Reinforcement Learning

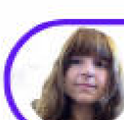
Q-Learning, Deep Q-Learning, REINFORCE, Hybrid

Business applications

Math intuition

Case study (AI-Powered Cyber Resilience)

Hands on: step-by-step and



Julia Lenc

Analytics Journey (links)

Business Analytics. Data Science. Machine Learning

1. Intro: Business and Revenue models. KPIs
2. Business models translated into analytics
3. Techniques: Descriptive, Diagnostic, Predictive, Prescriptive

Diagnostic Techniques

1. Inference: hypotheses testing
2. Unsupervised Learning: clustering, dimensionality reduction, anomalies

Predictive Techniques

1. Supervised learning: overview
2. Preparation: data pre-processing
3. Foundations: model choice and evaluation
4. Regression: linear and non-linear
5. Classification: logistic regression, Naive Bayes, k-NNs
6. Time series: ARIMA, SARIMA, Exponential Smoothing
7. Advanced models: a. Decision Trees, b. SVM, c. (G)ARCH
8. Ensemble: bagging, boosting, stacking
9. Neural Networks: FFNN, CNN, RNN, Transformers

Prescriptive Techniques

1. Prescriptive techniques and reinforcement learning: overview
2. Optimization: Linear, Non-linear and Dynamic programming
2. Simulation: Monte Carlo, Discreet Event Simulation, System Dynamics
3. Probabilistic Methods: Markov Chains, Markov Decision Processes
4. Reinforcement Learning: Q-Learning, Deep Q-Learning, REINFORCE, Hybrid

Julia Lenc

Reinforcement Learning (RL)

Intro

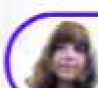
1. Four types of business analytics.
2. What is Reinforcement Learning? Types.
3. Simulation vs Probabilistic Methods vs RL
4. Business applications: Supply Chain, Finance, Marketing...

Math intuition + Case study

1. **Reinforcement Learning**: state, action, reward, policy, value functions, transition probability, discount factor, Bellman equations.
2. **Q-Learning**: update rule, learning rate.
3. **Deep Q-Learning**: deep Q-Network (DQN), experience replay, target network.
4. **REINFORCE**: expected return, gradient ascent.
5. **Hybrid Methods**: Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG).
6. **Case study (2025): AI-Powered Cyber Resilience for Urban Planning**

Hands on

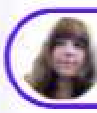
1. Business problem definition.
2. Method choice.
3. Data prepartion.
4. Model setup and Tool choice.
5. Interpretation
6. **Communication: from insight to action!**



Julia Lenc

Introduction

1. Four types of business analytics.
2. What is Reinforcement Learning? Types.
3. Simulation vs Probabilistic Methods vs Reinforcement Learning.
4. Business applications.



Julia Lenc

Types of business analytics

Descriptive

“What has happened?”

Exploratory Data Analysis (EDA), descriptive statistics, visualizations

Diagnostic

“Why did it happen?”

Inference (hypothesis testing). Unsupervised learning

Predictive

“What is likely to happen?”

Supervised learning (forecasting)

Prescriptive

“What should we do?”

Optimization. Simulation. Probabilistic (Markovian) Methods. Reinforcement learning

Read <u>here</u> about business analytics (details)

Julia Lenc

Reinforcement Learning

Definition:

- A family of machine learning techniques for learning how to make decisions by **interacting with an environment**. An agent learns to choose actions in different situations (states) to maximize long-term rewards, often when rules and outcomes are uncertain.
- Core idea: learn from **trial and error**, using feedback (**rewards or penalties**) to improve future decisions.
- Foundation: **Bellman equation**, showing how current rewards and the value of future situations are linked.

Types:

- **Value-based method - basic (Q-Learning)**: the value of taking each action in each situation, e.g., which actions yield highest rewards.
- **Value-based method - advanced (Deep Q-Learning)**: an extension of Q-Learning that uses a deep neural network to estimate Q-values (action values) for each state, allowing the agent to handle large or continuous state spaces where tabular Q-Learning is infeasible.
- **Policy-based methods (REINFORCE)**: a direct mapping from situations to actions -finding the best strategy (policy) without estimating values for all actions.
- **Hybrid methods (Actor-Critic algorithms)**: combine different methods for more efficient or stable solutions.

Julia Lenc

Prescriptive Methods comparison

Simulation

- Scenario testing
- No learning
- Any rules or logic

Probabilistic Methods

- Finding best action
- No learning
- Known and fixed rules

Reinforcement Learning

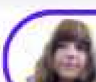
- Discovering what works
- Learning (trial & error)
- Unclear rules

All methods

Uncertainty (input uncertainty, random events, chances outcome)

Scenario exploration

Decision support



Julia Lenc

Business Applications

General rule

- **Value-based RL**: estimates value of different actions. Great for **pricing**.
- **Policy-based RL**: direct mapping from situation to action. Great for continuous/adaptive response, **marketing**.
- **Model-based RL**: learns/plans using a model of the system. Great for network, resource problems, **logistics**.
- **Hybrid**: combines strengths for real-world **messiness**.

Marketing

- **Key applications (great for test market):**
  a) Next-best-action personalization
  b) Optimal message/channel timing
  c) Flexible budget allocation
- **RL goals**: mapping rich customer context to best actions; exploring unknown possibilities for max campaign impact.
- **Best approach: Policy-based RL or Hybrid** (Actor-Critic).

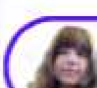
Business Applications

Supply Chain

- **Key applications:**
  a) Adaptive inventory & replenishment automation
  b) Real-time routing & truck loading ("Amazon trucks" case)
  c) Dynamic supplier selection & resource use
- **RL goals**: handling complex logistics, planning ahead, adapting to demand/supply surprises.
- **Best approach: Model-based RL or Hybrid** (Model + Value).

Pricing

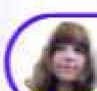
- **Key applications:**
  a) Dynamic, personalized pricing and offers (airlines, TEMU)
  b) Competitive real-time strategy
- **RL goals**: learnsg best prices and offers with constant feedback.
- **Best approach: Value-based** for discreet actions (offer/no offer), **Policy-based** for continuous pricing, **Hybrid (Actor-Critic)** for highly dynamic environments.



Julia Lenc

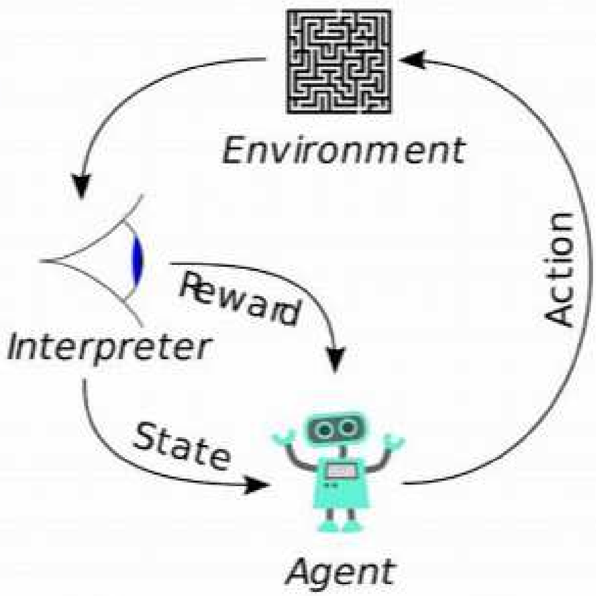
Math Intuition

1. Key principles
2. Q-Learning
3. Deep Q-Learning
4. REINFORCE
5. Hybrid Methods
6. Case study: AI-Powered Cyber Resilience



Julia Lenc

Key principles



Interaction loop

- **Agent** makes decisions.
- Agent chooses an **action** based on its current situation.
- The action affects **environment**.
- **Interpreter** translates environment to **State**, observed by the agent.
- After each action, environment returns a **reward** signal (**+ or -**.)

Key terms

- **State**: the current situation or observation of the environment.
- **Action**: the choice the agent makes in a given state.
- **Reward**: immediate feedback (e.g., +1 for hitting, -1 for missing).
- **Policy**: Agent's strategy - the rule for choosing actions based on states.
- **Deterministic**: always picks the same action.
- **Stochastic**: picks actions with certain probabilities.
- **Transition Probability**: how likely the environment is to move to state s', given current state s and action a.
- **Discount Factor (γ)**: importance of future vs immediate reward. Lower γ - focus on instant results. Higher γ - care more about long-term payoffs.

Bellman Equations: the ❤️ of RL

1. State Value Function: expected total reward starting from **state s** and following **policy π**.

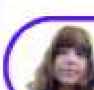
$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s'|s,a)\left[R(s,a) + \gamma V^{\pi}(s')\right]$$

2. Action Value Function: expected total reward starting from state s, taking **action a**, then following policy π.

$$Q^{\pi}(s,a) = \sum_{s' \in S} P(s'|s,a)\left[R(s,a) + \gamma \sum_{a' \in A} \pi(a'|s') Q^{\pi}(s',a')\right]$$

Legend (both equations):

- **s,s':** State, next state
- **a,a':** Action, next action
- **π(a | s):** Probability of taking action a in state s under policy π
- **P(s' | s,a):** Probability of ending up in state s' from state s after action a (transition probability)
- **R(s,a):** Immediate reward for action a in state s
- **γ:** Discount factor, trades off immediate vs. future rewards
- **Vπ(s):** Value of state s under policy π
- **Qπ(s,a):** Value of action a in state s under policy π



Julia Lenc

Bellman Optimality Equations

Optimality means the agent follows the policy that **maximizes** the expected total **reward from any state** - never settling for less.

1. Optimal State Value Function: maximum expected total reward starting from state s by always acting optimally.

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} P(s'|s,a)\left[R(s,a) + \gamma V^*(s')\right]$$

2. Optimal Action Value (Q) Function: maximum expected total reward if you start in state s, take action a and then act optimally.

$$Q^*(s,a) = \sum_{s' \in S} P(s'|s,a)\left[R(s,a) + \gamma \max_{a' \in A} Q^*(s',a')\right]$$

Legend (both equations):

- **s,s'**: state, next state
- **a,a'**: action, next action
- **A**: set of all possible actions
- **P(s' | s,a)**: probability of ending in state s' from s after action a
- **R(s,a)**: immediate (expected) reward for action a in state s
- **γ**: discount factor, favoring sooner rewards
- **V*(s)**: maximum attainable value at state s
- **Q*(s,a)**: maximum attainable value if taking action a in state s
- **max a, a'∈A**: always pick the action that yields the highest value

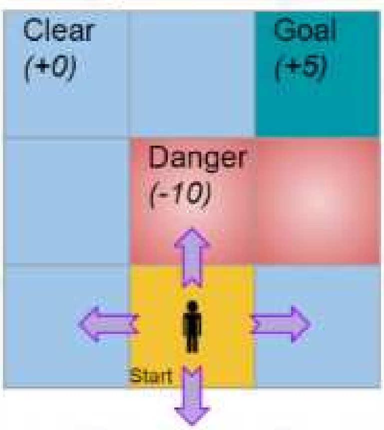
Julia Lenc

Q-Learning

Definition

Q-Learning is a popular **model-free**, **value-based** reinforcement learning algorithm. The agent learns the value of taking each action in each state (Q-values) purely from experience, by **trying actions and observing rewards**, without needing a model of the environment.

Example



The **agent** (person icon) makes decisions (**arrows**: left, right, up, down) in an **environment**.
Each move results in a **reward or penalty** ("Danger", "Goal", "Clear").
The agent learns from the environment by exploring and receiving feedback as rewards.

Business Applications

- **Dynamic pricing** in e-commerce or travel based on market response.
- **Promotion optimization**: testing and learning which promotion leads to the best customer uptake.
- **Inventory management**: learning optimal restocking actions.
- **Automated bidding**: online ads or auctions, selecting best bids over time.

Watchout! Not ideal if actions are expensive or risky to test physically.

Picture from: https://ketanhdoshi.github.io/Reinforcement-Learning-Q-Learning/

Julia Lenc

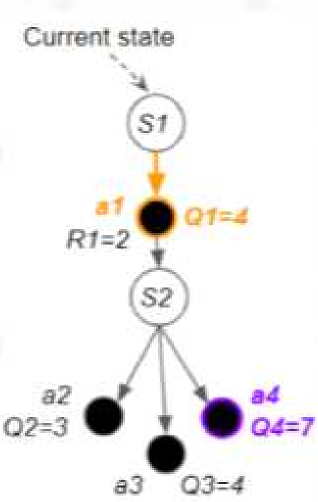
Q-Learning

Formula (Temporal Difference approach)

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

Legend:

- **S**: current state.
- **A**: action taken by the agent.
- **S'**: next state the agent moves to.
- **A'**: the best next action in state S'.
- **R**: reward received for taking action A in state S.
- **γ**: discount factor which balances immediate rewards with future rewards.
- **α**: **learning rate**; shows how much new information affects the old Q-values.



Updating Q-values:

- Bellman equation of optimality $Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s, a\right]$ is used for **update rule** - choosing the max Q for all next actions.
- This example illustrates one action (one step):

  Q1 = Q1 + α (R1 + γ * Q4 - Q1)

  Q1 = 4 + (2 + 7 - 4) = 9

  For simplicity: α = γ = 1

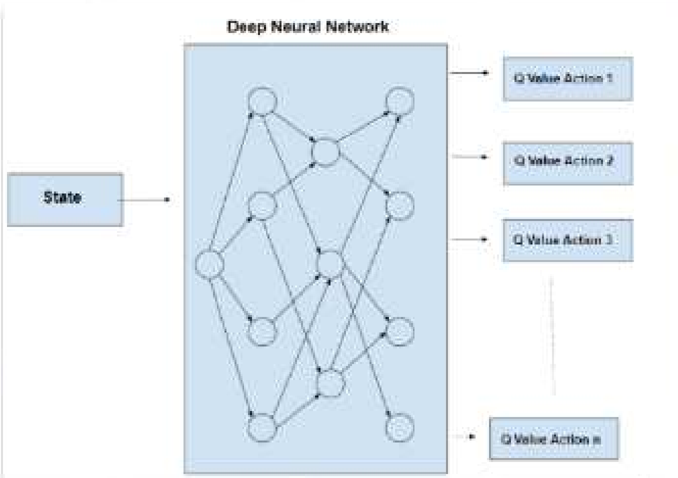
Picture from: https://ketanhdoshi.github.io/Reinforcement-Learning-Q-Learning/

Julia Lenc

Deep Q-Learning

Definition

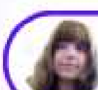
Deep Q-Learning is an extension of Q-Learning that uses a **deep neural network to estimate Q-values** (action values) for each state, allowing the agent to handle large or continuous state spaces where tabular Q-Learning is infeasible. Like Q-Learning, it is **model-free**, **value-based** and learns through **trial-and-error**. Instead of a table, it learns a function mapping states to Q-values for all possible actions.



Key terms

- **Input**: state (features or pixels).
- **Deep Neural Network** computes Q-values for all actions, one forward pass.
- **Output**: Q-value for each possible action. The action with the highest Q-value is selected.
- **Experience Replay** stores past transitions (state, action, reward, next state) in memory and samples random mini-batches to break correlations and stabilize learning.
- **Target Network**: a second neural network, updated less frequently, to provide stable target Q-values during training.

Picture from: https://www.geeksforgeeks.org/deep-q-learning/



Julia Lenc

Deep Q-Learning

How DQN works?

1. **Neural Network**
   - Approximates the Q-value function **Q(s,a;θ)**, where **θ** are the trainable weights.
   - Complex tasks: input can be raw pixels and output is a Q-value for each action.
2. **Experience Replay**
   - Past experiences **(s,a,r,s')** are stored in a replay buffer.
   - Training samples mini-batches randomly from the buffer, breaking correlations in data and improving learning stability.
3. **Target Network**
   - A separate target network (with parameters **θ−**) computes the target Q-values for the loss function. The target network is periodically synchronized with the main network to stabilize updates.
4. **Core formula (loss function)** measures the difference between the predicted Q-values and the target Q-values.

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a';\, \theta^{-}) - Q(s, a;\, \theta)\right)^{2}\right]$$

θ, θ−: weights of the current and target Q-networks
r: reward received after taking action a in state s
γ: discount factor for future rewards
(s, a): the current state and action taken
(s', a'): the next state after action a and the next possible actions in s'

Business applications

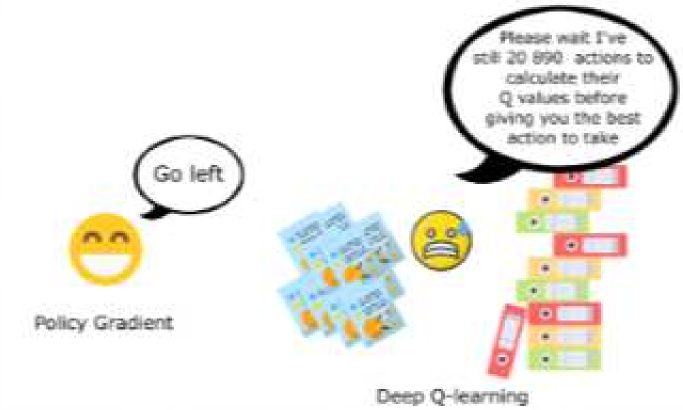
1. **Robotics and automated control with sensors**: self-driving cars, production.
2. **Finance**: stock trading, money and risk management.
3. **Healthcare**: personalized care (treatment planning).


Julia Lenc

REINFORCE

Definition

REINFORCE is a classic **model-free**, **policy-based** reinforcement learning algorithm. Instead of learning state or action values, the agent directly learns a parameterized **policy - how to choose actions in each state** - by sampling trajectories, observing rewards, and adjusting its policy parameters to maximize expected future rewards.



Business Applications

- **Continuous dynamic pricing** in airlines or ride sharing: price can be set anywhere in a range, not just "cheap/medium/expensive" (Q-Learning).
- **Marketing strategy**: personalized offer for each customer at each touchpoint to maximize conversion or retention.
- **Robotics**: automated warehouses using robots to pick, move and sort packages efficiently by directly optimizing policies with sensor feedback.

Picture from: https://www.freecodecamp.org/news/an-introduction-to-policy-gradients-with-cartpole-and-doom-495b5ef2207f/

Julia Lenc

REINFORCE

How REINFORCE works (4 stages)

1. Collect Episodes

The agent uses its current policy to **interact with the environment** and collects several **trajectories** - sequences of states, actions and rewards.

2. Calculate Expected Returns

For each time step in each episode, calculate the total **expected return** (sum of all future rewards, possibly discounted) from that step onward. This return is assigned to all actions and states along the path, showing which led to better outcomes.

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$

Gt: total reward the agent "expects" to get by following policy, starting from time t.

T: the rewards are adding up from the start time **t** to the final episode **T**.

γk−t: discount factor showing how far the reward **rk** is in the future. If k=t, the reward is immediate. As **k** increases (rewards come later), later rewards matter less.

3. Policy Gradient Update

Use the calculated returns to estimate the **policy gradient** - how to change the policy parameters to increase the chance of good actions and decrease the chance of bad ones. Update the policy's parameters in the direction that increases future expected rewards ("**gradient ascent**" on the expected return).

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

θ: policy parameters

α: learning rate

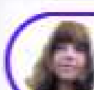
πθ(at I st): probability of taking action atat at state stst , according to the policy

$$\nabla_\theta \log \pi_\theta(a_t|s_t)$$

represents how much the **policy probability** for action **at** at state **st** should be **adjusted** based on the obtained return.

4. Repeat

With the improved policy, repeat the process.



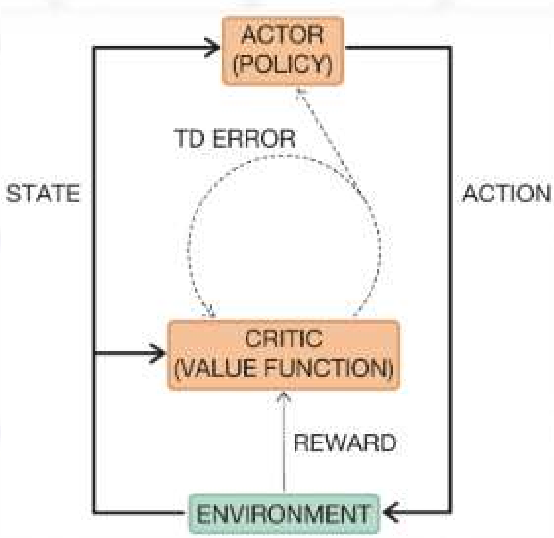
Julia Lenc

Hybrid (Actor-Critic) Methods

Definition

Methods that combine policy-based (actor) and value-based (critic) appoaches:

- The **actor** learns a **policy** $\pi(a \mid s)$ - how to select actions.
- The **critic** learns a **value** function $V(s)$ - how good the chosen actions are.
- By learning both, Actor-Critic methods **leverage the strengths of policy** gradients (exploration) **and value estimation** (low variance, stable learning).



Types

1. **A2C (Advantage Actor-Critic)**
Runs two networks - an actor (policy) and a critic (value). The actor is improved using the advantage function for more stable learning.

2. **DDPG (Deep Deterministic Policy Gradient)**
Uses actor and critic networks for continuous actions, combining Q-learning techniques (experience replay, target networks) with deterministic policy gradients.

Business applications

1. **Robotics:** smooth, precise control of robotic arms or vehicles.
2. **Automated Trading:** managing portfolio actions in continuous spaces.
3. **Personalization:** fine-grained recommendation and content ranking.
4. **Resource Allocation:** dynamic allocation in cloud computing or telecoms.
5. **Simulations:** high-performance control in complex environments.

Picture from: https://cnvrg.io/reinforcement-learning/

Julia Lenc

Case study: AI Cyber Resilience

Challenge

- This digital transformation, while enabling **smart cities** and automated service delivery, has also brought **risks**.
- There are discussions about implementation of AI, but **no example of structured AI framework to address specific challenges of smart cities** - safeguarding the city's essential functions, ensuring public confidence and protecting the economic and social fabric from the significant and growing **threat of cyber warfare**.

Approach

The researcher **tested 5-layer** framework:

1. **Dynamic data collection**: real-time.
2. **ML detection and prediction** to flag unusual activity and forecast potential threats.
3. **Reinforcement learning** to determine the **best course of action**.
4. **Reinforcement learning** for self-healing and continuity mechanisms.
5. **Monitoring ethics and compliance.**

Source: https://www.researchgate.net/publication/392764751_AI-BASED_CYBER_RESILIENCE_FRAMEWORKS_FOR_CRITICAL_URBAN_INFRASTRUCTURE

Julia Lenc

Case study: AI Cyber Resilience

Outcome

The 5-layer AI implementation framework for smart cities:

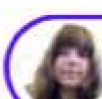
1. **Sensing and data collection**: real-time, continuous stream from IoT sensors, operational systems, network logs, and user interfaces.
2. **ML detection and prediction**: a combination of convolutional neural networks (**CNNs**), support vector machines (**SVMs**) and **anomaly detection** algorithms to to flag unusual activity and forecast potential threats.
3. **Reinforcement learning** and decision trees to determine the **best course of action** and execute automated responses (isolating affected nodes, adjusting access levels, or activating redundancies).
4. **Reinforcement learning** for self-healing and continuity mechanisms (services are rerouted, corrupted data restored, and systems rebooted in a controlled, non-disruptive manner).
5. **Monitoring ethics and compliance**: data protection, fair decision-making, retraining models on updated policies.

Source: https://www.researchgate.net/publication/392764751_AI-BASED_CYBER_RESILIENCE_FRAMEWORKS_FOR_CRITICAL_URBAN_INFRASTRUCTURE

Julia Lenc

Hands On

1. Business problem definition.
2. Method choice.
3. Data preparation
4. Model setup and Tool choice
5. Interpretation
6. Deployment: from insight to action!



Julia Lenc

Stage 1: Business problem

Specific: Define the decision-making challenge in terms of learning from experience and actions over time.

- What **sequential decision** must a system or agentmake under uncertainty?
- Which dynamic **processes** or systems change in response to actions?
- What are the possible **states and actions** the agent can take at each step?
- What are the **rewards** and how do they relate to the business goal?
- Which **performance metrics** (total reward, efficiency, cost, risk)?

Measurable: How will you measure the **agent's success**?

- Cumulative reward or total returns over time.
- Achievement rates of business-specific targets (e.g., service level, profit).
- Comparison benchmarks: baseline behavior, expert policy, random policy...

Achievable: Is RL the right method?

- Can you simulate or collect data from interactions (real or virtual)?
- Are the reward signals and feedback observable and learnable?
- Are the states, actions and rewards well-defined?

Relevant: The **decision makers** are known and the output can drive action.

Time-bound: Identify stakeholders and key **deadlines** for delivery.

Julia Lenc

Stage 2: Method choice

Methods:

- **Q-Learning** for tabular RL for simple environments with discrete states and actions. Example: warehouse robot navigation.
- **Deep Q-Learning** (DQN) handles large or complex state spaces using neural networks. Example: customer support chatbot.
- **REINFORCE**: policy-gradient method for learning stochastic policies directly from rewards. Example: digital marketing campaign optimization.
- **Actor-Critic Methods** (A2C, DDPG, etc.) combine value-based and policy-based learning; suitable for both discrete and continuous action spaces. Example: automated stock trading.

Tools:

- **Python** (numpy, pandas, PyTorch, Tensorflow)
- **RL-specific libraries** (see in Stage 4)

Stage 3: Data preparation (p1)

1. Clarify Agent-Environment Setup

- **Environment**: What is the "world" the agent interacts with?
- **State variables**: What information does the agent receive at each step (observation space)?
- **Action space**: Which actions can the agent choose at decision points?
- **Reward signals**: What feedback (numeric reward/cost) will guide the agent toward desired behavior?

2. Process Sequence Data for Episodes

- **Sequence creation**:
  - Extract or simulate sequences of (state, action, reward) tuples (trajectories/episodes).
  - If using logs: reformulate event logs, sensor data or user journeys into time-ordered transitions.
- **Episode segmentation**:
  - Split long records into episodes (e.g., 1 order, robot run, trading day).
  - Mark episode termination (done/goal state) clearly for RL training.
- **Time consistency & intervals**:

Ensure regular or appropriately sampled time steps.

Julia Lenc

Stage 3: Data preparation (p2)

3. Reward Engineering

- **Reward shaping**: adjust, rescale, or design rewards: make sure learning signals align with business goals and avoid sparse/confusing feedback.
- **Handling sparse or delayed rewards**: intermediate/incremental rewards for long-horizon problems.
- **Clip/normalize rewards** when needed to stabilize learning.

4. Action & State Encoding

- **Discrete vs. continuous**: encode discrete categories; scale/normalize numerical features for neural nets.
- **One-hot or label encoding** for actions and states if needed.

5. Data Robustness, Quality, Features

- Remove or flag **invalid transitions** (impossible actions, illegal states).
- Augment/simulate data for **rare but important scenarios** (e.g., system failures).
- Check for **duplicates, missing values and out-of-bounds** (anomalies in sequences.

Julia Lenc

Stage 4: Model Set Up (links)

Python tutorials

- Q-Learning: script
- Deep Q-Learning (DQN): script
- REINFORCE: script
- Actor-Critic: script

Core Python Libraries for RL:

- NumPy, pandas - general
- TensorFlow, PyTorch - deep learning

RL-Specific Libraries & Frameworks:

- OpenAI Gymnasium
- Stable Baselines3
- Ray RLlib
- TF-Agents (by Google/DeepMind)

Stage 5: Interpretation (p1)

1. **Policy Performance Over Time**

- Track how the agent's **policy improves** across training episodes (reward curves, loss curves).
- Plot episode rewards, steps to goal, % of successful episodes over time.
- Example: Monitor how quickly a warehouse robot with RL learns optimal path efficiency and fewer collisions per episode.

2. **State and Action Distributions**

- Analyze which states and actions are **most/least frequent** under the learned policy.
- Identify "**preferred**" or "**avoided**" states and actions—insight into agent strategy (e.g., does a trading RL agent avoid volatile stocks?).
- Example: In customer support RL chatbot, see which answers/actions are selected most often in real use.

3. **Sensitivity to Starting Conditions**

- Test learned agent's robustness to different initial states or environments.
- Evaluate how much agent behavior and results change when started in a new, unexpected, or adverse situation.
- Example: Does a self-driving RL agent successfully adapt when traffic is denser than in training?

Julia Lenc

Stage 5: Interpretation (p2)

4. **Policy Comparisons and Ablations**
   - **Compare baselines**: RL policy vs. random policy, vs. rule-based, vs. previous best.
   - **Quantify gains**: cumulative reward, time to goal, policy stability under perturbations.
   - Example: Compare Deep Q-Learning agent's profits to simple buy-and-hold in stock trading.

5. **Scenario and What-if Analysis**
   - Re-run simulations with **modified** rewards, altered environment dynamics, or new constraints to see **how the RL agent adapts**.
   - Understand **failure modes**: when does the agent's policy break down?
   - Example: After retraining an RL warehouse bot to handle new obstacles, test edge cases to check for learned adaptability.

6. **Extras!**
   - **Policy visualization**: for low-dimensional problems, visualize state→action mapping or "**heatmaps**" of action probability.
   - **Value function analysis**: inspect Q-values, advantage functions or policy gradients to understand agent reasoning.
   - **Exploration vs. exploitation**: track if the agent continues to explore or settles on a small action set.

Julia Lenc

Stage 6: Action! (p1)

1. **Lead with business value**:
   - Emphasize the **measurable business impact** from smart automation and decision-making: higher **revenue**, lower **costs**, faster **response**, improved customer **retention**, fewer **mistakes**.
   - Example: "With this RL-powered pricing strategy, gross margin increases by 3% with no drop in sales volume."

2. **Use stakeholders' language**:
   - Translate RL results into relevant terms for finance, sales, marketing or supply chain: **ROI, sales growth,** campaign conversion, time savings, risk reduction, order fulfillment rates.
   - Example: "The RL agent's order routing is projected to reduce late deliveries from 1 in 20 to 1 in 40."

3. **Show simple visuals**:
   - Use clear bar/line charts, funnel plots, or before/after comparisons. Show impact on KPIs or processes.
   - Example Scenario: "If we let the RL tool optimize discounts, average time to clear inventory drops from 6 to 4 weeks."

Julia Lenc

Stage 6: Action! (p2)

4. **Tell a Story**:
   - Explain decisions as **experiments** and **learning**:
   - Example 1: "We let our system test hundreds of pricing/campaign options and it found patterns we'd miss by hand."
   - Example 2: "With dynamic RL scheduling, trucks are dispatched smarter. That means more on-time orders and lower fuel use"

5. **Highlight sensitivities and trade-offs**:
   - Surface **what-ifs** and **resource-use** questions:

     "What if demand spikes? The RL system's policy is robust - delay grows by only 1 day, not 5."

     "After a point, spending more on ads gives less lift - the model tells us where to stop."
   - Show **diminishing returns, risk** or **opportunity cost**.

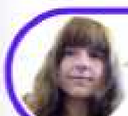
6. **Show what's next**:
   - Recommend **real-world trial, pilot or phased rollout** based on insight:
     Example: "Let's use RL for holiday season pricing - run a pilot in two regions before scaling."
   - Identify **quick wins** and **risks**:
     Example: "If shoppers shift online faster than expected, the RL agent's recommendations still adapt in real-time."

Julia Lenc

Did you find it useful?

Save
Share
Follow

Analytics, Market Research, Machine Learning and AI


Julia Lenc