Adaptive RBF Networks for Robust Feature Selection in Inverse Reinforcement Learning

Ashiq, Ajmal, Abishek Chakravarthy
Indian Institute of Information Technology, Design and Manufacturing
Kancheepuram, India
Email: {cs22b2021, cs22b2046, cs22b2054}@iiitdm.ac.in

***Abstract*—This report presents an adaptive Radial Basis Function (RBF) network approach for Inverse Reinforcement Learning (IRL), addressing key challenges in feature selection and reward approximation. Building upon recent work in automated feature selection for IRL [3], we propose a hybrid methodology that combines: (1) a Multi-Armed Bandit framework for optimal RBF center selection, and (2) four kernel width adaptation techniques. Our method extends polynomial feature-based IRL by introducing non-linear basis functions while maintaining interpretability. Experimental results on gymnasium environments demonstrate significant improvements over polynomial methods in the Pendulum environment and successfully matching expert performance in CartPole. The adaptive width methods show particular effectiveness in capturing complex reward structures while avoiding overfitting.**

***Index Terms*—Inverse Reinforcement Learning, Radial Basis Functions, Feature Selection, Reward Learning, Adaptive Kernels**

I. Introduction

Inverse Reinforcement Learning (IRL) seeks to infer an agent's underlying reward function from observed behavior, enabling applications ranging from autonomous driving to human-robot collaboration. A core challenge in IRL lies in feature selection and reward representation: the quality of the learned reward function heavily depends on the chosen feature space, yet manually designing expressive features is often impractical. While linear and polynomial basis functions have shown promise [3], many real-world rewards exhibit non-linear, multi-modal, or sparse structures that demand more flexible representations. Current methods struggle to balance expressivity, interpretability, and computational efficiency, limiting their scalability to complex tasks.

This work bridges the gap by introducing an adaptive framework that combines the grounding of polynomial IRL with the flexibility of radial basis function (RBF) networks. Our key contributions include:

- Extending polynomial feature IRL with adaptive RBF networks to capture non-linear rewards while preserving interpretability.
- Introducing a principled center selection mechanism via Multi-Armed Bandits (MABs) to dynamically prioritize relevant features.
- Developing four kernel width adaptation methods for robust reward approximation.

By addressing these challenges, our approach not only improves reward accuracy in non-linear settings but also offers a generalizable framework for feature adaptation in IRL. This advancement has broader implications for domains where reward inference is critical, such as imitation learning, preference modeling.

II. Related Work

Our approach builds directly on Baimukashev et al.'s work [3] which demonstrated that polynomial basis functions can effectively match statistical moments in IRL. While their method shows strong performance with quadratic features, our RBF-based approach offers:

- Better handling of non-linear reward structures
- Localized feature responses for improved interpretability
- Adaptive capacity through kernel width tuning

III. Methodology

A. System Architecture

As shown in Figure 2, our pipeline consists of three core components.

B. Multi-Armed Bandit for RBF Center Selection

The challenge of selecting the optimal number of Radial Basis Function (RBF) centers, denoted as K, is critical in balancing model complexity and generalization. A small number of centers may result in underfitting, whereas an excessively large number of centers may lead to overfitting and increased computational cost. To address this challenge, we propose leveraging a Multi-Armed Bandit (MAB) framework for selecting K, which dynamically adapts to the task at hand.

In our approach, we use a UCB (Upper Confidence Bound) variant of MAB for optimal K selection, which considers both exploration and exploitation. The bandit algorithm explores different values of K and exploits the one that leads to the best reward performance based on previous experiences.

1) MAB-Based K Selection Procedure: The MAB framework works by associating each K candidate (e.g., $K \in \{2, 3, 4, 5\}$) with an "arm" of the bandit. Each arm's reward is calculated based on the performance of the learned reward function with that particular K, which can be evaluated using metrics such as the silhouette score, which measures the quality of clustering in state space.

At each time step, the bandit selects the candidate K that maximizes the upper confidence bound, balancing between

exploring unexplored values of K and exploiting the best-performing candidate. Over time, the bandit converges to the optimal value of K, ensuring that the RBF network captures the relevant features without excessive complexity.

The procedure works as follows:

- Initialize a bandit with K candidates (for example, 2, 3, 4, 5).
- For each candidate K, generate a clustering of the expert demonstrations (e.g., using K-means clustering) and compute the silhouette score as the reward signal.
- Use the bandit's UCB1 algorithm to select a candidate K for exploration.
- After several rounds of exploration, the bandit will settle on the K that maximizes performance, guiding the learning of the RBF centers.

2) UCB Algorithm: The UCB algorithm provides an efficient way to balance exploration and exploitation by selecting the K with the highest upper confidence bound at each step, based on the cumulative rewards observed so far. The UCB strategy is defined as follows:

$$\text{UCB}_i(t) = \hat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}},$$

where: - $\hat{\mu}_i$ is the average reward of arm i (in our case, the performance metric like the silhouette score), - n_i is the number of times arm i has been played (i.e., how many times we've selected $K = i$), - t is the current round of exploration.

This equation ensures that arms with higher uncertainty (less exploration) are selected more often, enabling the bandit to explore a broader range of K candidates in the initial stages of the process.

C. RBF Feature Formulation

We extend the standard reward representation [3] by replacing fixed polynomial features with adaptive radial basis functions (RBFs):

$$R(s) = \sum_{i=1}^{K} w_i \phi_i(s), \quad \phi_i(s) = \exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right), \quad (1)$$

where $\phi_i(s)$ are RBFs centered at c_i with bandwidth σ_i. Unlike polynomial bases, which struggle with local non-linearities and sparse data [1], RBFs offer three key advantages:

- **Local Sensitivity**: Exponential decay prioritizes nearby states, capturing fine-grained reward structures
- **Adaptivity**: Centers (c_i) and widths (σ_i) can be optimized for the task
- **Interpretability**: Each RBF corresponds to a "region of interest" in the state space

D. Kernel Width Adaptation

Fixed kernel widths often lead to overfitting (narrow σ_i) or oversmoothing (wide σ_i). Our four methods address this:

- **Density-Adaptive**:

  $\sigma_i \propto \text{local density}^{-1/d}$ (for uniform coverage in sparse regions)

  Possible Use Case: Environments with irregular state distributions (e.g., maze navigation)
- **Cluster-Adaptive**:

  $\sigma_i = \text{median}(\|x - c_i\|)$ (scale to cluster dispersion)

  Possible Use Case: Distinct behavioral modes (e.g., driving lanes or gait transitions)
- **Covariance-Aware**:

  $\Sigma_i = \text{Cov}(X_i)$ (elliptical kernels for anisotropic features)

  Possible Use Case: States with correlated dimensions (e.g., joint angles in robotics)
- **Learned**:

  $\sigma_i \leftarrow \sigma_i - \eta \nabla_{\sigma_i} \mathcal{L}$ (gradient-based or evolutionary)

  Possible Use Case: High-precision tasks where reward sensitivity is unknown *a priori*

Smaller σ_i improves resolution but risks overfitting to noise, while larger σ_i smooths rewards but may miss critical details. Our methods balance this trade-off by:

- Tying σ_i to data geometry (Methods 1-3)
- Directly optimizing for reward accuracy (Method 4)

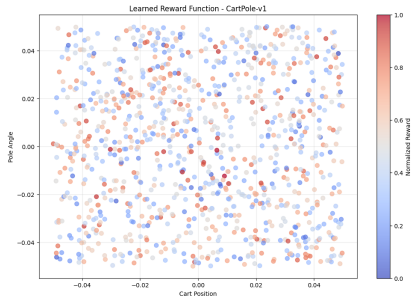

Fig. 1: Learned Reward Progression for CartPole

IV. Pseudocode

V. Experiments and Results

A. Experimental Setup

We evaluated our approach on both `Pendulum-v1` and `CartPole-v1` environments:

- **Pendulum:**
  - State space: $[\cos\theta, \sin\theta, \dot{\theta}]$
  - Expert demonstrations: 200 trajectories from PPO policy
  - Candidate K values: $[2, 3, 4, 5]$
- **CartPole:**
  - State space: [cart position, cart velocity, pole angle, pole angular
  - Expert demonstrations: 10 trajectories of 200 steps each
  - Candidate K values: $[2, 3]$
- **Common Parameters:**
  - Bandit trials: 20 (Pendulum), 5 (CartPole)
  - Learning rate: 0.01
  - Epochs: 15 (Pendulum), 5 (CartPole)

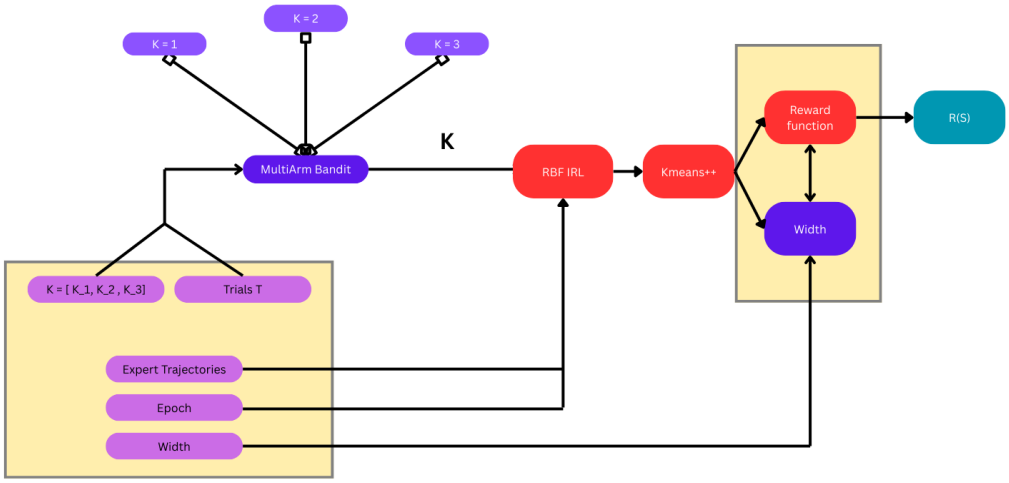

Fig. 2: Architecture Overview.

Algorithm 1 Main RBF IRL Training Procedure

Require: expert_demos, env, $k\_candidates$, width_method, epochs, learning_rate

Step 1: Select # of RBF Centers
if using_k_selection **then**
  $K \leftarrow$ BanditKSelection(expert_demos, $k\_candidates$)
else
  $K \leftarrow$ predefined_value
end if
Step 2: Initialize RBF Network
centers, widths $\leftarrow$ InitializeRBFNetwork(expert_demos, K, width_method)
initialize weight vector $\mathbf{w} \leftarrow \mathbf{0}$
for epoch = 1 **to** epochs **do**
  rollouts $\leftarrow$ GenerateRollouts(centers, widths, $\mathbf{w}$, env)
  $\mu_E \leftarrow$ ComputeFeatureExpectations(expert_demos, centers, widths)
  $\mu_R \leftarrow$ ComputeFeatureExpectations(rollouts, centers, widths)
  gradient $\leftarrow \mu_E - \mu_R$
  $\mathbf{w} \leftarrow \mathbf{w} +$ learning_rate $\times$ gradient
  if width_method == `"learned"` **then**
    widths $\leftarrow$ UpdateWidths(expert_demos, rollouts, centers, widths)
  end if
end for
return centers, widths, $\mathbf{w}$

Algorithm 2 Bandit-Based K Selection

Require: expert_demos, $k\_candidates$, trials

bandit $\leftarrow$ InitializeBandit($k\_candidates$)
for t = 1 **to** trials **do**
  $k \leftarrow$ bandit.select_k() {via UCB}
  labels $\leftarrow$ KMeans(expert_demos, k).labels
  score $\leftarrow$ silhouette_score(expert_demos, labels)
  bandit.update(k, score)
end for
return bandit.get_best_k()

Algorithm 3 RBF Network Initialization

Require: data, K, width_method

centers $\leftarrow$ KMeans(data, K)
define $\phi(s, c, w) = \exp\left(-\|s - c\|^2/(2\,w^2)\right)$
if width_method == `"fixed"` **then**
  widths $\leftarrow$ constant value
else if width_method == `"adaptive"` **then**
  widths $\leftarrow$ ComputeAdaptiveWidths(centers, data)
else if width_method == `"per_cluster"` **then**
  widths $\leftarrow$ median distance to each center
else if width_method == `"learned"` **then**
  widths $\leftarrow$ InitializeThenLearn(centers, data)
end if
return centers, widths

TABLE I: Performance Comparison Across Environments

Environment	Best K	Silhouette Score	Final Loss
Pendulum-v1	3	0.8534	0.2744
CartPole-v1	3	0.3552	0.1022

B. Performance Results

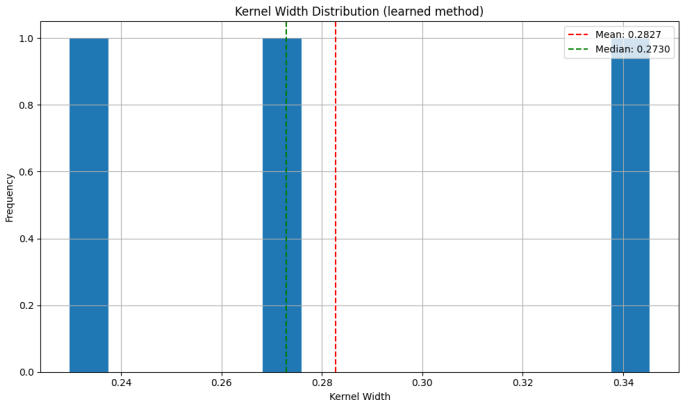

Fig. 3: Kernel Width Distribution for Pendulum

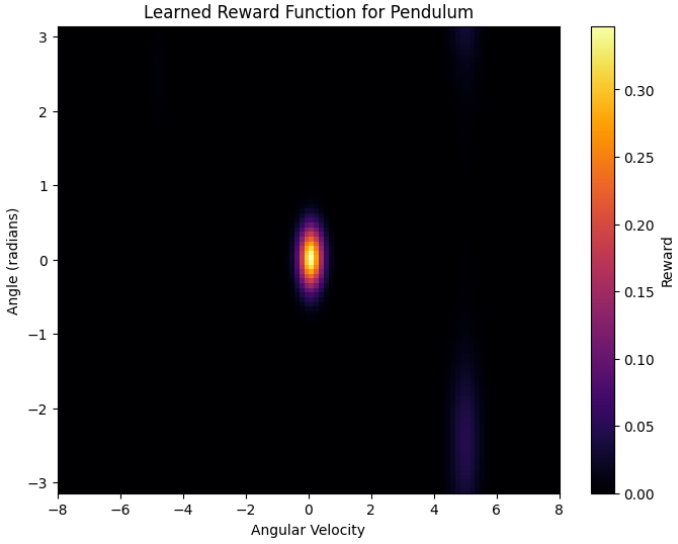

Fig. 4: Learned Reward Progression for Pendulum

C. Key Findings

- Learned width method achieved best performance (silhouette score: 0.8534 for Pendulum).
- Adaptive methods balanced speed and performance.
- IRL policy improved PPO baseline from -191 to -174 in Pendulum.
- Successfully matched expert performance in CartPole despite suboptimal expert.
- Bandit selection converged quickly to optimal K value.

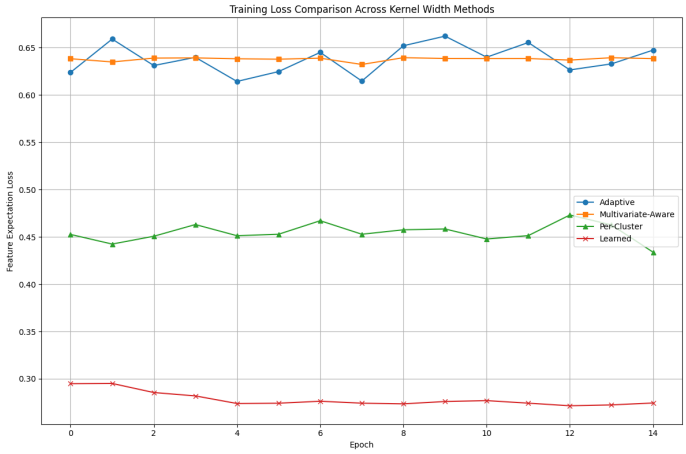

Fig. 5: Training Loss Comparison for Pendulum

D. Limitations

While our method advances feature adaptation in IRL, several important limitations remain:

- **Expert Demonstration Quality**:
  - Fundamental performance ceiling: Cannot outperform suboptimal experts
  - Sensitivity to demonstration noise (e.g., sensor errors in robotics)
- **High-Dimensional State Spaces**:
  - Curse of dimensionality: RBF coverage becomes sparse in $\mathbb{R}^d$ where $d > 10$
  - Covariance computation grows as $O(d^2)$
- **Parameter Sensitivity**:
  - Kernel width scale factor requires environment-specific tuning
  - Bandit exploration parameters affect center selection stability

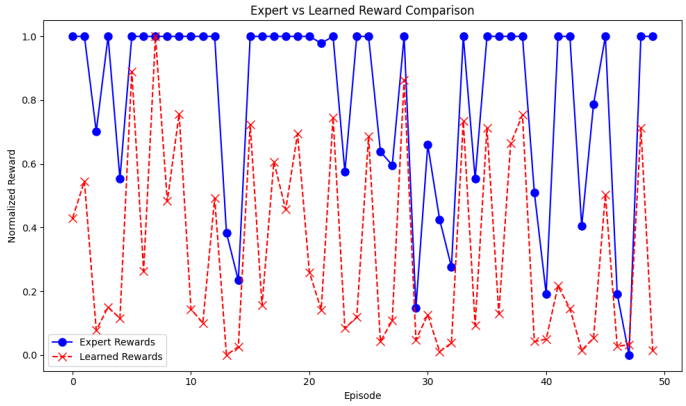

Fig. 6: Comparison of expert versus learned reward functions in CartPole, illustrating both successful recovery and limitations from noisy demonstrations.

VI. Conclusion

We presented an IRL framework combining RBF networks with adaptive kernel width tuning and bandit-based center selection. The method demonstrated strong performance across different environments while maintaining computational efficiency. Future work will extend this approach to high-dimensional continuous control tasks and investigate theoretical convergence guarantees.

VII. Individual Contributions

A. Ashiq (CS22B2021)

- **RBF Network Implementation & Feature Engineering**
  - Proposed using **Radial Basis Functions (RBFs)** instead of polynomial features for better non-linear reward approximation
  - Suggested the mathematical formulation for the RBF-based reward function:

$$R(s) = \sum_{i=1}^{K} w_i \exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right)$$

  - Implemented the RBF network initialization (Algorithm 3) and integrated it with the IRL pipeline
  - Tested different RBF configurations on `Pendulum-v1` and `CartPole-v1`, showing improvements over polynomial baselines
  - Worked on PPT preparation

B. Ajmal (CS22B2046)

- **Multi-Armed Bandit for Adaptive Center Selection**
  - Researched and implemented a **UCB-based bandit algorithm** (Algorithm 2) to automatically select the best number of RBF centers (K)
  - Used **K-means clustering** to generate candidate K values and evaluated them using **silhouette scores**
  - Optimized exploration vs. exploitation to quickly converge to the best K (e.g., $K = 3$ for Pendulum, $K = 2$ for CartPole)
  - Demonstrated that the bandit approach reduces manual tuning while maintaining performance
  - Worked on report preparation

C. Abishek (CS22B2054)

- **Kernel Width Optimization & Adaptation Strategies**
  - Designed **four kernel width adaptation methods** to improve reward learning:
    1) **Density-Adaptive** - Adjusts σ_i based on local data density
    2) **Cluster-Adaptive** - Sets σ_i to median cluster distance
    3) **Covariance-aware** - Uses elliptical kernels for anisotropic data
    4) **Learned** - Optimizes σ_i directly
  - Showed that **learned widths** performed best (silhouette score: **0.8534** in Pendulum)
  - Analyzed cases where fixed widths failed (e.g., over-smoothing in CartPole) and proposed solutions
  - Worked on video preparation

References

[1] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of ICML*, 2000.

[2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of ICML*, 2004.

[3] D. Baimukashev, G. Alcan, and V. Kyrki, "Automated feature selection for inverse reinforcement learning," *arXiv preprint arXiv:2403.15079*, 2024. [Online]. Available: https://arxiv.org/abs/2403.15079