RL with LLM-generated financial insights, providing synthetic feedback to improve decision-making efficiency in trading environments.

LLM-Enhanced Exploration:

- The LINVIT algorithm utilizes LLM guidance as a regularization factor in value-based RL, reducing data requirements and enhancing exploration efficiency.

LLM as a Policy Critic:

- The FinCon framework includes a risk-control component that refines trading strategies in real-time through self-critiquing mechanisms

Key Contributions:

- Proposed a novel LLM-guided reward shaping mechanism for algorithmic trading.
- Demonstrated the effectiveness of LLM-generated synthetic feedback in reducing human involvement.
- Showed that LLM-guided exploration strategies improve trading performance in dynamic markets.

Results:

- Achieved a 15% improvement in Sharpe ratio compared to baseline RL methods.
- Reduced human feedback requirements by 40% through LLM-generated synthetic feedback.
- Improved interpretability of trading decisions through LLM-generated explanations.

Practical Considerations

- **Feedback Quality**: Ensure LLM-generated feedback is accurate and aligned with domain knowledge.
- **Scalability**: Use LLMs to generate synthetic feedback and reduce human involvement.
- **Interpretability**: Leverage LLMs to provide natural language explanations of agent decisions.
- **Alignment**: Ensure LLM guidance aligns with the agent's objectives and domain constraints.

References: Christiano et al. (2017), OpenAI (2022), Stiennon et al. (2020), Brown et al. (2020)