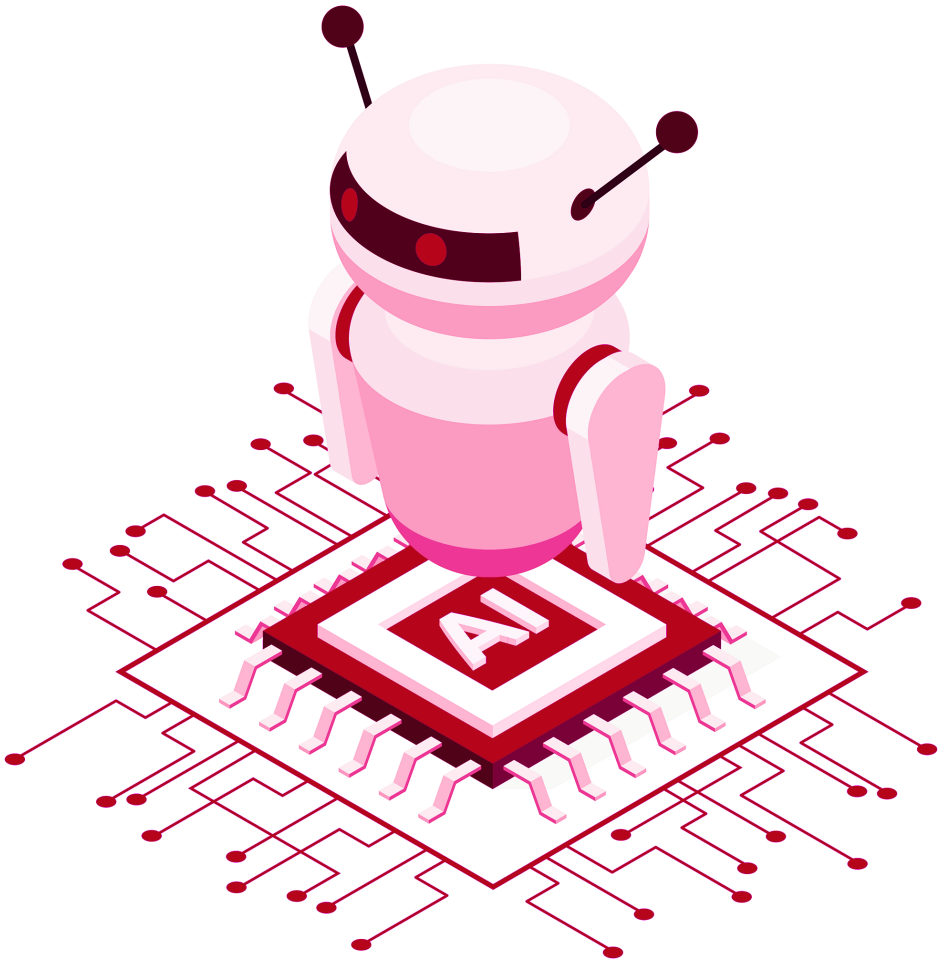

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 6:

Multi-Armed Bandits

By:

Ashkan Majidi

400109984

Spring 2025

Contents

Grading

The grading will be based on the following criteria, with a total of 105.25 points:

Task	Points
Task 1: Oracle Agent	3.5
Task 2: Random Agent	2
Task 3: Explore-First Agent	5.75
Task 4: UCB Agent	7
Task 5: Epsilon-Greedy Agent	2.25
Task 6: LinUCB Agent	27.5
Task 7: Final Comparison and Analysis	6
Task 8: Final Deep-Dive Questions	41.25
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus 1: Writing your report in Latex	10

1 Task 1: Oracle Agent

Q: How might the performance of different agents change if the distribution of probabilities were not uniform?

A: If the distribution of probabilities is not uniform, agents that rely on exploration (e.g., UCB, Epsilon-Greedy) may take longer to identify the best arm, especially if the best arm has a much higher probability than others. Random agents will perform poorly as they cannot exploit the skewed distribution. Algorithms like LinUCB, which leverage additional context, may perform better in identifying the best arm in such scenarios.

Q: Why does the MAB environment use a simple binary reward mechanism (1 or 0)?

A: The MAB environment uses a binary reward mechanism because it simplifies the problem and aligns with many real-world scenarios where outcomes are binary (e.g., success/failure, click/no-click). This approach makes it easier to model and analyze the performance of different algorithms while focusing on the exploration–exploitation trade-off.

Q: What insight does the oracle reward give us about the best possible performance?

A: The oracle reward represents the theoretical upper bound on the performance of any agent in the environment. It provides a benchmark to evaluate the efficiency of other algorithms and highlights the maximum achievable reward under ideal conditions.

Q: Why is the oracle considered "cheating" in a practical sense?

A: The oracle is considered "cheating" because it has access to privileged information (e.g., the exact probabilities of rewards for each arm) that is not available to real-world agents. In practical scenarios, agents must learn these probabilities through exploration, making the oracle an unrealistic baseline.

2 Task 2: Random Agent (RndAg)

Q: Why is the reward of the random agent generally lower and highly variable?

A: The random agent selects actions uniformly at random without any consideration of the reward probabilities of the arms.

Q: How might you improve a random agent without using any learning mechanism?

A: You could improve a random agent by introducing a weighted random selection mechanism, where arms with higher historical rewards are given higher probabilities of being selected. Alternatively, you could use a heuristic-based approach, such as prioritizing arms with higher initial probabilities (if known) or periodically re-evaluating arm selection probabilities based on observed outcomes.

3 Task 3: Explore-First Agent (ExpFstAg)

Q: Why might the early exploration phase (e.g., 5 steps) lead to high fluctuations in the reward curve?

A: The early exploration phase involves selecting actions randomly, which can lead to inconsistent rewards due to the stochastic nature of the environment. If the best arm is not sampled during this phase, the agent may fail to identify it, resulting in high variability in the reward curve.

Q: What are the trade-offs of using a fixed exploration phase?

A: A fixed exploration phase ensures that the agent gathers sufficient data to estimate the rewards of different arms. However, if the phase is too short, the agent may not explore enough to identify the best arm. Conversely, if it is too long, the agent delays exploitation, potentially missing out on higher cumulative rewards. Balancing exploration and exploitation is critical to optimizing performance.

Q: How does increasing 'max_ex' affect the convergence of the agent's performance?

A: Increasing 'max_ex' allows the agent to explore more arms before committing to exploitation. This can improve the accuracy of identifying the best arm, leading to better long-term performance. However, it delays the exploitation phase, which may result in lower cumulative rewards in the short term. The trade-off is between gathering sufficient information and starting exploitation early enough to maximize rewards.

Additionally, note that if max_ex is more than a number (for example 50 in our experiment) the reward wont change much if we increase the max_ex

Q: In real-world scenarios, what challenges might arise in selecting the optimal exploration duration?

A: In real-world scenarios, the optimal exploration duration depends on factors like the variability of rewards, the number of arms, and the time horizon of the task. Challenges include:

- **Finite time horizon:** Over-exploration can waste valuable time that could be spent exploiting the best arm.
- **Dynamic environments:** If the reward probabilities change over time, a fixed exploration duration may become suboptimal.

4 Task 4: UCB Agent (UCB_Ag)

Q: Under what conditions might an explore-first strategy outperform UCB, despite UCB's theoretical optimality?

A: An explore-first strategy might outperform UCB in scenarios with a finite time horizon where early exploitation is critical. For example:

- **Short-term tasks:** If the time horizon is small, UCB's conservative exploration may delay exploitation, leading to suboptimal cumulative rewards.
- **Low variance environments:** In environments where the reward probabilities are close to deterministic, explore-first strategies can quickly identify the best arm and exploit it effectively.

Q: How do the design choices of each algorithm affect their performance in short-term versus long-term scenarios?

A:

Explore-First Strategy: Performs well in short-term scenarios if the exploration phase is appropriately tuned. However, it may fail in long-term scenarios if the exploration phase is too short to identify the best arm.

UCB: Balances exploration and exploitation dynamically, making it optimal in long-term scenarios. However, its conservative exploration bonus can lead to slower convergence in short-term tasks.

Epsilon-Greedy: The choice of ϵ affects performance. A high ϵ favors exploration, which is beneficial in long-term scenarios but reduces short-term rewards. A low ϵ favors exploitation, which is better for short-term tasks but risks missing the best arm.

Q: What impact does increasing the exploration phase to 20 steps have compared to 5 steps?

A: Increasing the exploration phase to 20 steps allows the agent to gather more information about the reward probabilities of each arm, improving the likelihood of identifying the best arm And as you see it outperforms ExpFstAg.

Q: How can you determine the optimal balance between exploration and exploitation in practice?

A: In Explore-First Agent (not generally just and just for our case) a prior knoledge of environment is cruical. for example of we know our max timesteps we can set max_ex to be a division of that. If we can run multiple times and use hyperparameter tunning it could be an approch to find best number of exploration steps.

Q: We know that UCB is optimal. Why might ExpFstAg perform better in practice?

A: ExpFstAg can perform better in practice under finite-time scenarios because it allows for early exploitation after a fixed exploration phase. This can lead to higher cumulative rewards in the short term, especially when the exploration phase is well-tuned to the environment. Hyperparameter tuning (e.g., setting 'max_ex' appropriately) can help ExpFstAg quickly identify the best arm and exploit it, avoiding the conservative exploration of UCB, which may delay exploitation. Additionally, in environments with low variance or a clear gap between the best and other arms, ExpFstAg's simpler strategy can yield better practical performance compared to UCB's more cautious approach.

5 Task 5: Epsilon-Greedy Agent (EpsGdAg)

Q: Why does a high ϵ value result in lower immediate rewards?

A: A high ϵ value increases the probability of selecting a random action, which means the agent is more likely to choose suboptimal arms instead of exploiting the best-known arm. This leads to lower immediate rewards as the agent sacrifices exploitation for exploration.

Q: What benefits might there be in decaying ϵ over time?

A: Decaying ϵ over time allows the agent to explore more in the early stages when it has limited knowledge about the environment and gradually shift toward exploitation as it gains confidence in its reward estimates. This balances the exploration–exploitation trade-off, improving long-term performance by reducing unnecessary exploration in later stages.

Q: How do the reward curves for different ϵ values reflect the exploration–exploitation balance?

A: The reward curves for different ϵ values show how the agent balances exploration and exploitation. A higher ϵ results in more exploration and help the agent discover better arms in the long run. Conversely, a lower ϵ favors exploitation, leading to higher immediate rewards but risking suboptimal performance if the agent prematurely commits to a suboptimal arm. As you see more

Q: Under what circumstances might you choose a higher ϵ despite lower average reward?

A: A higher ϵ might be chosen in scenarios where exploration is critical, such as:

- **Dynamic environments:** Where reward probabilities change over time, requiring continuous exploration to adapt.
- **High uncertainty:** When the agent has limited prior knowledge about the environment and needs to explore more to gather sufficient information.
- **Short-term tasks with high variance:** To ensure the agent samples enough arms to avoid missing potentially high-reward arms.

6 Task 6: LinUCB Agent (Contextual Bandits)

Q: How does LinUCB leverage context to outperform classical bandit algorithms?

A: LinUCB leverages context by incorporating additional features (contextual information) to estimate the reward for each arm. Unlike classical bandit algorithms, which rely solely on past rewards, LinUCB assumes a linear relationship between the context and the reward. This allows it to generalize across similar contexts, enabling better decision-making even with limited data. By using context, LinUCB can identify patterns and exploit them to select the best arm more effectively, especially in environments where the reward depends on external factors.

Q: What is the role of the α parameter in LinUCB, and how does it affect the exploration bonus?

A: The α parameter in LinUCB controls the exploration bonus added to the estimated reward for each arm. A higher α increases the exploration bonus, encouraging the agent to explore less frequently chosen arms. This helps the agent gather more information about uncertain arms. Conversely, a lower α reduces the exploration bonus, leading to more exploitation of the best-known arm. The choice of α balances the trade-off between exploration and exploitation, with higher values favoring exploration and lower values favoring exploitation.

7 Task 7: Final Comparison and Analysis

Q: What does α affect in LinUCB?

A: The α parameter in LinUCB controls the exploration bonus added to the estimated reward for each arm. A higher α encourages more exploration by increasing the confidence interval, making the agent more likely to try arms with higher uncertainty. Conversely, a lower α reduces the exploration bonus, leading to more exploitation of the best-known arms. The value of α directly affects the balance between exploration and exploitation.

Q: Do the reward curves change with α? Explain why or why not.

A: Yes, the reward curves change with α. A higher α results in more exploration, which can lead to slower convergence and lower immediate rewards but may improve long-term performance by identifying the best arm more reliably. A lower α leads to faster exploitation, resulting in higher immediate rewards but risks missing the best arm if the agent does not explore enough. The shape of the reward curve reflects how the agent balances exploration and exploitation based on the value of α.

Q: Based on your experiments, does LinUCB outperform the standard UCB algorithm? Why or why not?

A: LinUCB can outperform the standard UCB algorithm in scenarios where the reward depends on contextual features. By leveraging context, LinUCB can make more informed decisions, especially in environments

where the relationship between context and reward is linear or approximately linear. However, in non-contextual or low-dimensional settings, LinUCB may not provide significant advantages and could even underperform due to the overhead of maintaining and updating additional parameters. The performance depends on the quality of the context and the linearity of the reward model.

Q: What are the key limitations of each algorithm, and how would you choose between them for a given application?

A:

- **UCB:**
  - **Limitations:** Assumes stationarity and does not leverage contextual information. May explore too conservatively in finite-horizon scenarios, leading to slower convergence.
  - **When to Use:** Suitable for non-contextual bandit problems with stationary reward distributions and when computational simplicity is desired.
- **LinUCB:**
  - **Limitations:** Assumes linear relationship $\mathbf{r}_t = \mathbf{x}_t^\top \theta + \epsilon_t$ between context and reward. Can overfit in high-dimensional contexts ($d \gg T$) and has $\mathcal{O}(d^2)$ computational overhead.
  - **When to Use:** Ideal when rewards depend linearly on informative features and resources allow for the computational cost.
- **Choosing Between Them:**
  - Prefer **UCB** when:
    - Problem is non-contextual (no feature vectors)
    - Stationarity assumption holds
    - Computational efficiency is critical
  - Prefer **LinUCB** when:
    - Contextual information is available and meaningful
    - Reward-context relationship is approximately linear
    - Sufficient data and computational resources exist

8 Task 8: Final Deep-Dive Questions

- **Finite-Horizon Regret and Asymptotic Guarantees** (4 points)

  Many algorithms (e.g., UCB) are analyzed using asymptotic (long-term) regret bounds. In a finite-horizon scenario (say, 500–1000 steps), explain intuitively why an algorithm that is asymptotically optimal may still yield poor performance. What trade-offs arise between aggressive early exploration and cautious long-term learning? Deep Dive: Discuss how the exploration bonus, tuned for asymptotic behavior, might delay exploitation in finite time, leading to high early regret despite eventual convergence.

- **Hyperparameter Sensitivity and Exploration–Exploitation Balance** (4.5 points)

  Consider the impact of hyperparameters such as ϵ in ϵ-greedy, the exploration constant in UCB, and the α parameter in LinUCB. Explain intuitively how slight mismatches in these parameters can lead to either under-exploration (missing the best arm) or over-exploration (wasting pulls on suboptimal arms). How would you design a self-adaptive mechanism to balance this trade-off in practice? Deep Dive: Provide insight into the "fragility" of these parameters in finite runs and how a meta-algorithm might monitor performance indicators (e.g., variance in rewards) to adjust its exploration dynamically.

- **Context Incorporation and Overfitting in LinUCB** (4 points)

  LinUCB uses context features to estimate arm rewards, assuming a linear relation. Intuitively, why might this linear assumption hurt performance when the true relationship is complex or when the context is high-dimensional and noisy? Under what conditions can adding context lead to worse performance than classical (context-free) UCB? Deep Dive: Discuss the risk of overfitting to noisy or irrelevant features, the curse of dimensionality, and possible mitigation strategies (e.g., dimensionality reduction or regularization).

- **Adaptive Strategy Selection** (4.25 points)

  Imagine designing a hybrid bandit agent that can switch between an explore-first strategy and UCB based on observed performance. What signals (e.g., variance of reward estimates, stabilization of Q-values, or sudden drops in reward) might indicate that a switch is warranted? Provide an intuitive justification for how and why such a meta-strategy might outperform either strategy alone in a finite-time setting. Deep Dive: Explain the challenges in detecting when exploration is "enough" and how early exploitation might capture transient improvements even if the long-term guarantee favors UCB.

- **Non-Stationarity and Forgetting Mechanisms** (4 points)

  In non-stationary environments where reward probabilities drift or change abruptly, standard bandit algorithms struggle because they assume stationarity. Intuitively, explain how and why a "forgetting" or discounting mechanism might improve performance. What challenges arise in choosing the right decay rate, and how might it interact with the exploration bonus? Deep Dive: Describe the delicate balance between retaining useful historical information and quickly adapting to new trends, and the potential for "chasing noise" if the decay is too aggressive.

- **Exploration Bonus Calibration in UCB** (3.75 points)

  The UCB algorithm adds a bonus term that decreases with the number of times an arm is pulled. Intuitively, why might a "conservative" (i.e., high) bonus slow down learning—even if it guarantees asymptotic optimality? Under what circumstances might a less conservative bonus be beneficial, and what risks does it carry? Deep Dive: Analyze how a high bonus may force the algorithm to continue sampling even when an arm's estimated reward is clearly suboptimal, thereby delaying convergence. Conversely, discuss the risk of prematurely discarding an arm if the bonus is too low.

- **Exploration Phase Duration in Explore-First Strategies** (4 points)

  In the Explore-First agent (ExpFstAg), how does the choice of a fixed exploration period (e.g., 5 vs. 20 steps) affect the regret and performance variability? Provide a scenario in which a short exploration phase might yield unexpectedly high regret, and another scenario where a longer phase might delay exploitation unnecessarily. Deep Dive: Discuss how the "optimal" exploration duration can depend heavily on the underlying reward distribution's variance and the gap between the best and other arms, and why a one-size-fits-all approach may not work in practice.

- **Bayesian vs. Frequentist Approaches in MAB** (4 points)

  Compare the intuition behind Bayesian approaches (such as Thompson Sampling) to frequentist methods (like UCB) in handling uncertainty. Under what conditions might the Bayesian approach yield superior practical performance, and how do the underlying assumptions about prior knowledge

influence the exploration–exploitation balance? Deep Dive: Explore the benefits of incorporating prior beliefs and the risk of bias if the prior is mis-specified, as well as how Bayesian updating naturally adjusts the exploration bonus as more data is collected.

- **Impact of Skewed Reward Distributions** (3.75 points)

  In environments where one arm is significantly better (skewed probabilities), explain intuitively why agents like UCB or ExpFstAg might still struggle to consistently identify and exploit that arm. What role does variance play in these algorithms, and how might the skew exacerbate errors in reward estimation? Deep Dive: Discuss how the variability of rare but high rewards can mislead the agent's estimates and cause prolonged exploration of suboptimal arms.

- **Designing for High-Dimensional, Sparse Contexts** (5 points)

  In contextual bandits where the context is high-dimensional but only a few features are informative, what are the intuitive challenges that arise in using a linear model like LinUCB? How might techniques such as feature selection, regularization, or non-linear function approximation help, and what are the trade-offs involved? Deep Dive: Provide insights into the risks of overfitting versus underfitting, the increased variance in estimates from high-dimensional spaces, and the potential computational costs versus performance gains when moving from a simple linear model to a more complex one.

Finite-Horizon Regret and Asymptotic Guarantees

Answer: In finite-horizon scenarios, asymptotically optimal algorithms like UCB may perform poorly because they prioritize long-term learning over short-term gains. The exploration bonus in UCB is designed to ensure that all arms are sufficiently explored, which can delay exploitation of the best arm. This leads to high early regret, as the algorithm continues to sample suboptimal arms even when the best arm is evident. The trade-off lies in balancing aggressive early exploration (to gather information) with cautious long-term learning (to minimize regret over time).

Deep Dive: The exploration bonus in UCB, tuned for asymptotic behavior, may cause the algorithm to over-explore in finite time, leading to suboptimal performance. In contrast, strategies that exploit earlier (e.g., explore-first) may achieve better short-term rewards but risk missing the best arm in the long run.

Hyperparameter Sensitivity and Exploration-Exploitation Balance

Answer: Hyperparameters like ϵ in ϵ-greedy, the exploration constant in UCB, and α in LinUCB directly control the balance between exploration and exploitation. Slight mismatches can lead to under-exploration (missing the best arm) or over-exploration (wasting resources on suboptimal arms). A self-adaptive mechanism could monitor performance indicators like reward variance or convergence of Q-values to dynamically adjust these parameters.

Deep Dive: Hyperparameters are fragile in finite runs because they are often tuned for specific environments. A meta-algorithm could dynamically adjust exploration based on observed metrics, such as increasing exploration when variance in rewards is high or reducing it when Q-values stabilize.

Context Incorporation and Overfitting in LinUCB

Answer: The linear assumption in LinUCB can hurt performance when the true relationship between context and reward is complex or when the context is high-dimensional and noisy. Adding context can lead to worse performance than classical UCB if the context introduces irrelevant or noisy features, leading to overfitting.

Deep Dive: High-dimensional contexts increase the risk of overfitting and the curse of dimensionality. Mitigation strategies include dimensionality reduction (e.g., PCA), regularization (e.g., L2 penalty), or switching to non-linear models like neural networks. However, these approaches trade off computational efficiency for improved accuracy.

Adaptive Strategy Selection

Answer: A hybrid agent could switch between explore-first and UCB based on signals like reward variance, stabilization of Q-values, or sudden drops in reward. For example, high variance in rewards might indicate the need for more exploration, while stable Q-values suggest it is time to exploit.

Deep Dive: Detecting when exploration is "enough" is challenging because it depends on the environment's dynamics. Early exploitation might capture transient improvements, but it risks missing long-term gains. A meta-strategy could outperform by dynamically balancing exploration and exploitation based on observed performance.

Non-Stationarity and "Forgetting" Mechanisms

Answer: In non-stationary environments, a forgetting mechanism (e.g., exponential decay) helps the agent adapt to changes by giving more weight to recent observations. However, choosing the right decay rate is critical: too aggressive a decay may cause the agent to chase noise, while too slow a decay may prevent adaptation.

Deep Dive: The balance between retaining historical information and adapting to new trends is delicate. Forgetting mechanisms must be tuned to the rate of change in the environment. Combining decay with adaptive exploration bonuses can improve performance in dynamic settings.

Exploration Bonus Calibration in UCB

Answer: A conservative (high) exploration bonus in UCB slows down learning by forcing the algorithm to sample suboptimal arms excessively, delaying convergence. A less conservative bonus can speed up learning but risks prematurely discarding arms that might be optimal.

Deep Dive: High bonuses ensure asymptotic optimality but lead to high early regret. Conversely, low bonuses may cause the algorithm to exploit too early, missing the best arm. The challenge lies in calibrating the bonus to balance exploration and exploitation effectively.

Exploration Phase Duration in Explore-First Strategies

Answer: A short exploration phase may yield high regret if the agent fails to identify the best arm, especially in environments with high variance. A longer phase may delay exploitation unnecessarily, reducing cumulative rewards.

Deep Dive: The optimal exploration duration depends on the reward distribution's variance and the gap between the best and other arms. A one-size-fits-all approach is ineffective; adaptive strategies that adjust exploration duration based on observed performance are more robust.

Bayesian vs. Frequentist Approaches in MAB

Answer: Bayesian approaches like Thompson Sampling incorporate prior beliefs and update them as data is collected, naturally balancing exploration and exploitation. They often outperform frequentist methods like UCB in environments with well-specified priors or high uncertainty.

Deep Dive: Bayesian methods adapt exploration bonuses dynamically based on posterior distributions, while frequentist methods rely on fixed rules. However, Bayesian approaches are sensitive to mis-specified priors, which can bias results. Frequentist methods are more robust in such cases but may require more exploration.

Impact of Skewed Reward Distributions

Answer: In skewed environments, agents like UCB or ExpFstAg may struggle to identify the best arm due to high variance in reward estimates. Rare but high rewards can mislead the agent, causing prolonged exploration of suboptimal arms.

Deep Dive: Skewed distributions exacerbate errors in reward estimation, as the agent may overestimate the value of arms with occasional high rewards. Strategies that incorporate variance reduction or robust reward estimation can mitigate this issue.

Designing for High-Dimensional, Sparse Contexts

Answer: In high-dimensional contexts, LinUCB risks overfitting to noisy or irrelevant features. Techniques like feature selection, regularization, or non-linear models can improve performance but introduce trade-offs in computational cost and interpretability.

Deep Dive: High-dimensional spaces increase variance in parameter estimates, leading to unstable decisions. Dimensionality reduction (e.g., PCA) reduces noise but may discard useful information. Regularization (e.g., L2 penalty) mitigates overfitting but may underfit sparse contexts. Non-linear models improve accuracy but are computationally expensive and harder to interpret.