Probabilistic Machine Learning with Julia

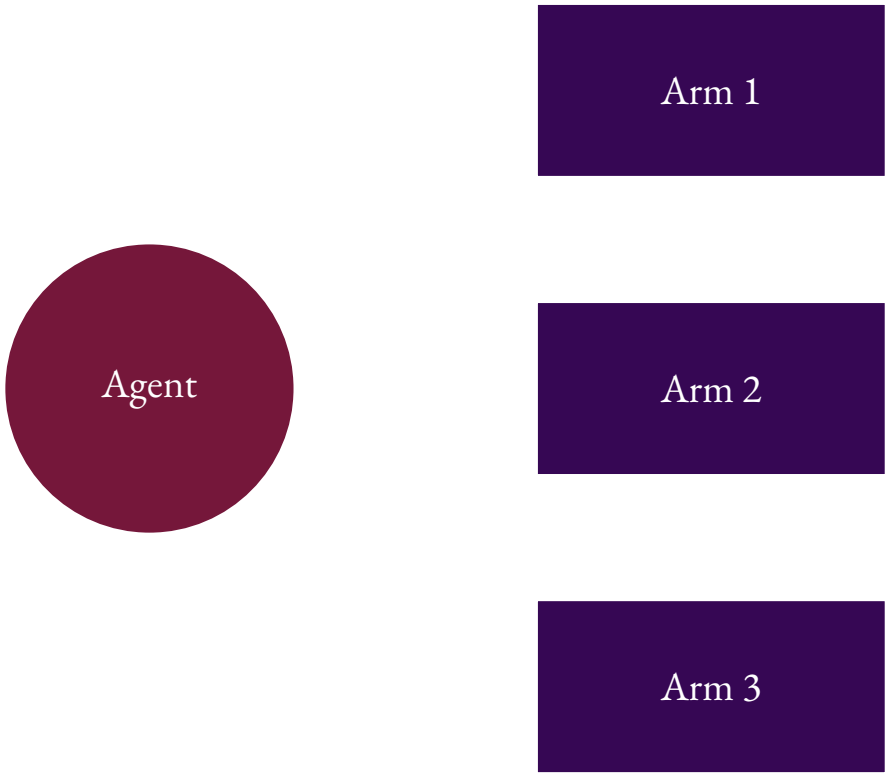
Lecture 7 | Decision-Making with Probabilistic Models

Amirabbas Asadi

amir.asadi78@sharif.edu

2025

Bandit Problems

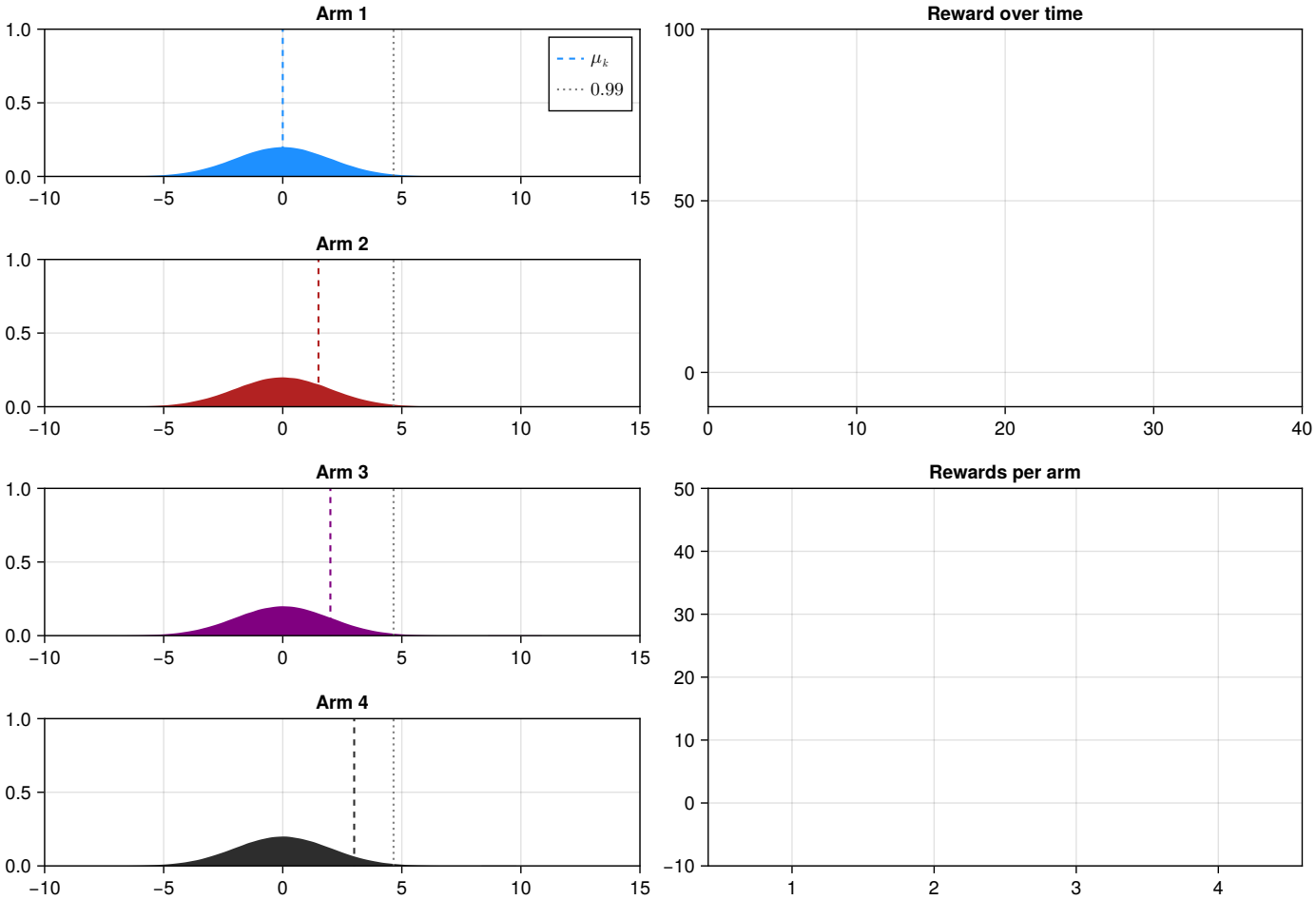


Basic Policies

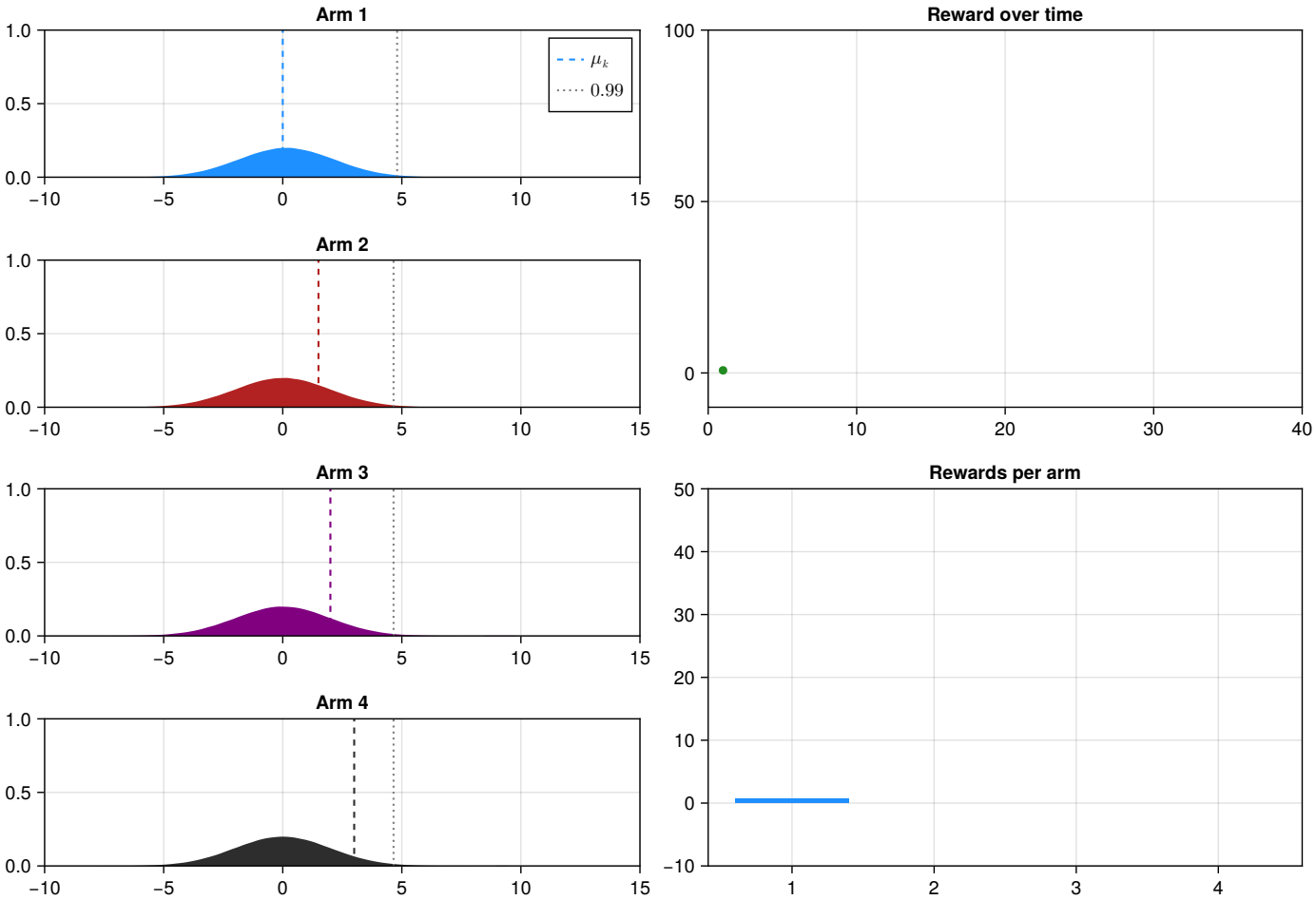
We will discuss some basic policies for this problem:

- Uniform policy
- Explore-then-commit
- ϵ-greedy policy
- Upper Confidence Bound (UCB) policy

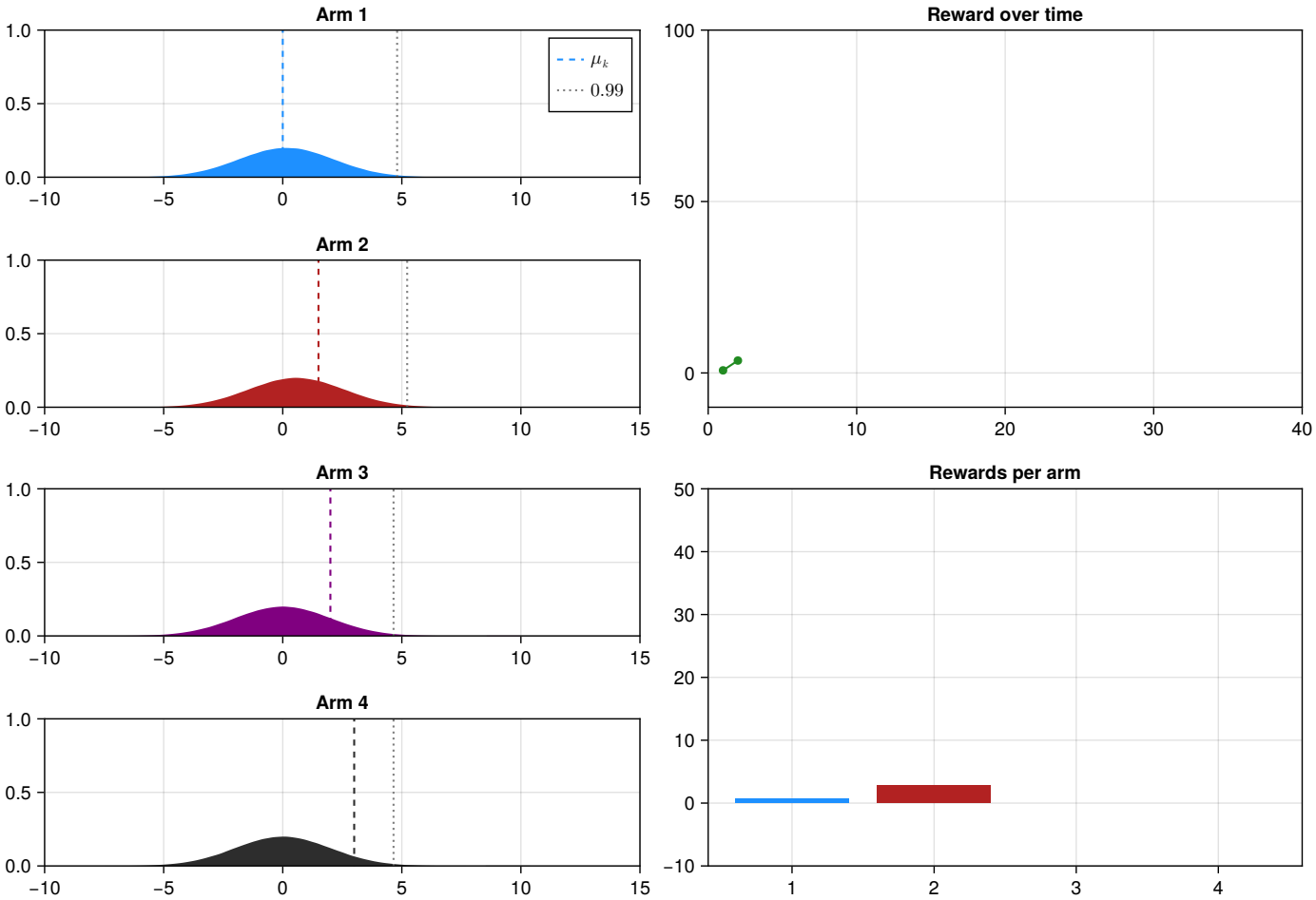
Upper Confidence Bound



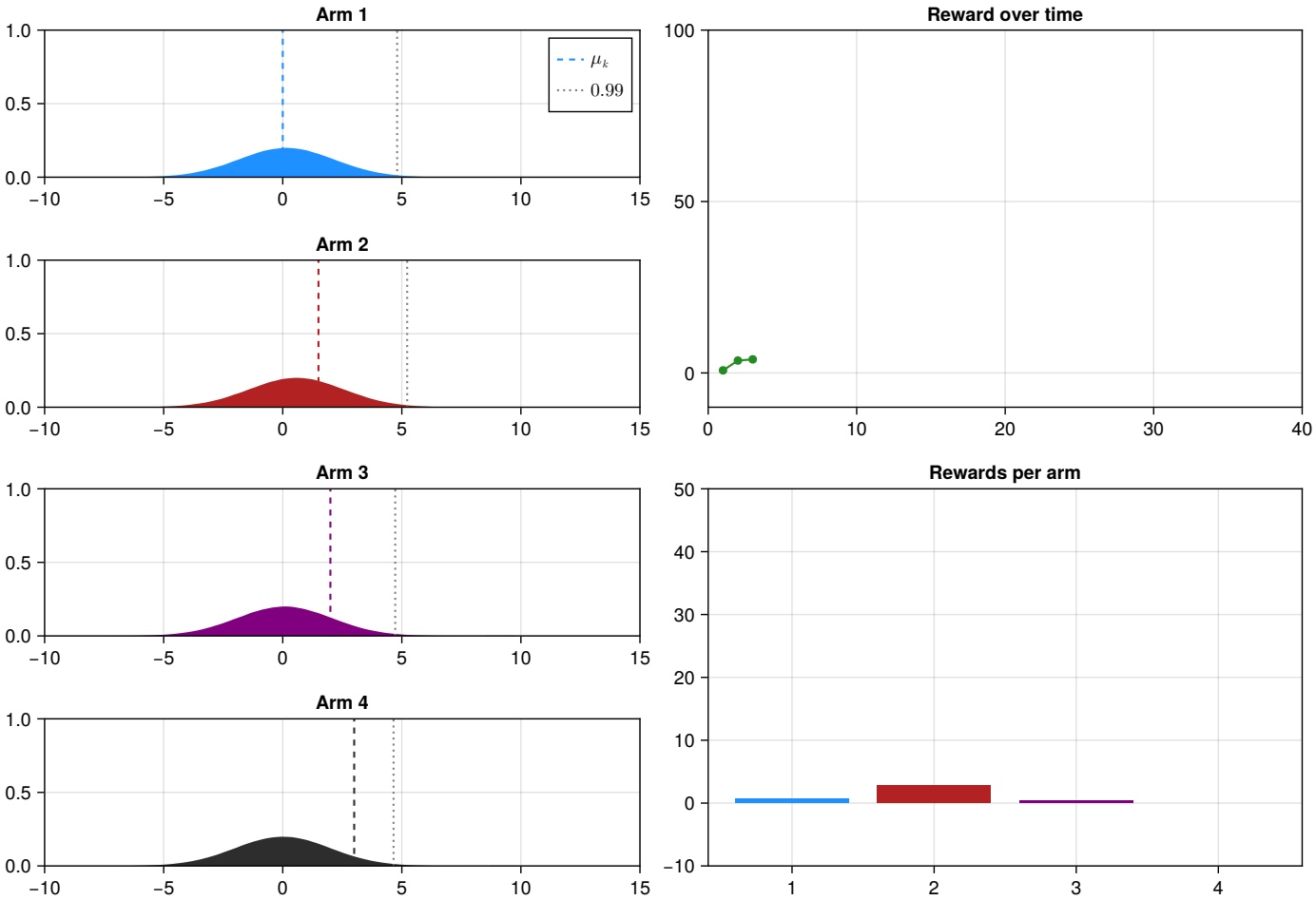
Upper Confidence Bound



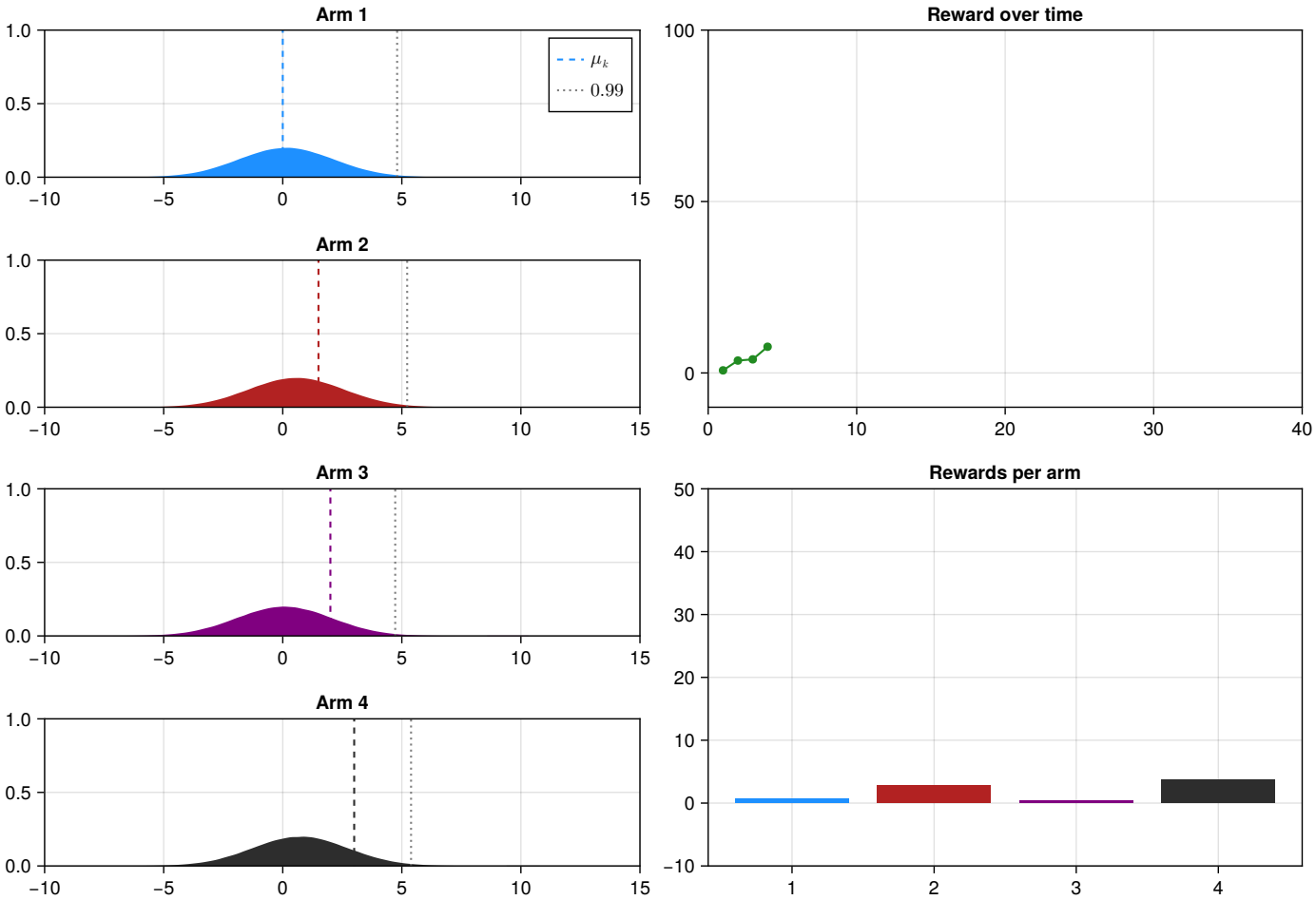
Upper Confidence Bound



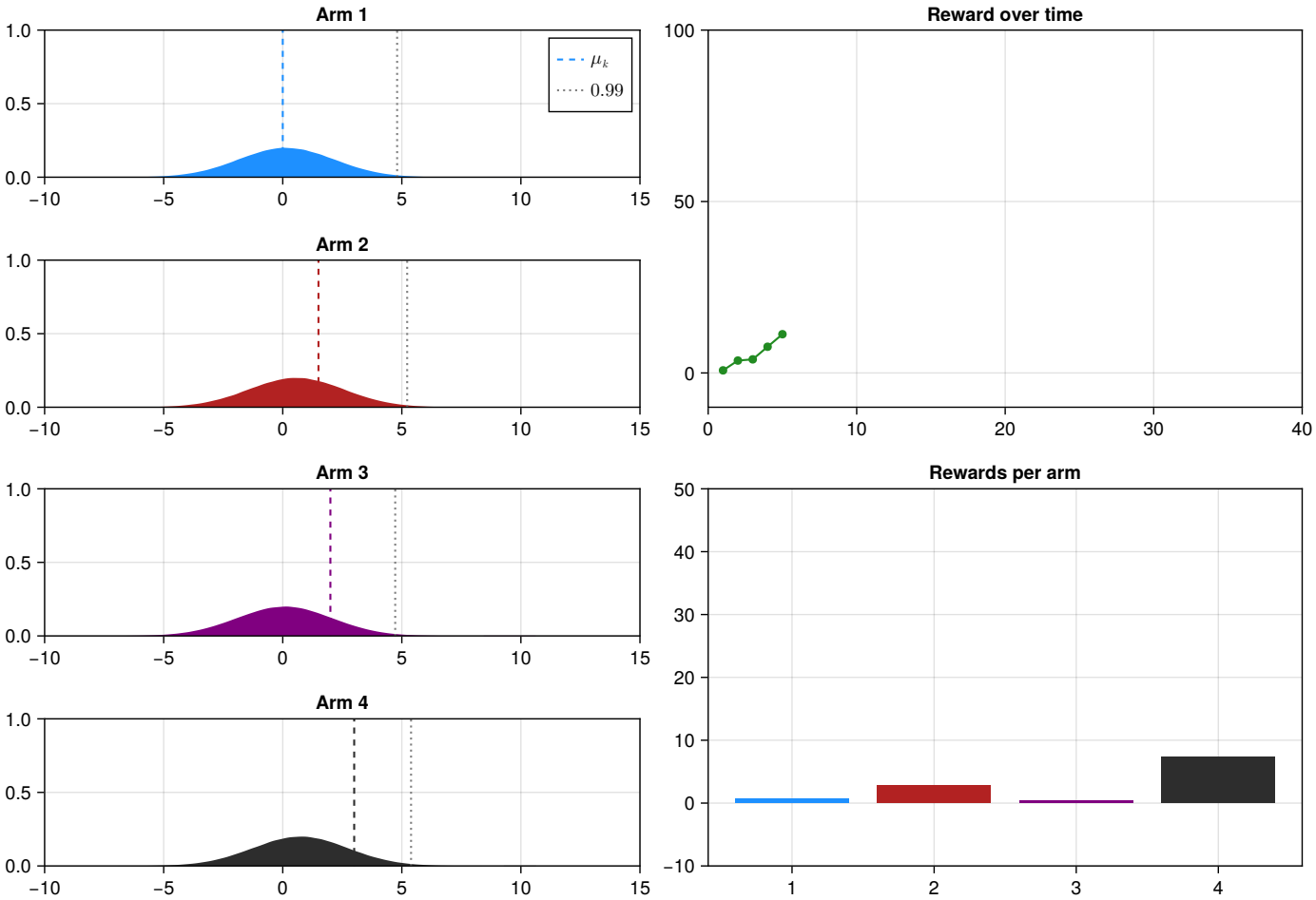
Upper Confidence Bound



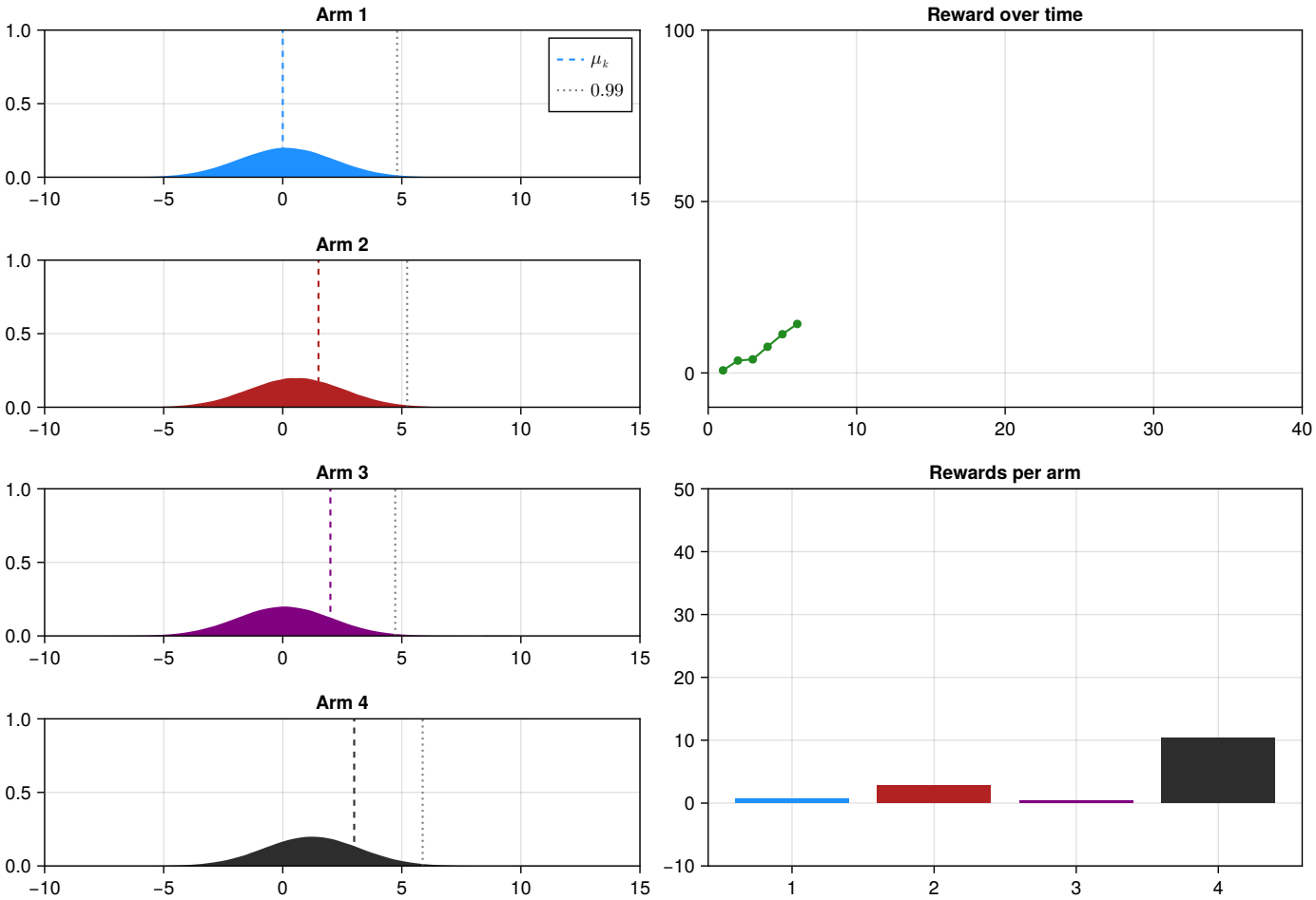
Upper Confidence Bound



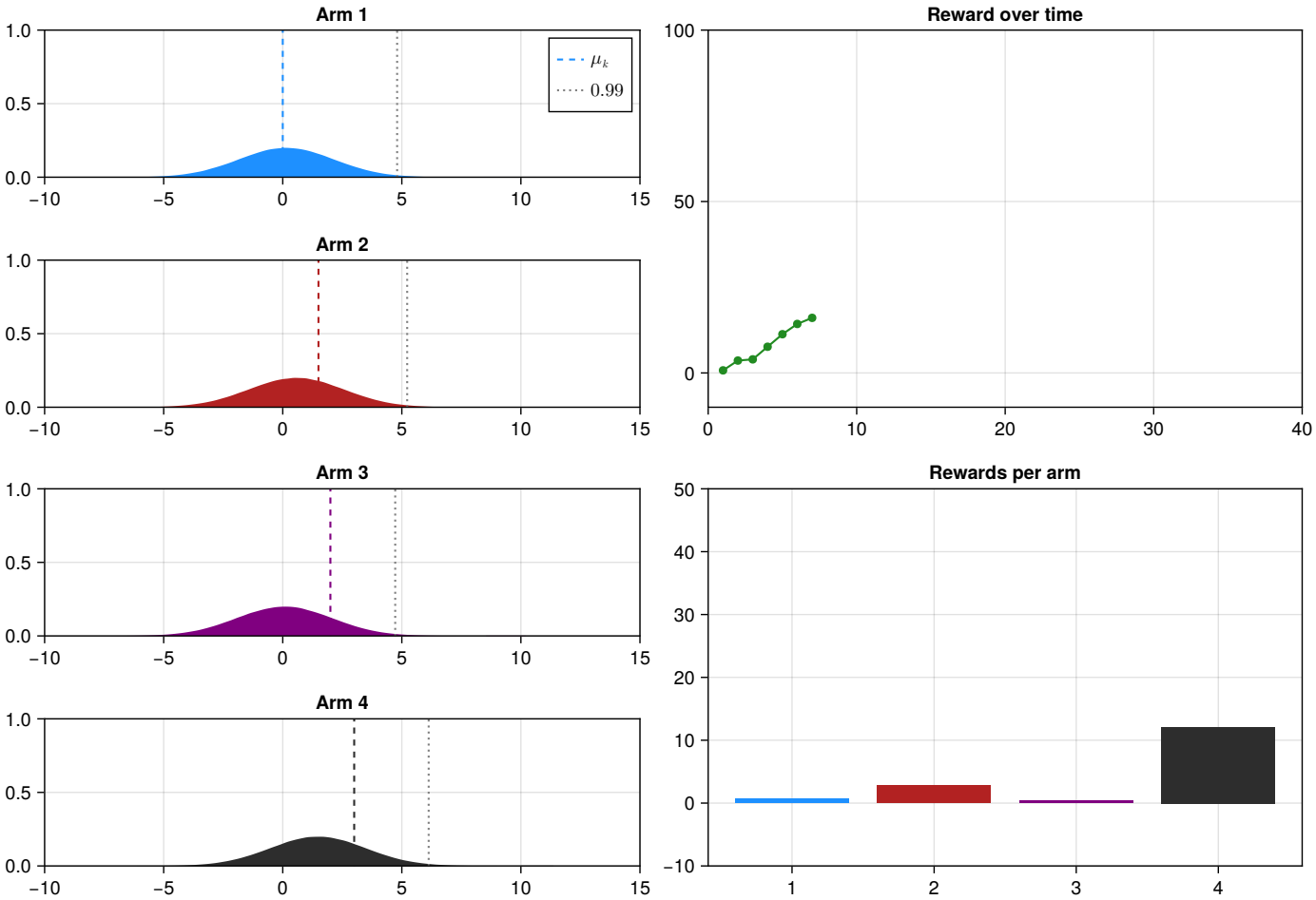
Upper Confidence Bound



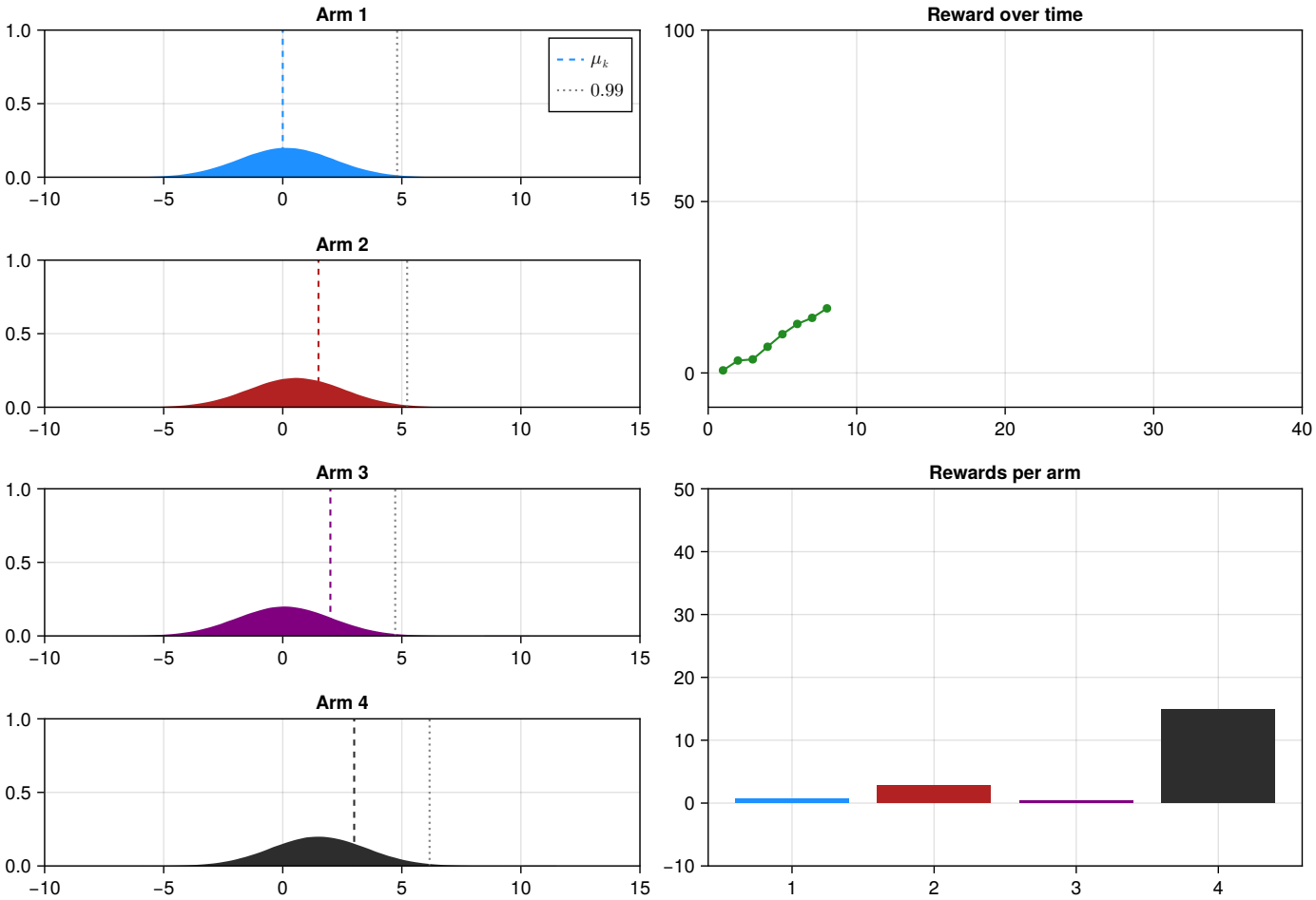
Upper Confidence Bound



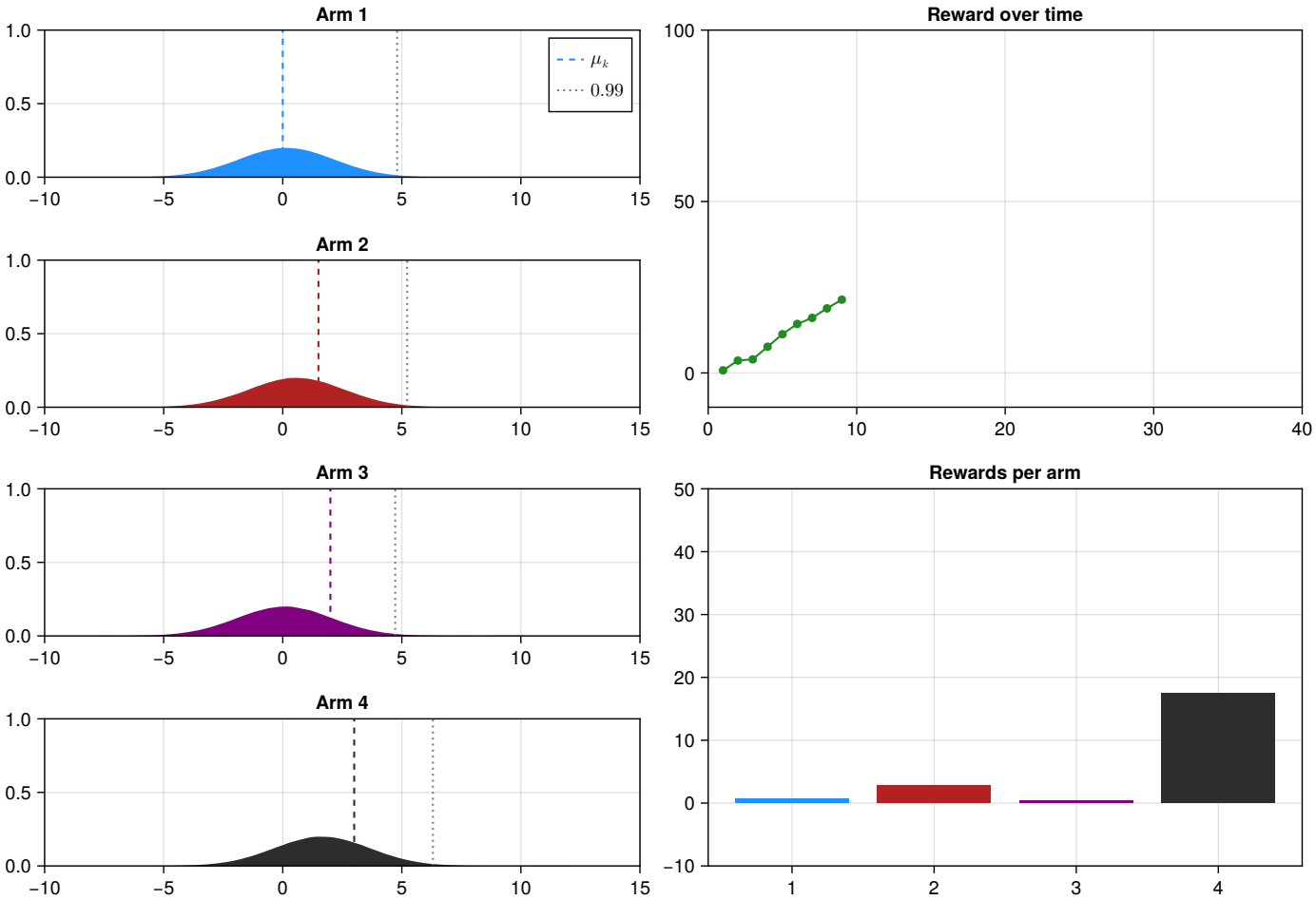
Upper Confidence Bound



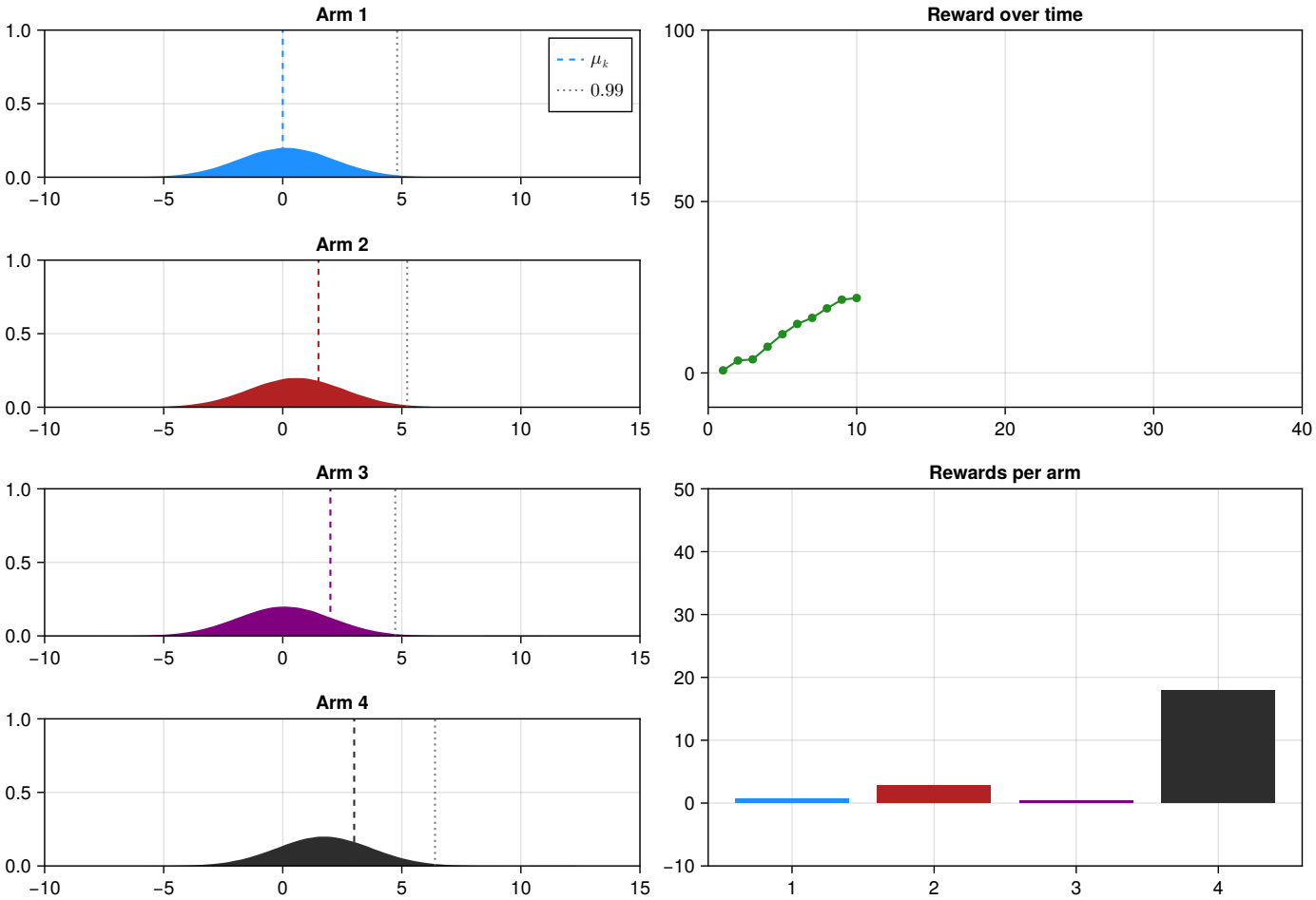
Upper Confidence Bound



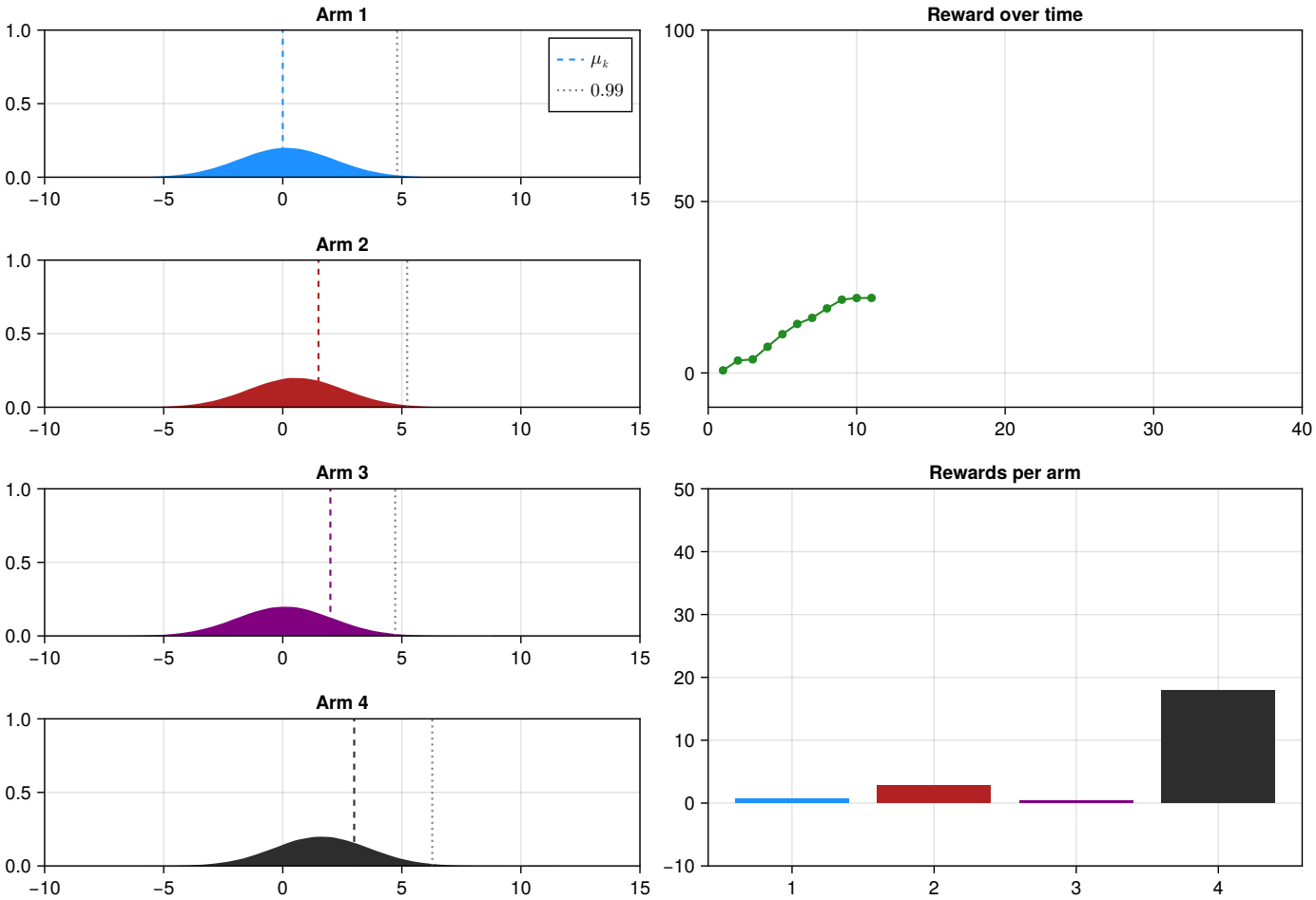
Upper Confidence Bound



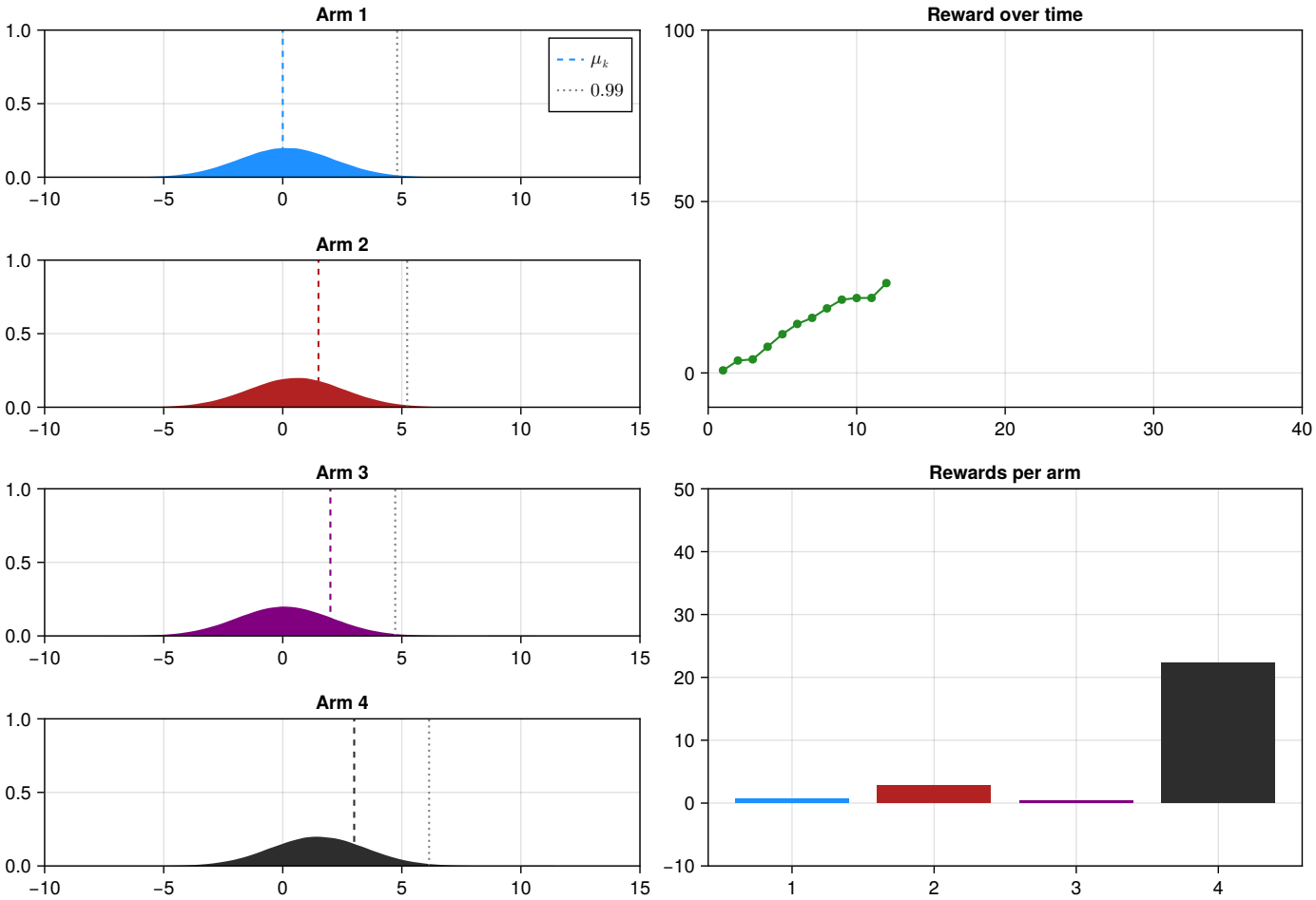
Upper Confidence Bound



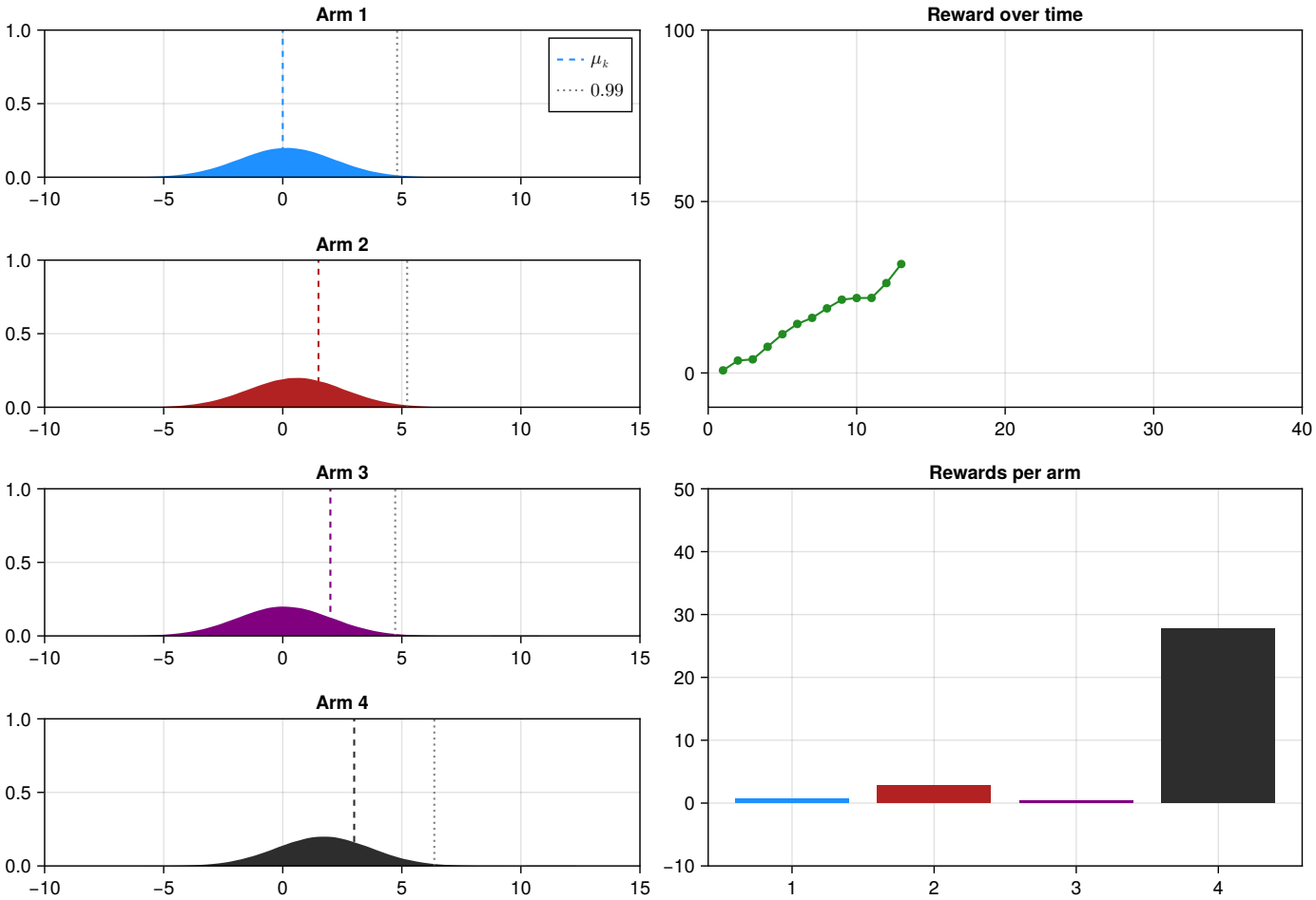
Upper Confidence Bound



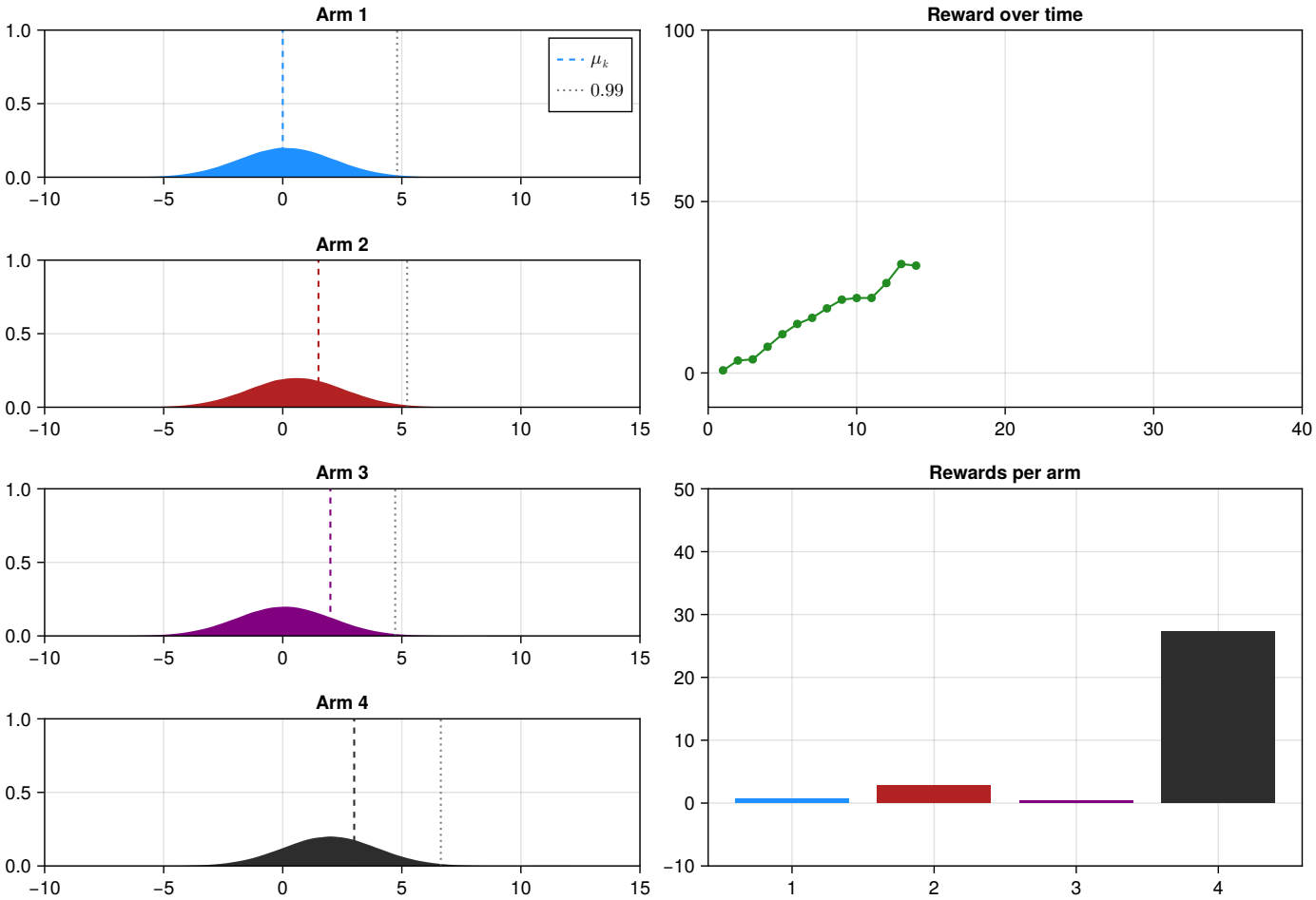
Upper Confidence Bound



Upper Confidence Bound



Upper Confidence Bound



Example: Bayesian Optimization

In the standard multi-armed bandit, the arms are independent.
In many problems by choosing an arm we can gain information about the others.
We will see two such examples:

- Blackbox optimization : Bayesian Optimization
- Collaborative filtering : Probabilistic Matrix Factorization

Example: Bayesian Optimization

Consider the problem of maximizing a function $f : \mathbb{R}^n \to \mathbb{R}$ such that:

- f is a blackbox function
- f is expensive to call

Example: Bayesian Optimization

Consider the problem of maximizing a function $f : \mathbb{R}^n \to \mathbb{R}$ such that:

- f is a blackbox function
- f is expensive to call

Can we formulate this as a bandit problem with infinite arms and apply something like UCB?

Example: Bayesian Optimization

Consider the problem of maximizing a function $f : \mathbb{R}^n \to \mathbb{R}$ such that:

- f is a blackbox function
- f is expensive to call

Can we formulate this as a bandit problem with infinite arms and apply something like UCB?

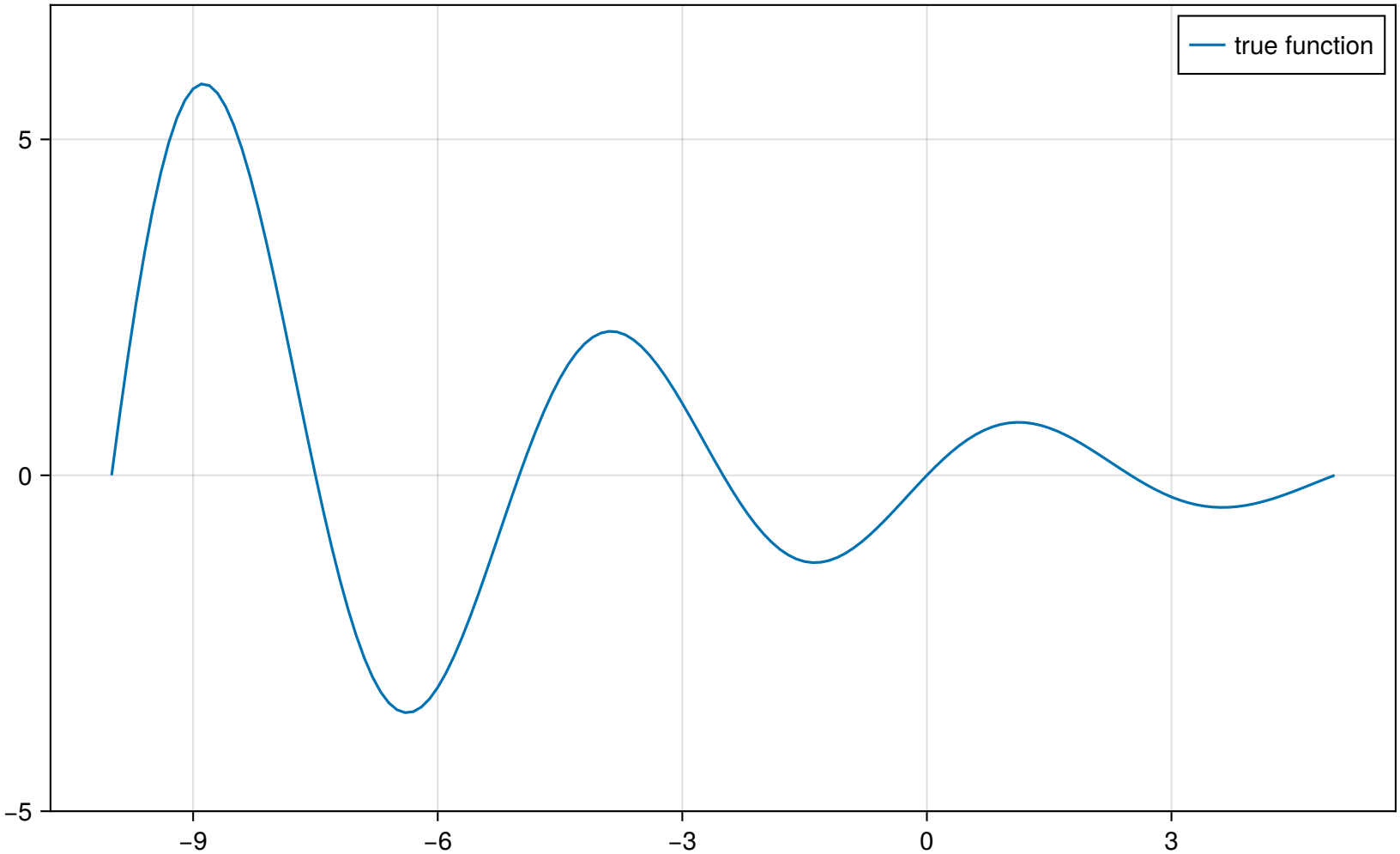
The are two main approaches to define probabilistic models with infinite latent variables:

- Nonparametric modeling
- Amortized inference

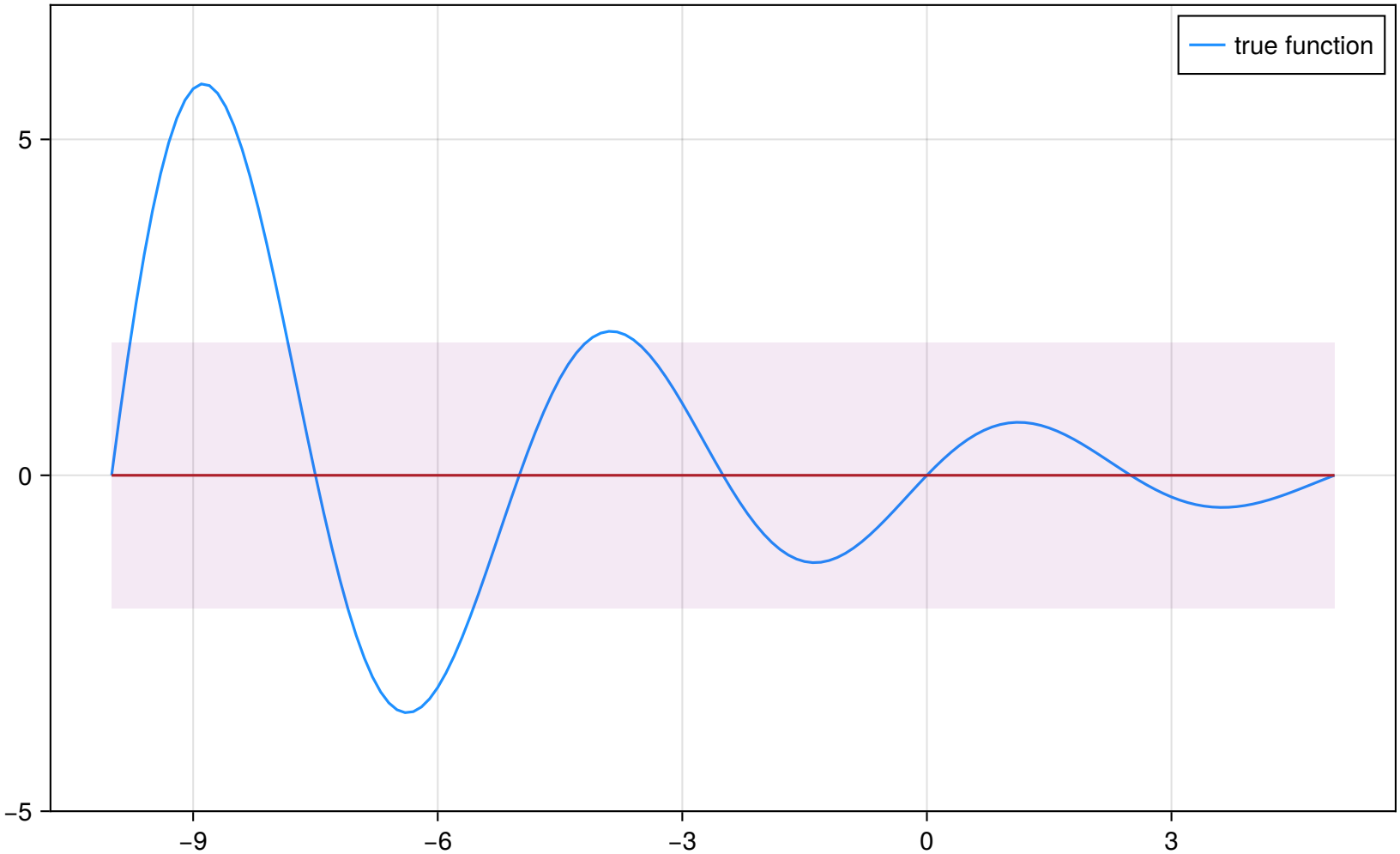
Example: Bayesian Optimization

We saw Gaussian Process in the previous lectures, so let's use it with a UCB policy

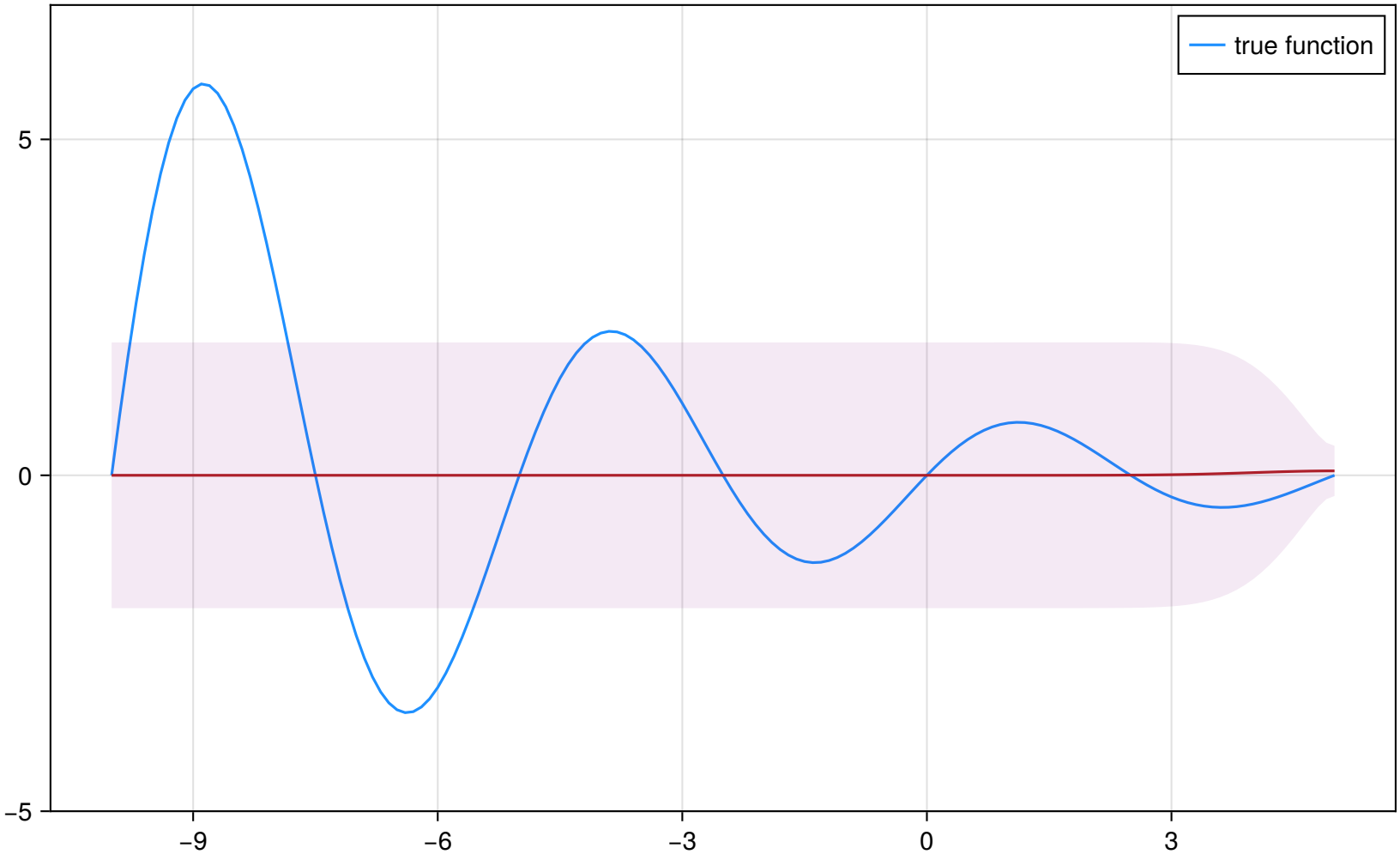
Example: Bayesian Optimization



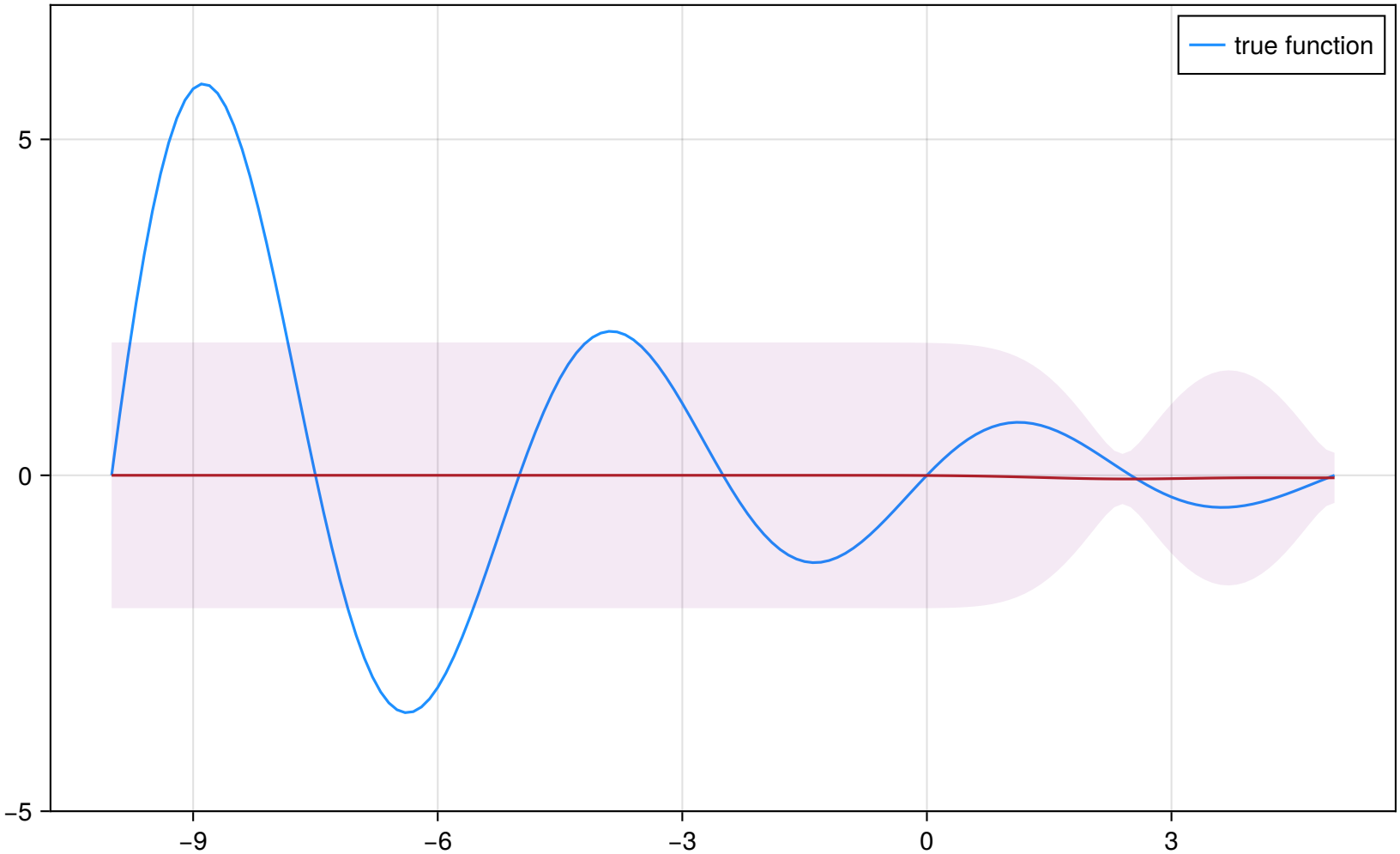
Example: Bayesian Optimization



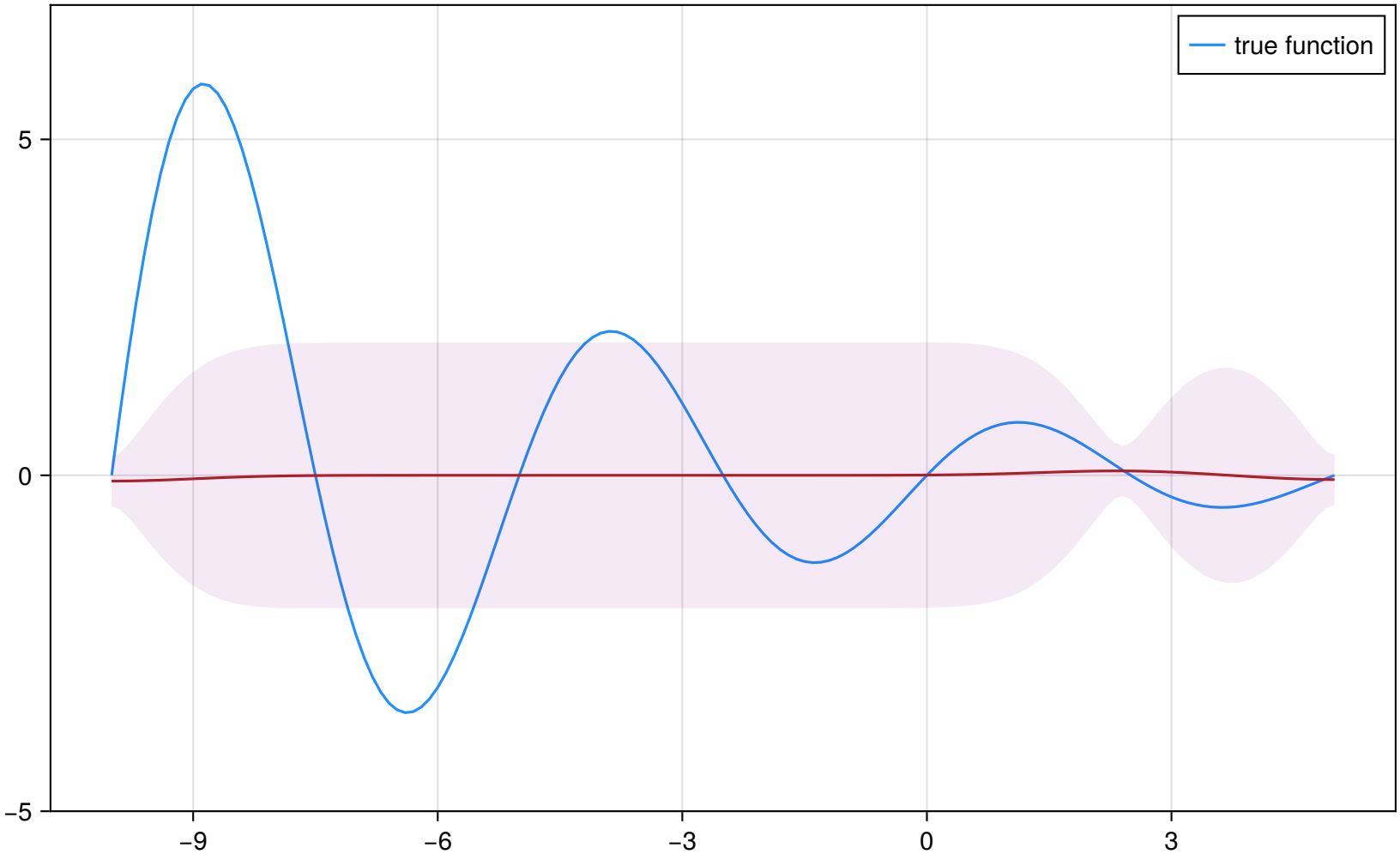
Example: Bayesian Optimization



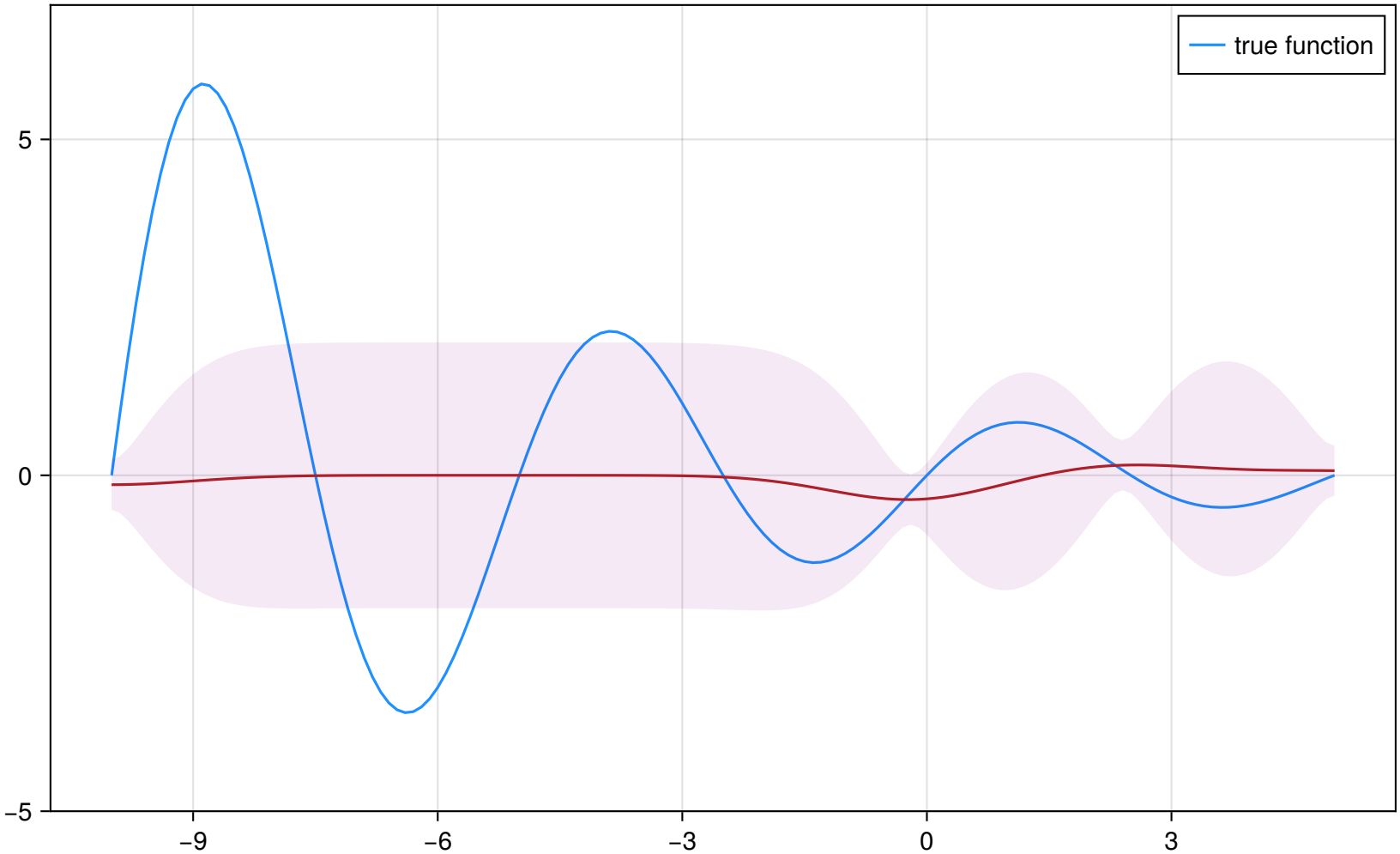
Example: Bayesian Optimization



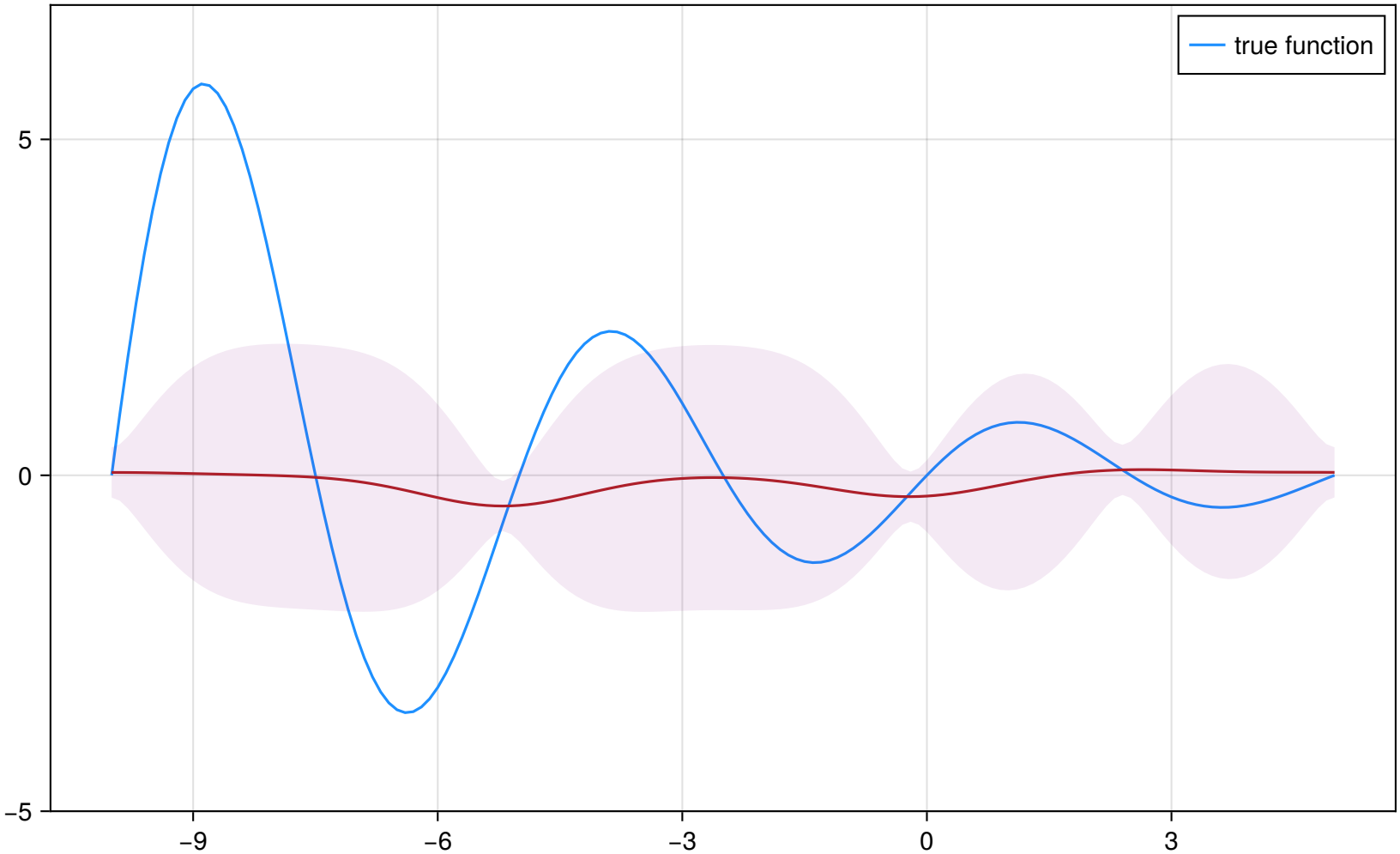
Example: Bayesian Optimization



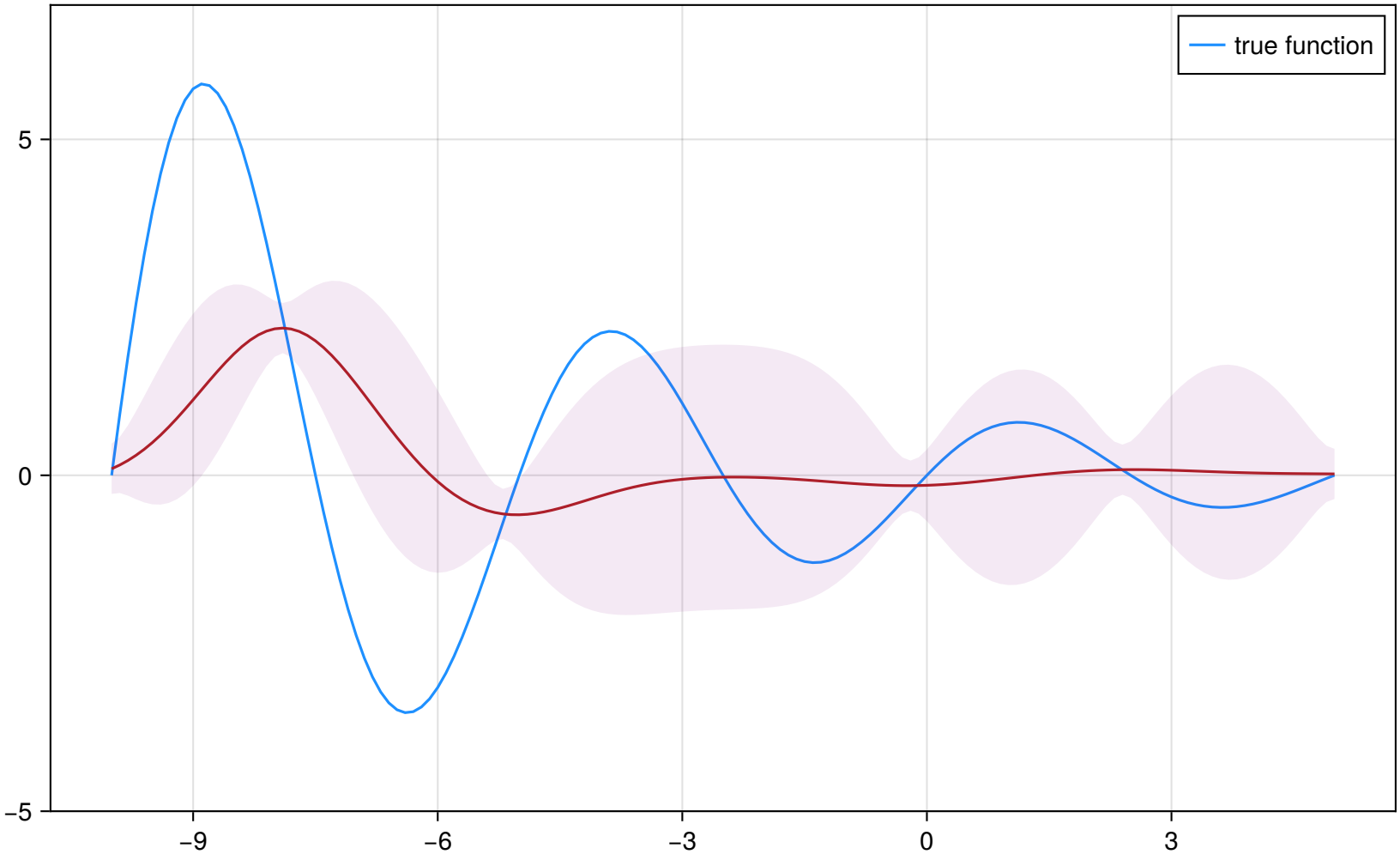
Example: Bayesian Optimization



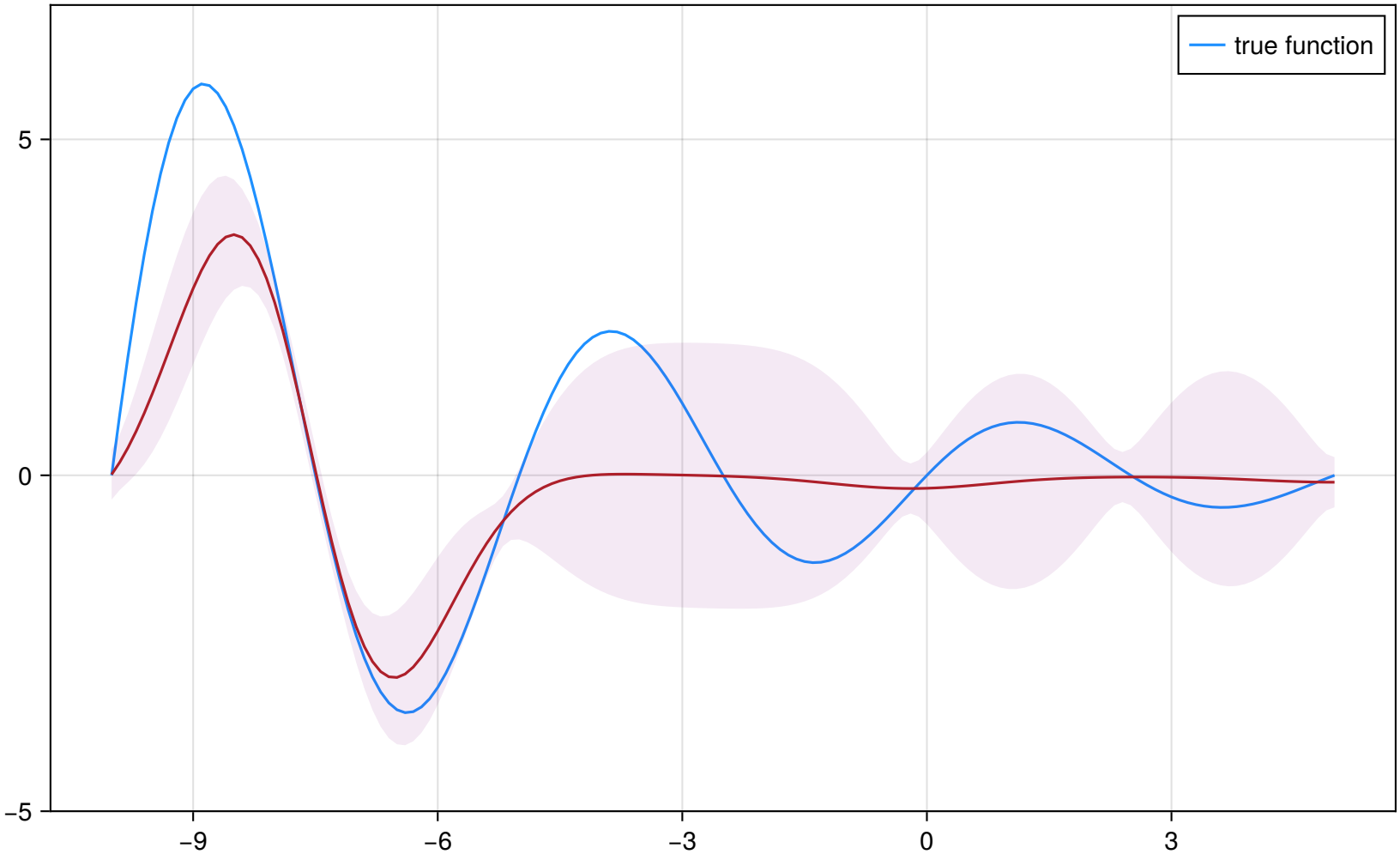
Example: Bayesian Optimization



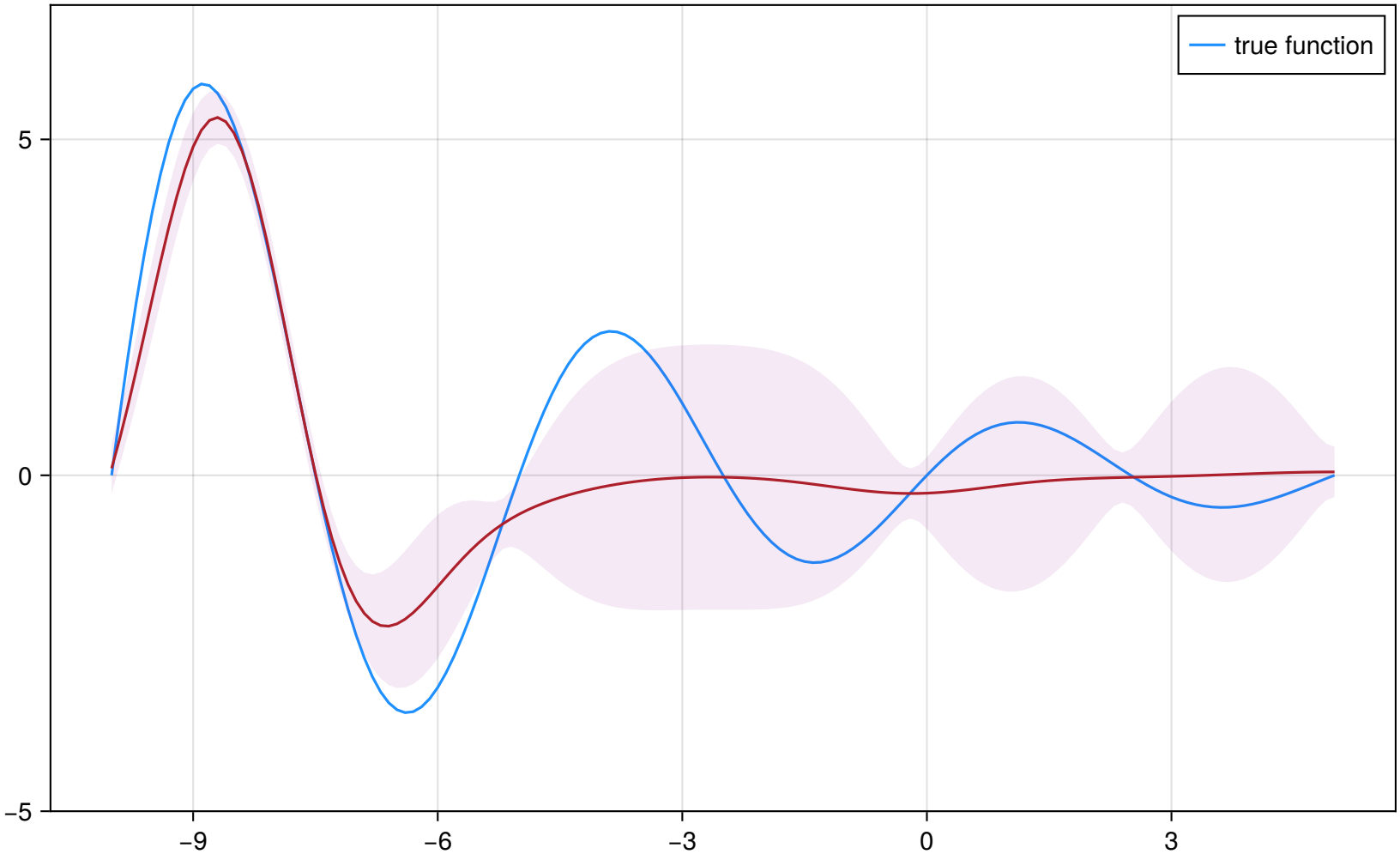
Example: Bayesian Optimization



Example: Bayesian Optimization



Example: Bayesian Optimization



Example: Recommendation with Collaborative Filtering

Another example of structured bandit problems arises in recommender systems (movies, music, ...). Observing a user's preferences can give us information about other similar users likewise about movies.

To apply a policy like UCB we need a probabilistic approach to collaborative filtering.

Probabilistic Matrix Factorization

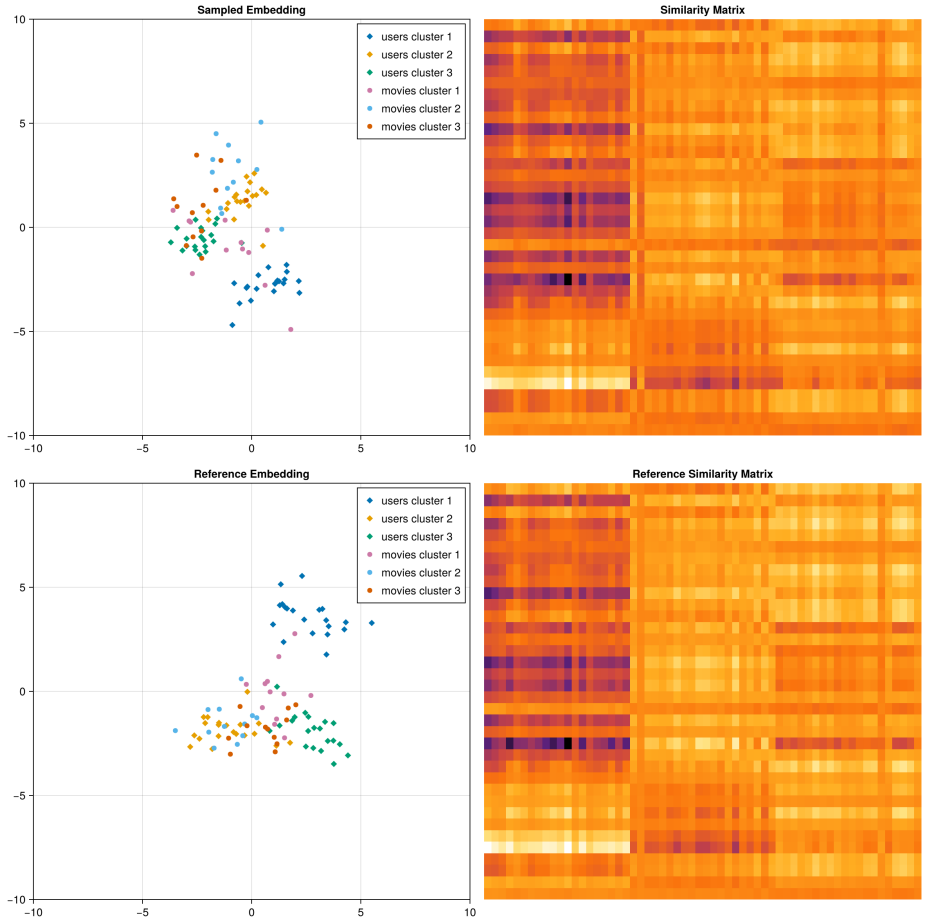
$$u_i \sim \text{MvNormal}(0, I)$$

$$m_j \sim \text{MvNormal}(0, I)$$

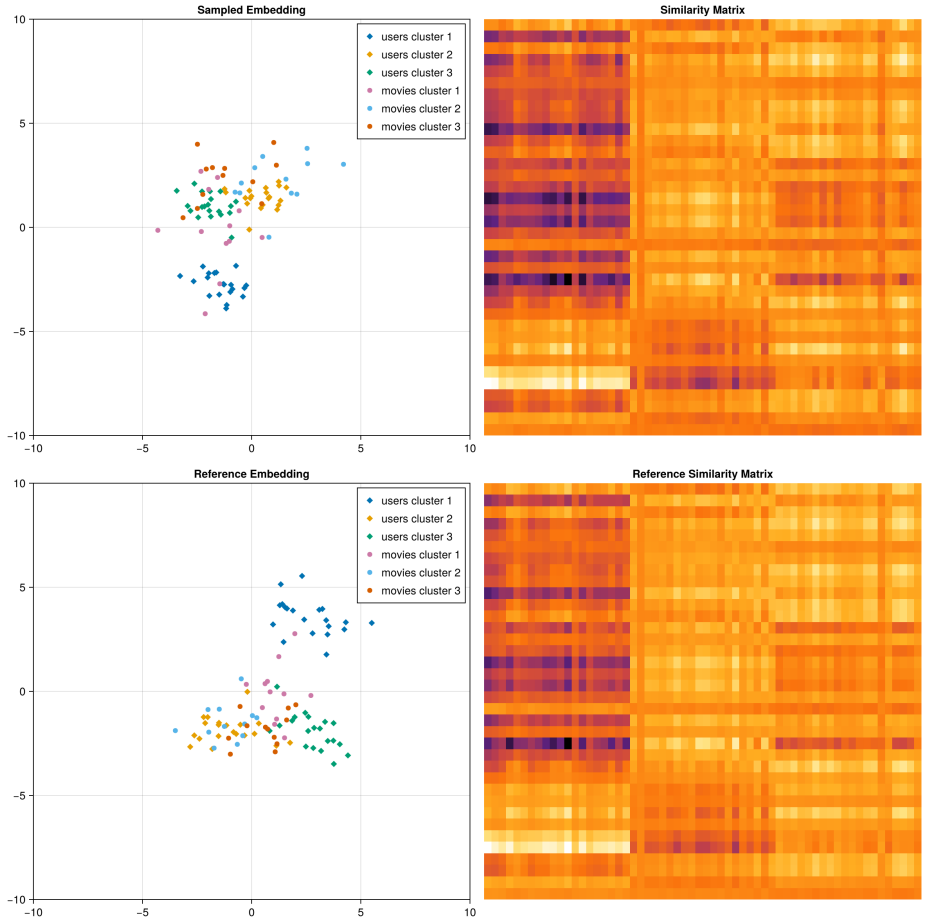
$$p_{ij} = \sigma(u_i . m_j)$$

$$x_{ij} \sim \text{Bernoulli}(p_{ij})$$

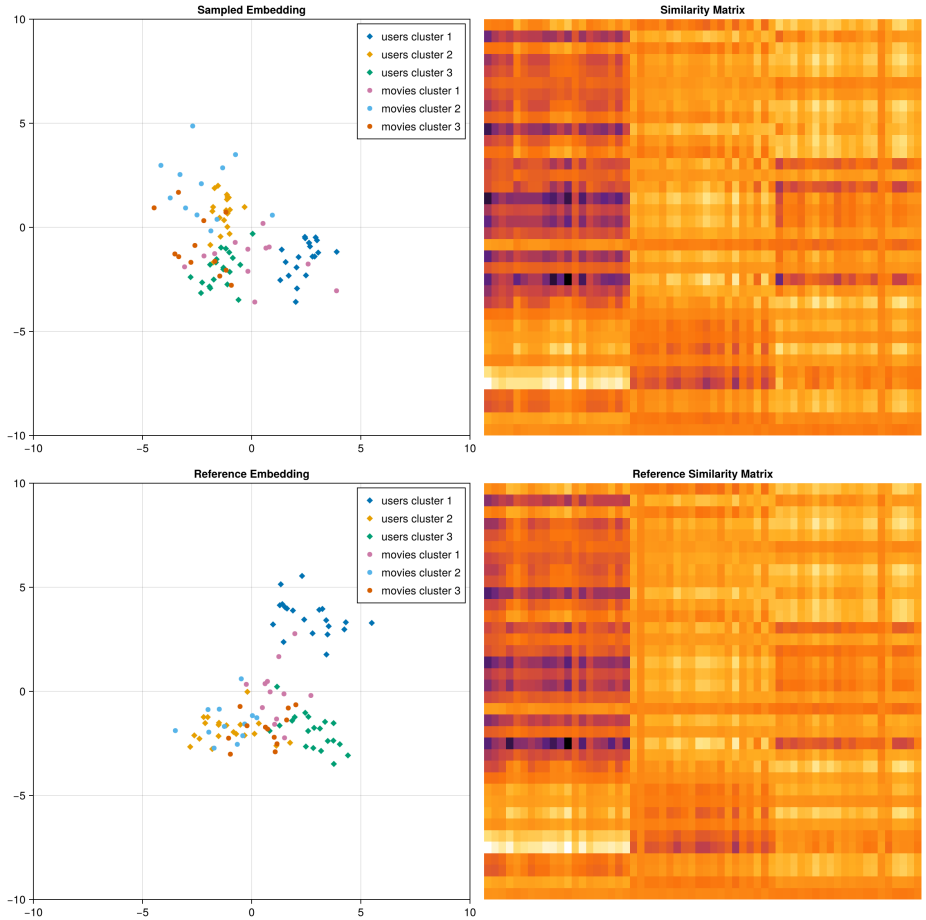
Probabilistic Matrix Factorization



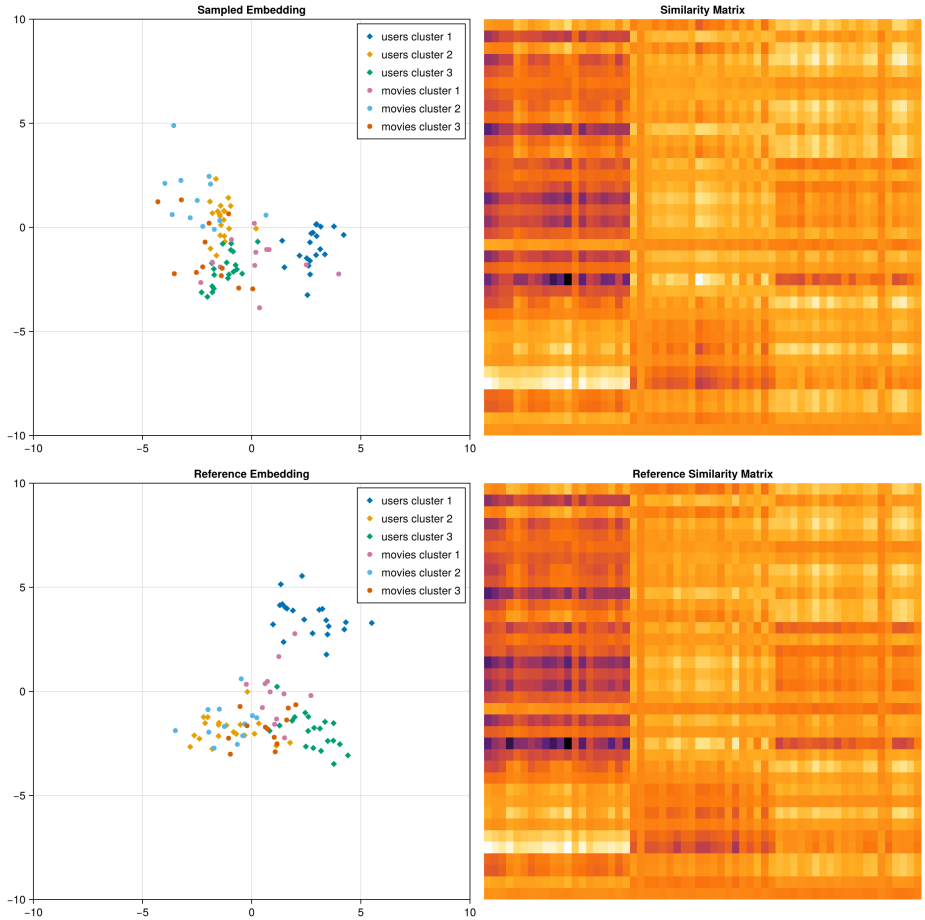
Probabilistic Matrix Factorization



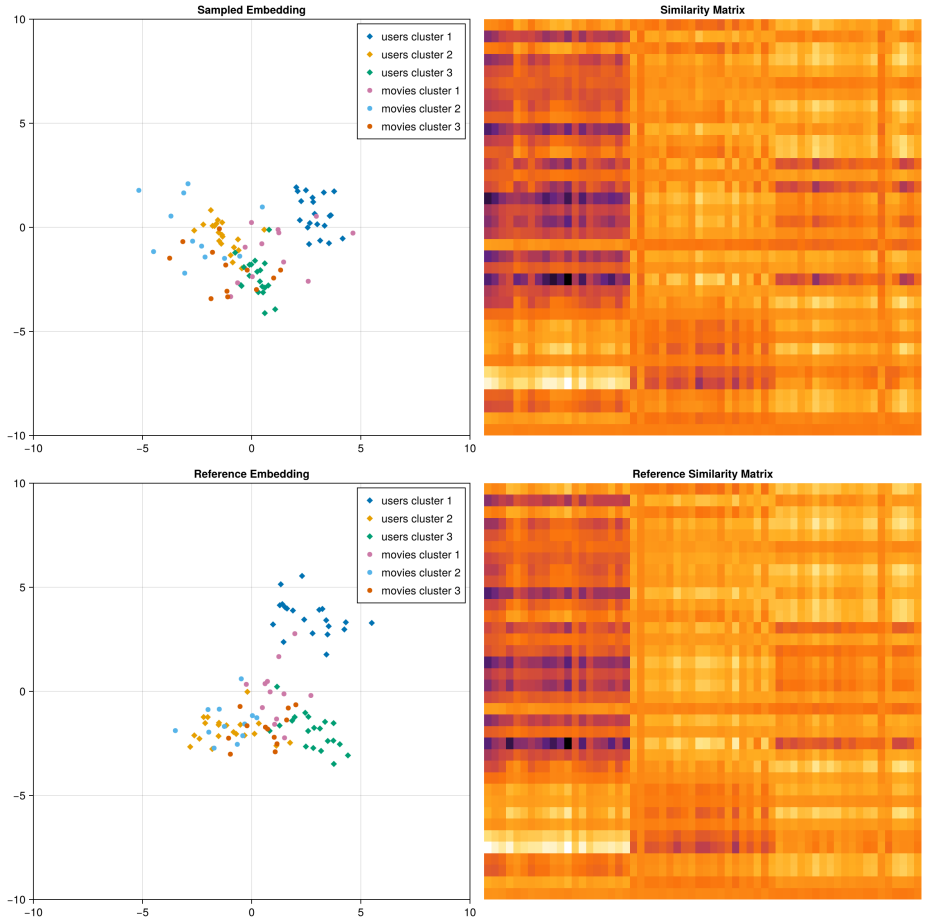
Probabilistic Matrix Factorization



Probabilistic Matrix Factorization



Probabilistic Matrix Factorization

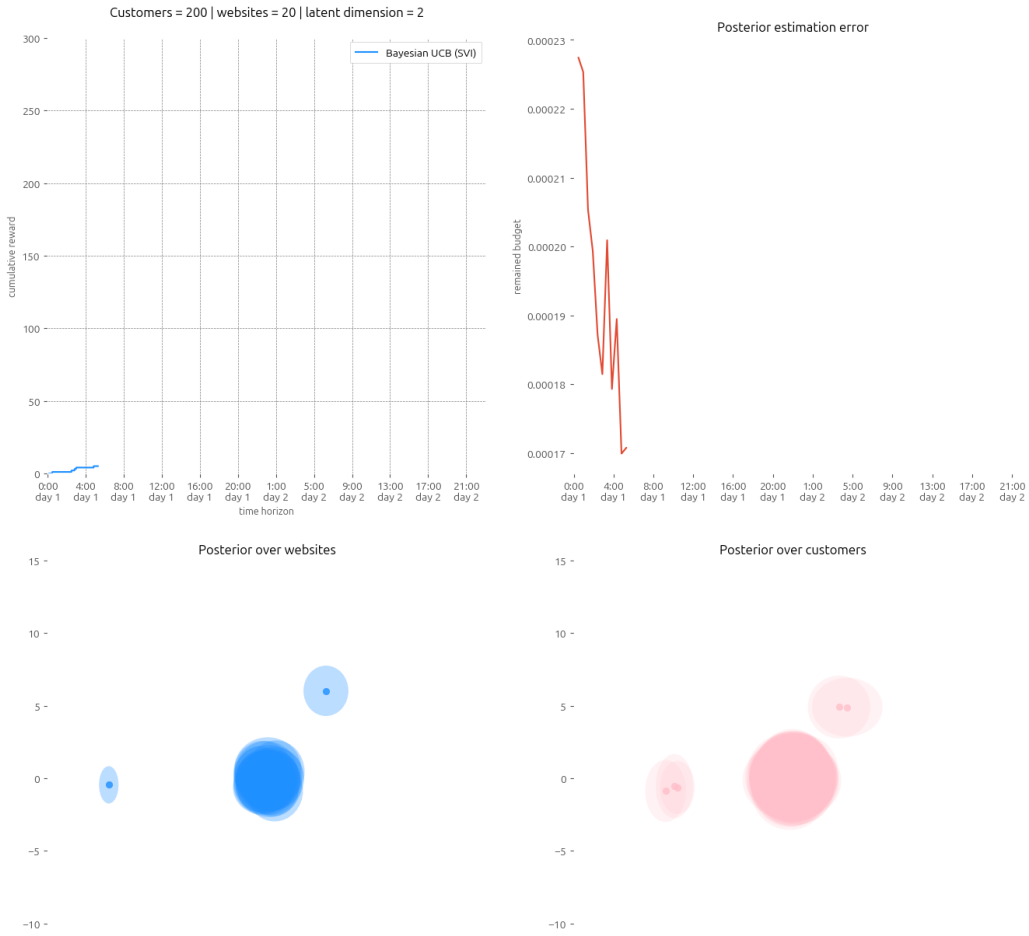


Probabilistic Matrix Factorization

Application: Ad click prediction and recommendation with Bayesian Upper Confidence Bound

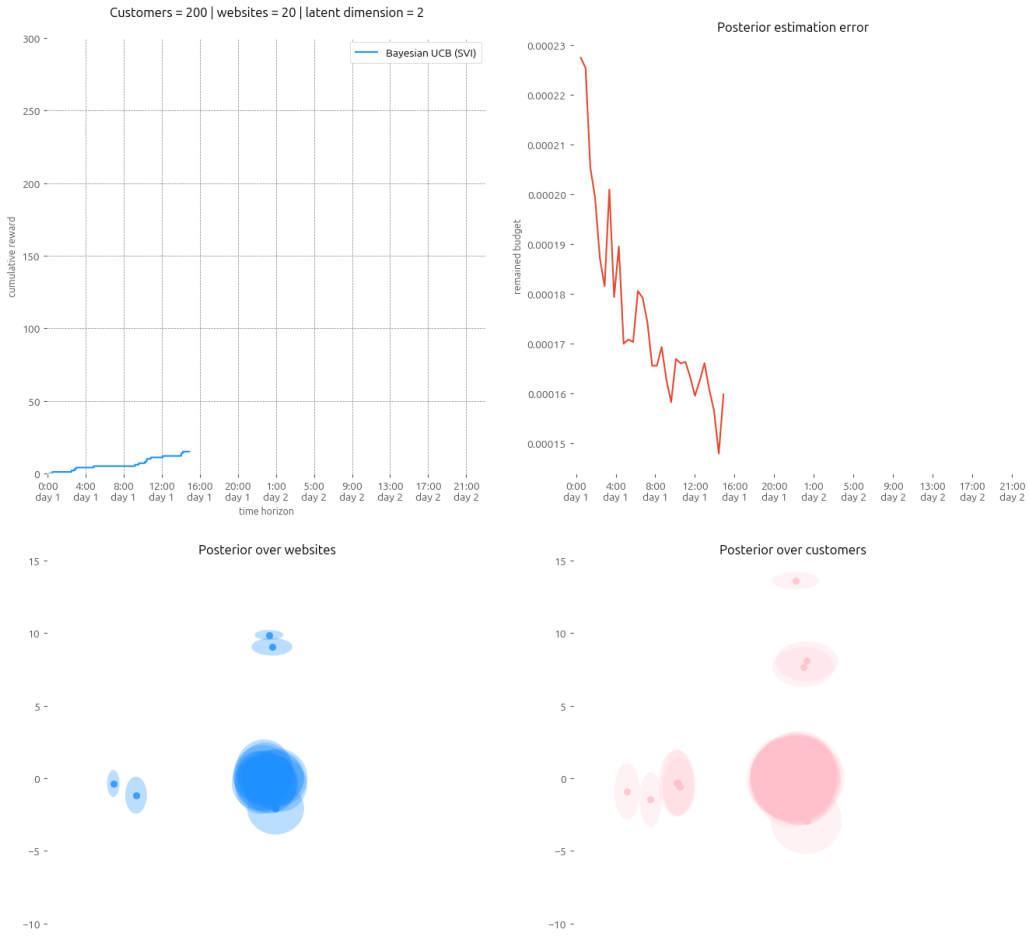
Probabilistic Matrix Factorization

Application: Ad click prediction and recommendation with Bayesian Upper Confidence Bound



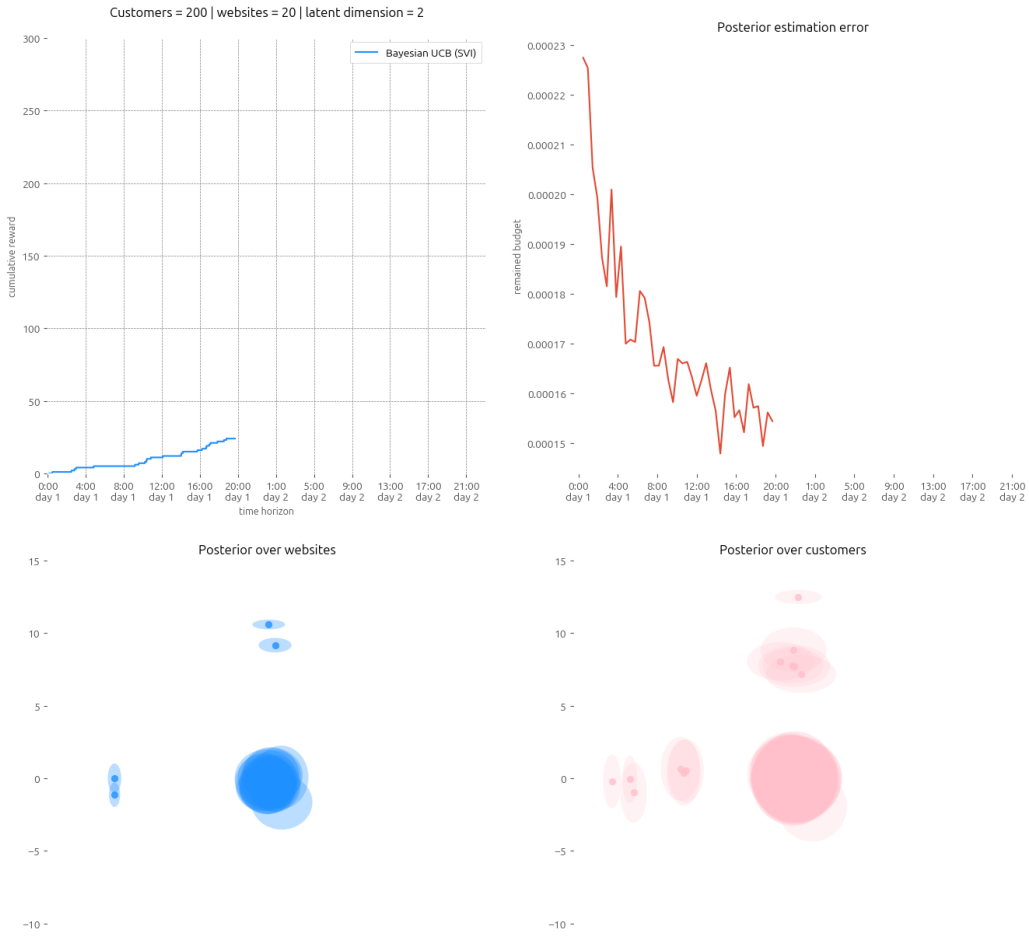
Probabilistic Matrix Factorization

Application: Ad click prediction and recommendation with Bayesian Upper Confidence Bound



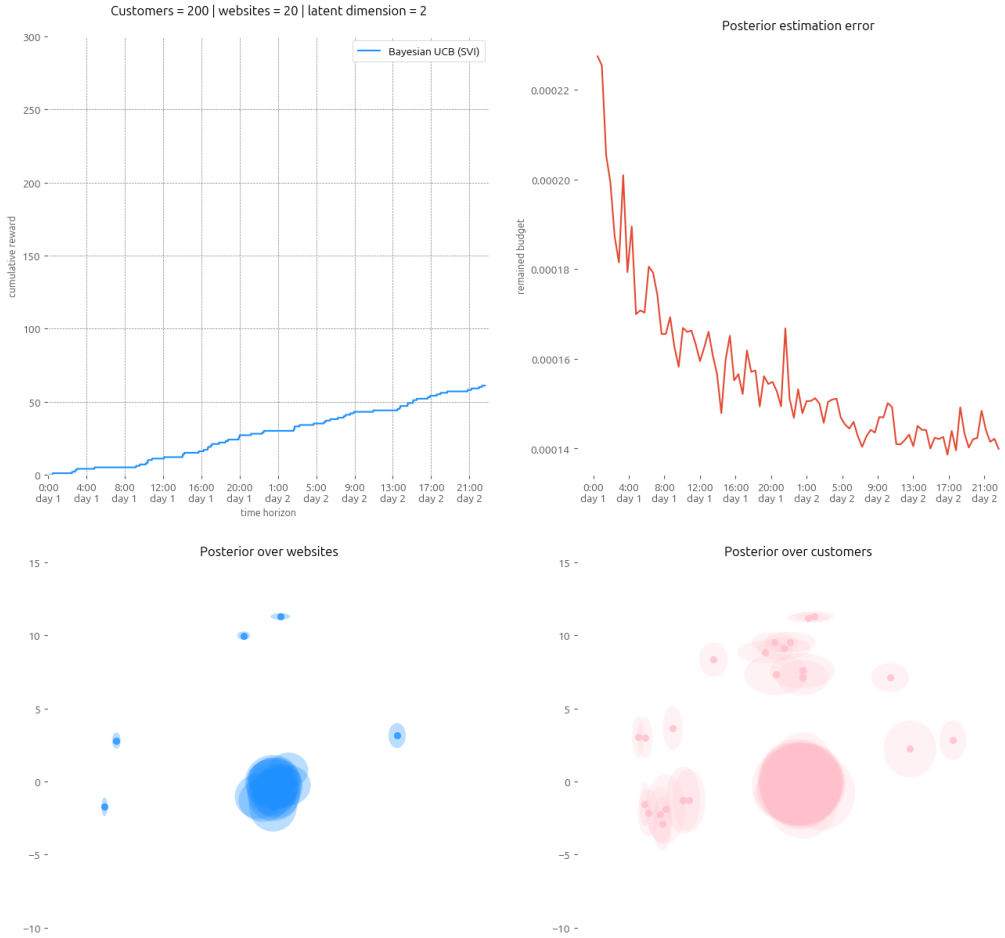
Probabilistic Matrix Factorization

Application: Ad click prediction and recommendation with Bayesian Upper Confidence Bound



Probabilistic Matrix Factorization

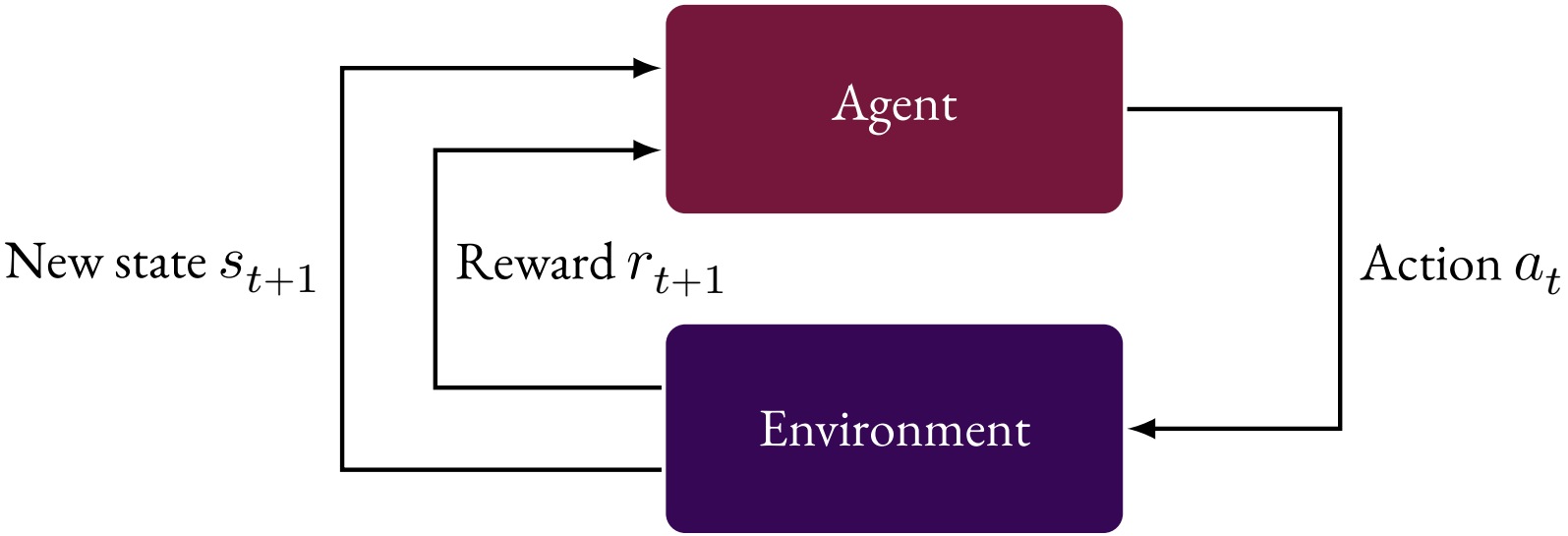
Application: Ad click prediction and recommendation with Bayesian Upper Confidence Bound



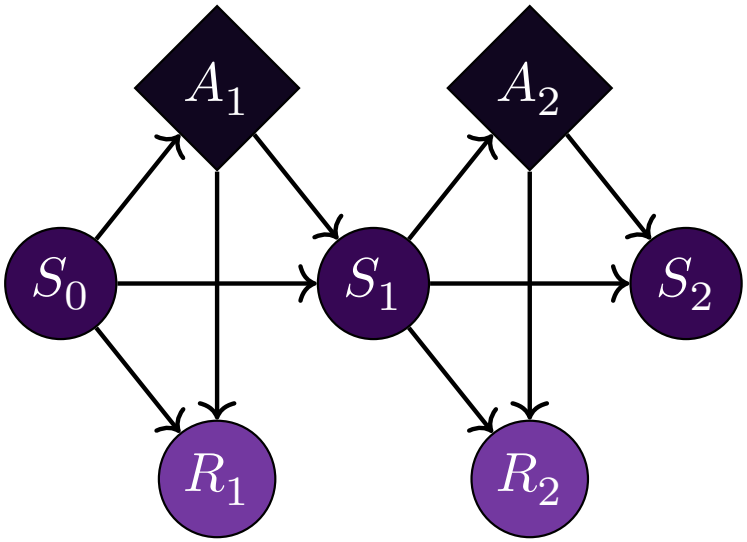
Reinforcement Learning

Sometimes, environment is not static and the decisions can affect it too

Reinforcement Learning



Markov Decision Process



...

S : State

A : Action

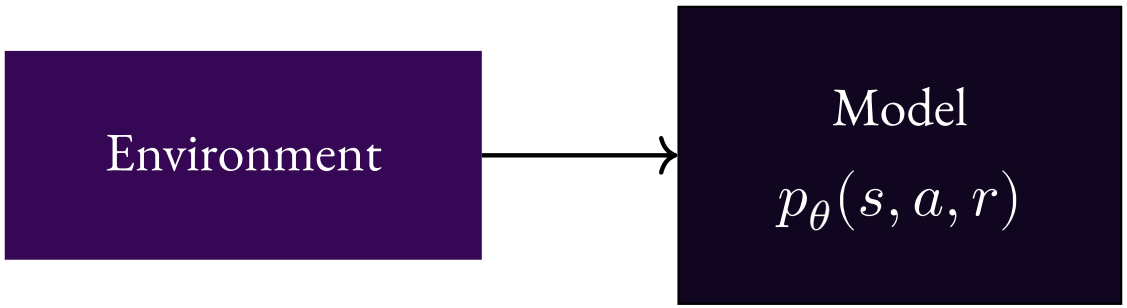
R : Reward

$\gamma \in (0, 1)$: Discount factor

The goal is to maximize $\mathbb{E}_\pi[G]$ the expected total reward $G = \sum_{t=1}^{\infty} \gamma^i R_i$

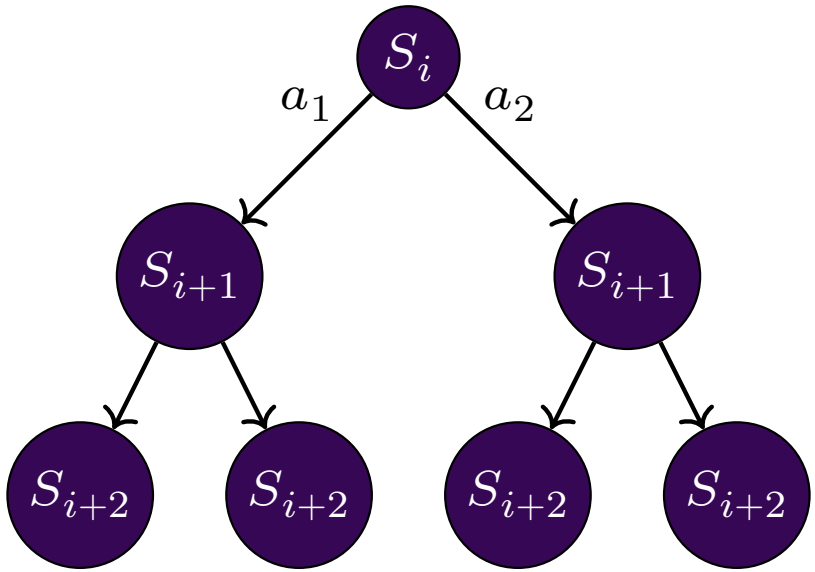
Model-Based RL

Main idea : designing a model of environment and using it for planning.



Model-Based RL

The model can be used to expand a search tree over future actions in order to perform approximate dynamic programming



Example: Monte Carlo Tree Search

Challenges with RL

- RL is not sample efficient.
- RL agents are difficult to train in problems with sparse reward.
- RL framework is too much dependent on the notion of extrinsic reward.
- RL agents are not modular.
- Robustness in non-stationery problems
- Generalizability

Active Inference

Is it possible to formulate decision-making as probabilistic inference?

Active Inference

$$p(s|o) = \frac{p(o|s)p(s)}{p(o)}$$

$$\text{Surprise} := -\log p(o) = -\log \sum_{s'} p(o, s')$$

Active Inference

How to avoid getting surprised?

- Understanding the environment better
- Affecting the surprising observations using actions

Active Inference

Variational Free Energy

$$
\begin{aligned}
-\log p(o) &= -\log \sum_{s'} p(o, s') \\
&= -\log \sum \frac{p(o, s')q(s')}{q(s')} \\
&= -\log \mathbb{E}_{s' \sim q}[\frac{p(o, s')}{q(s')}] \\
&\leq -\mathbb{E}_{s' \sim q}[\log \frac{p(o, s')}{q(s')}] \\
&= \mathbb{E}_{s' \sim q}[\log \frac{q(s')}{p(o, s')}]
\end{aligned}
$$

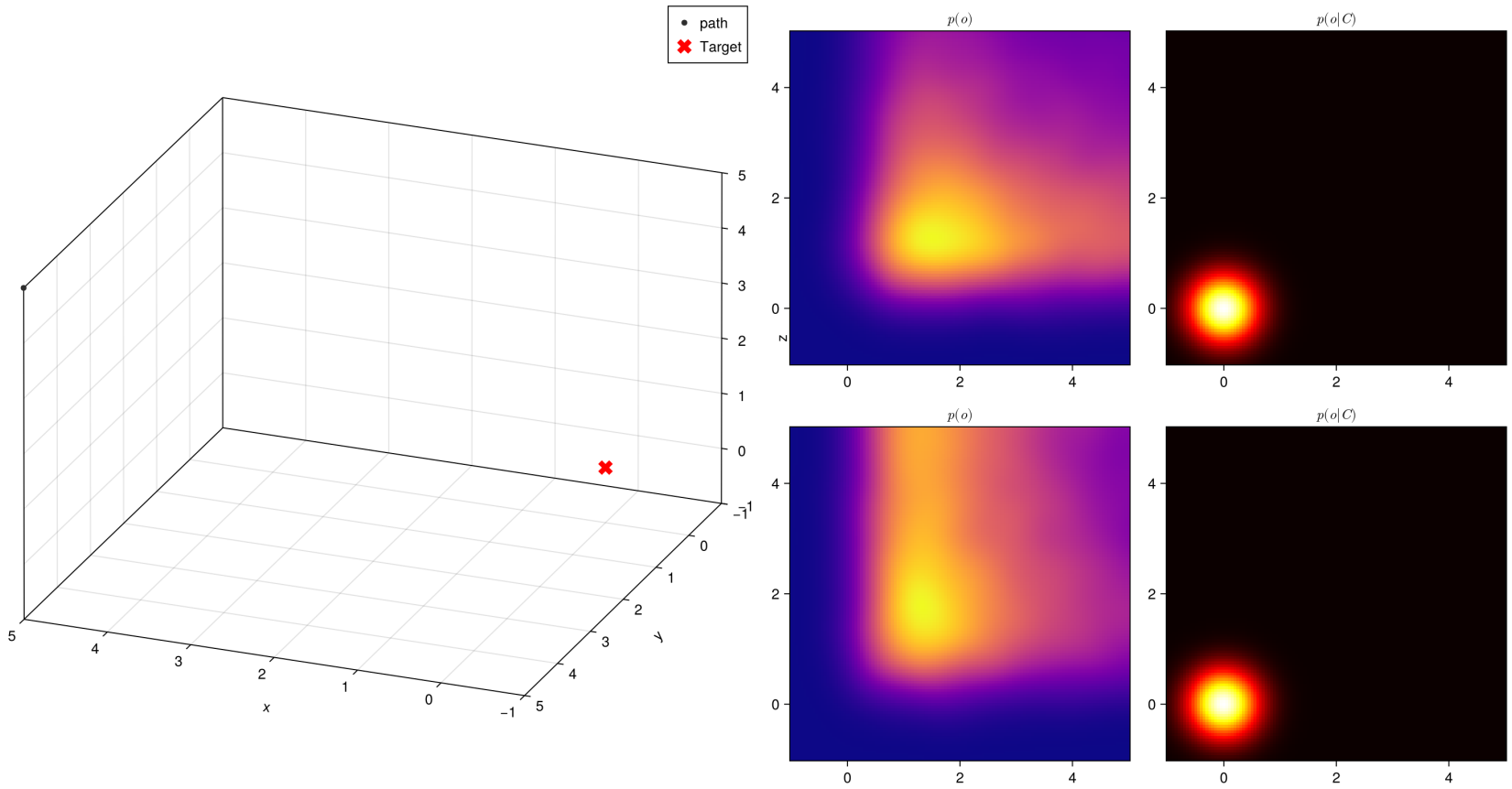
Active Inference

Instead of a scalar reward signal, In Active Inference we design a prior distribution $p(o|C)$ concentrated on the prefered outcomes. This way, deviation from $p(o|C)$ decreases evidence and consequently increases surprise.
So agent can decrease surprise by:

- Learning more about the environment dynamics, improving the variational approximation $q(s)$
- Choosing actions that do not lead to surprising states, reducing $D_{KL}[q(o)||p(o|C)]$

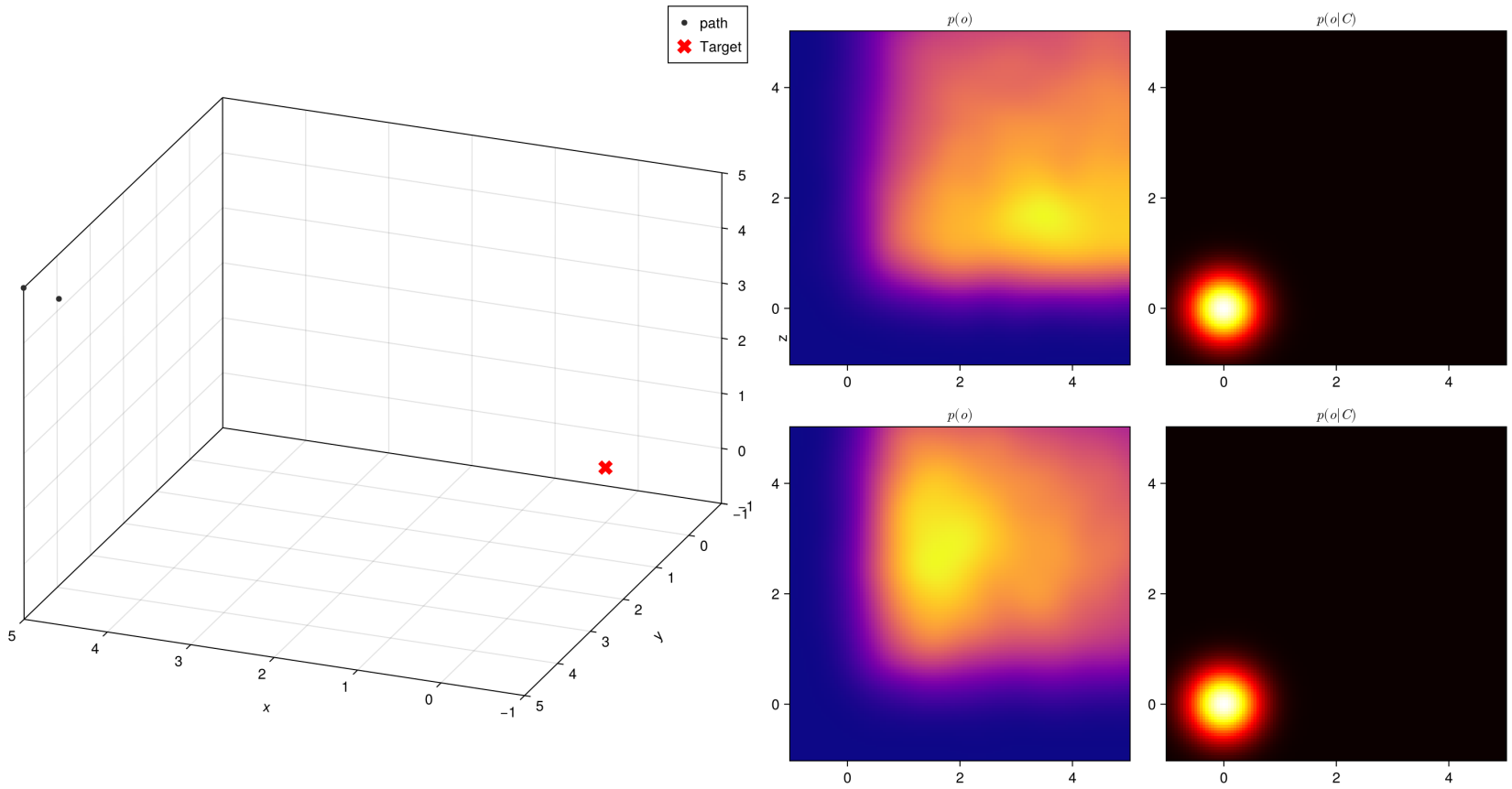
Active Inference

Example: Zero-shot navigation with uncertain dynamics



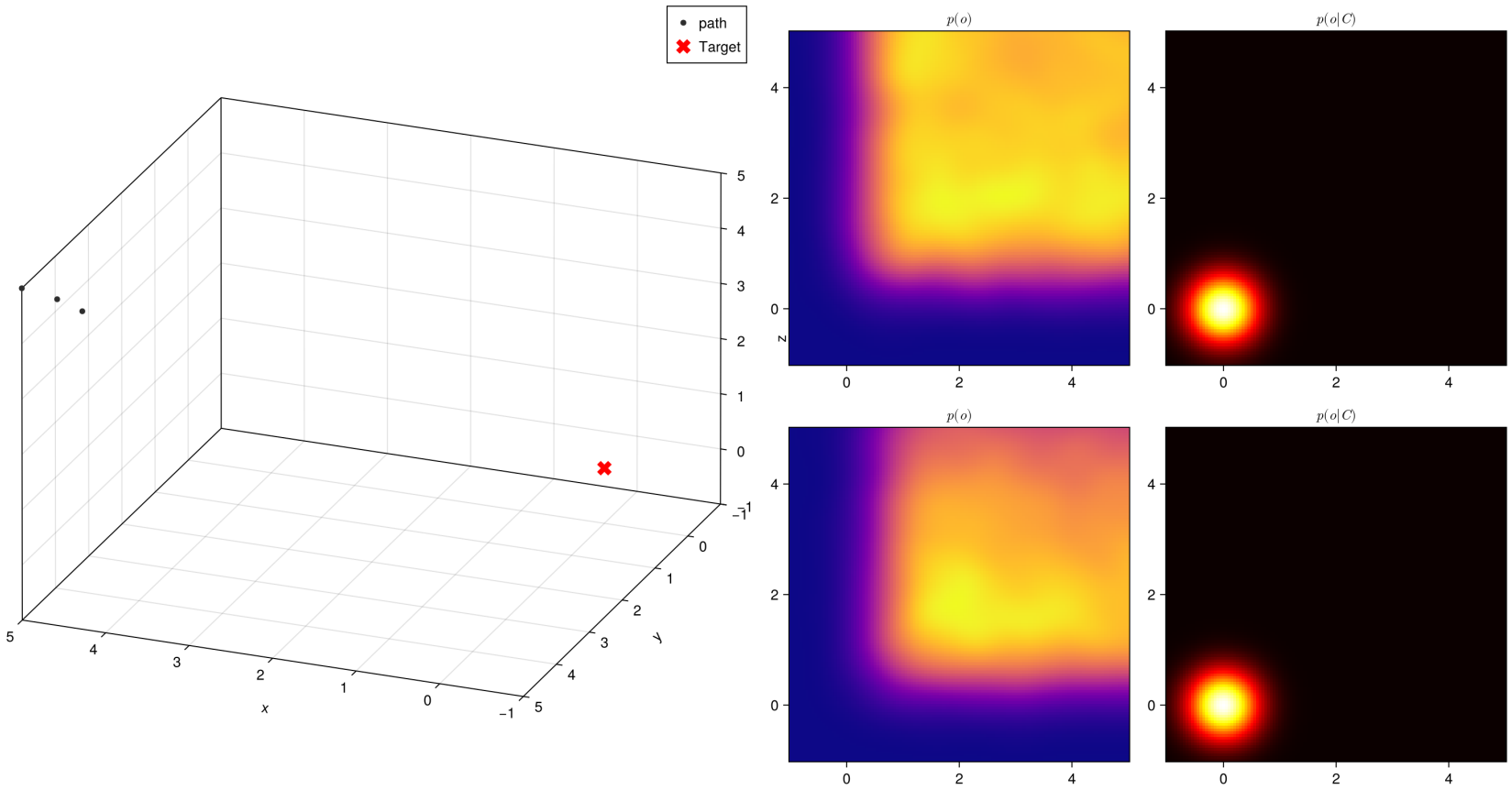
Active Inference

Example: Zero-shot navigation with uncertain dynamics



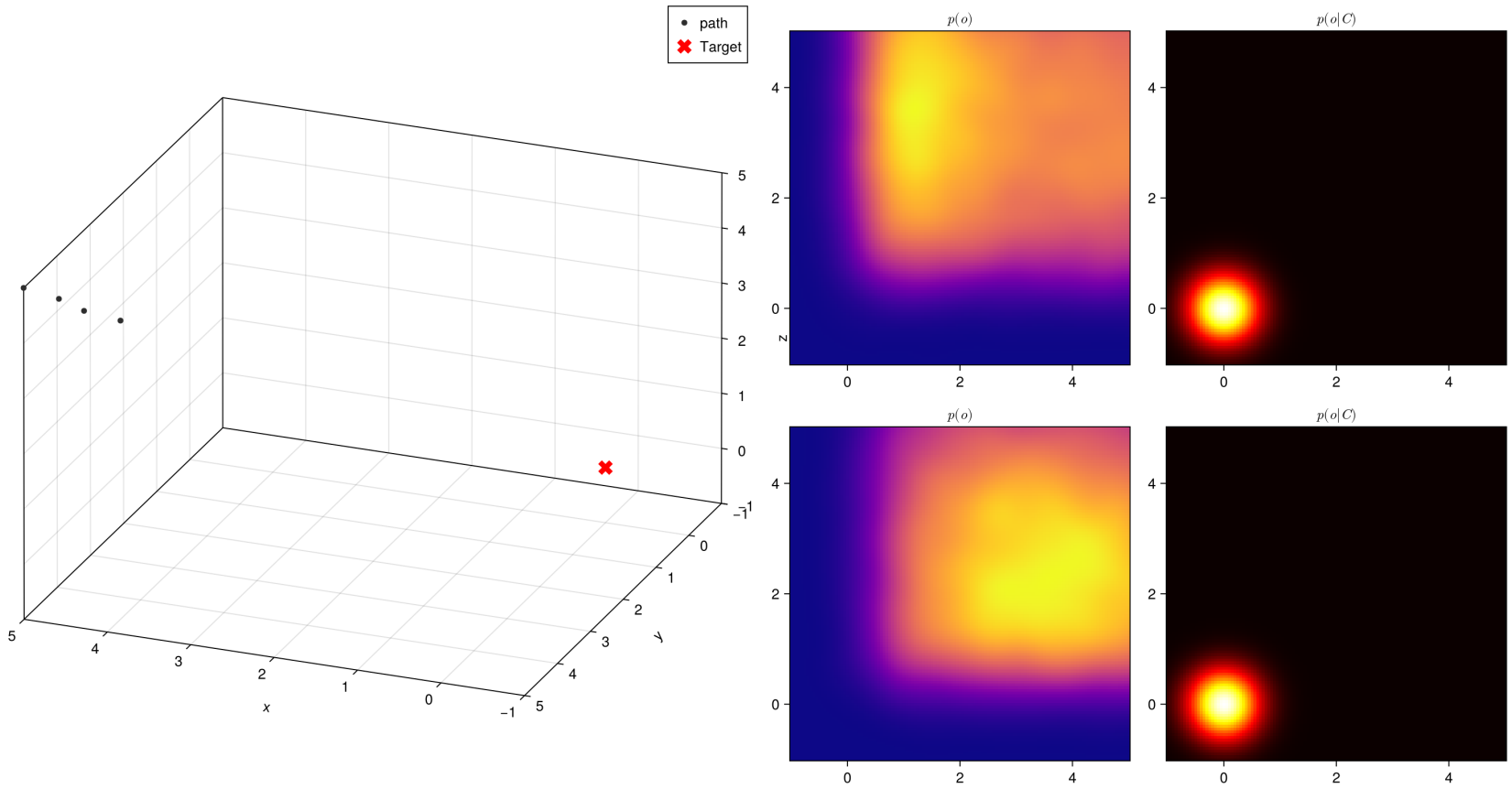
Active Inference

Example: Zero-shot navigation with uncertain dynamics



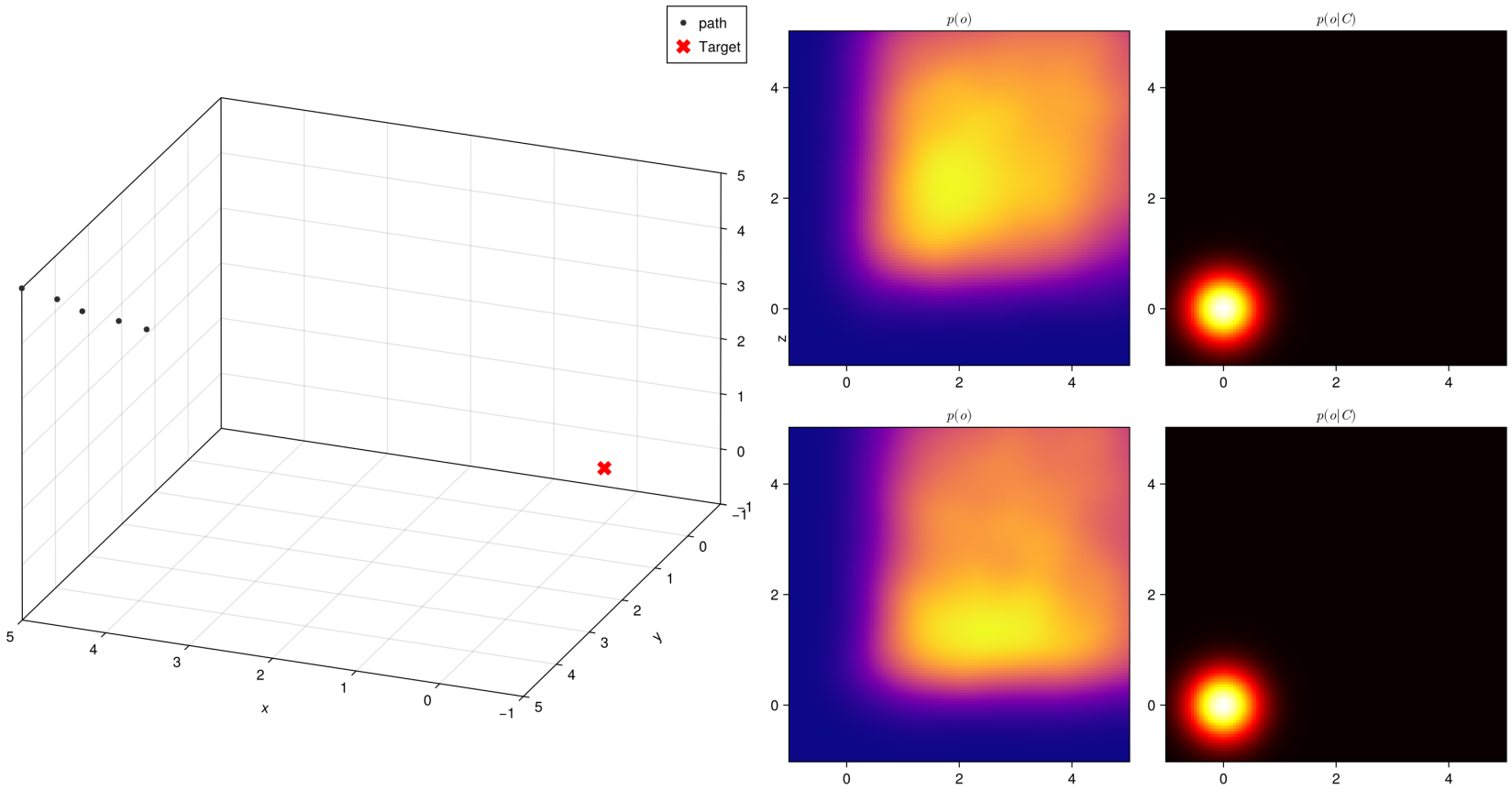
Active Inference

Example: Zero-shot navigation with uncertain dynamics



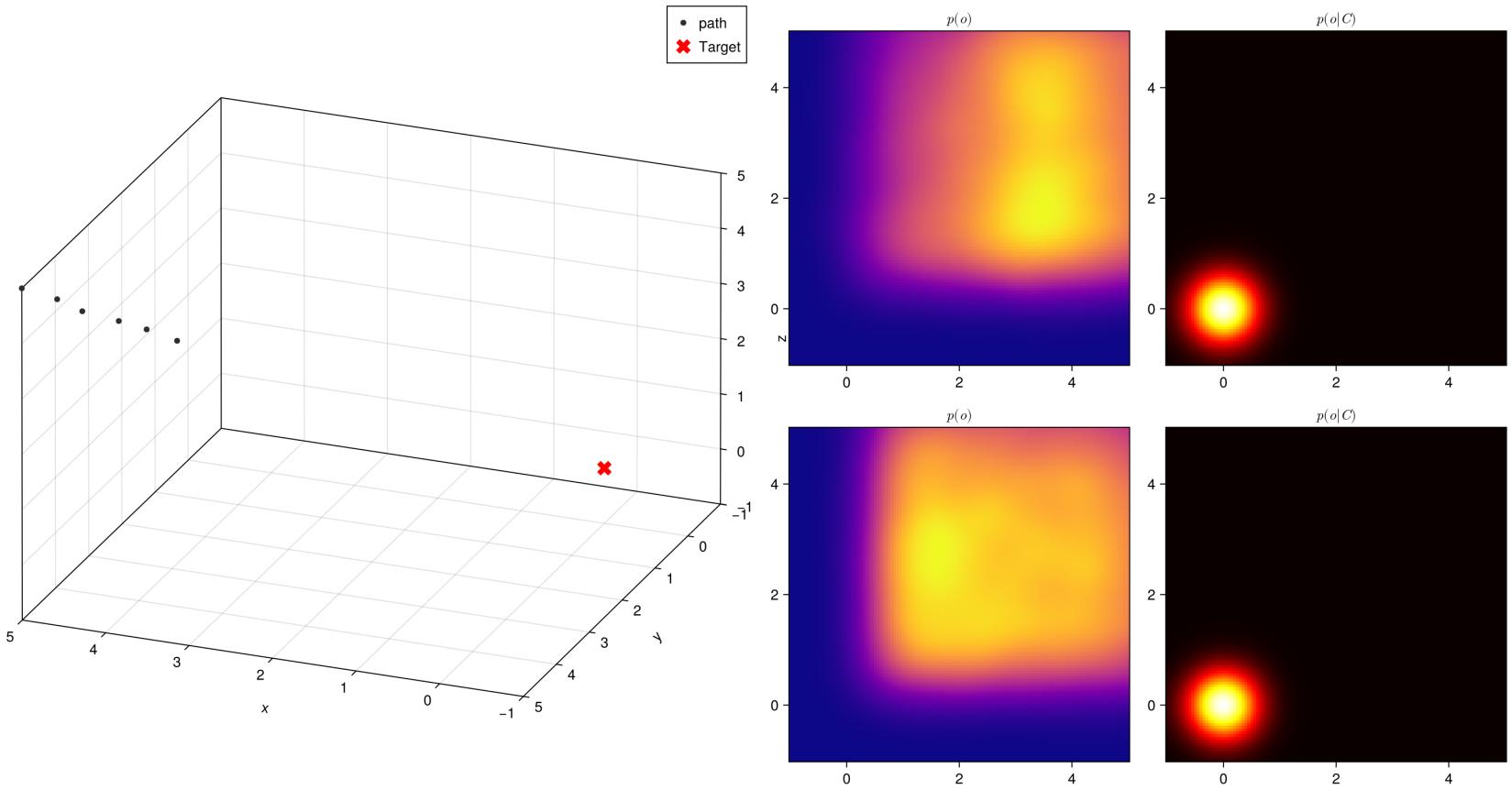
Active Inference

Example: Zero-shot navigation with uncertain dynamics



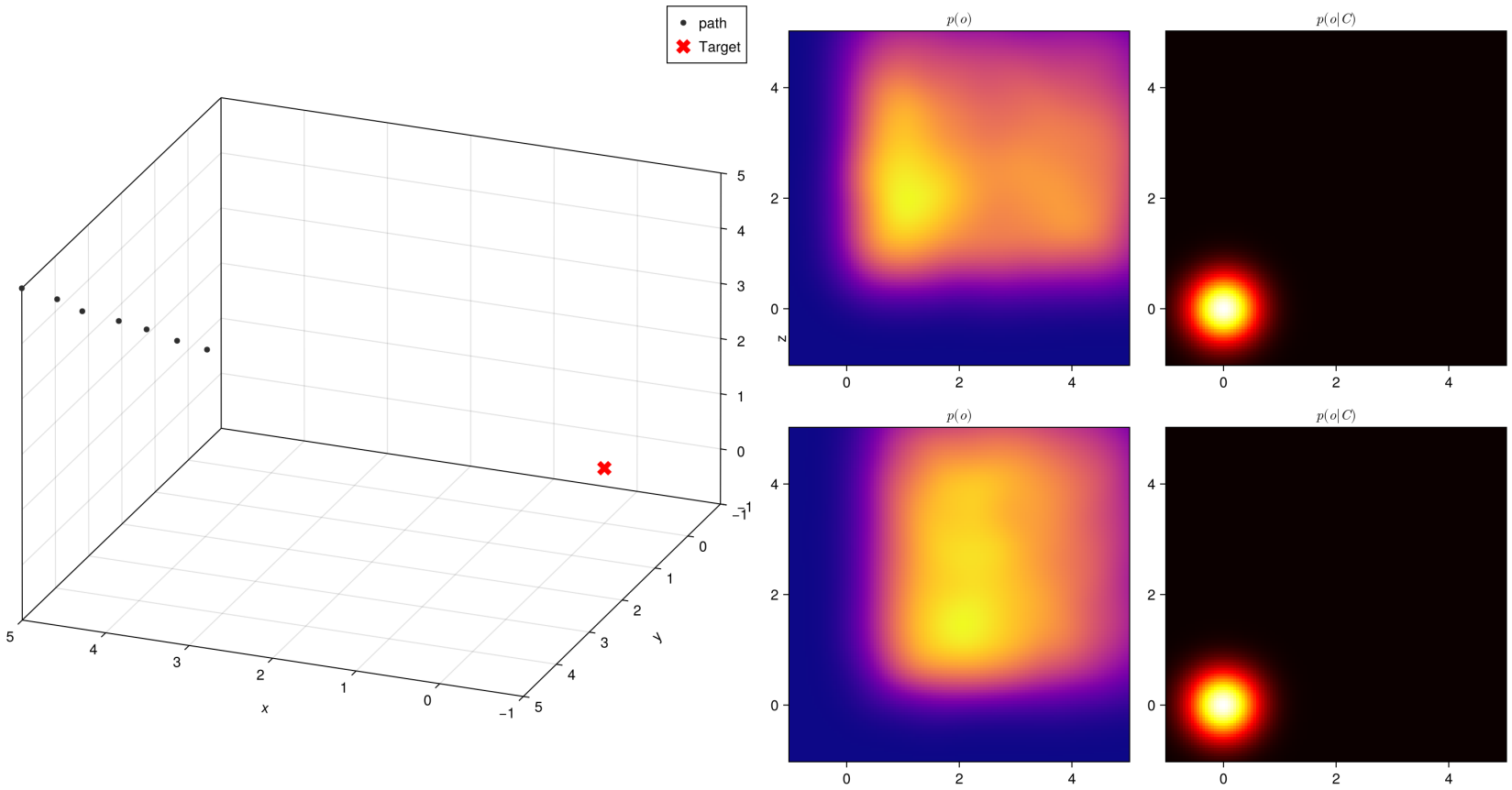
Active Inference

Example: Zero-shot navigation with uncertain dynamics



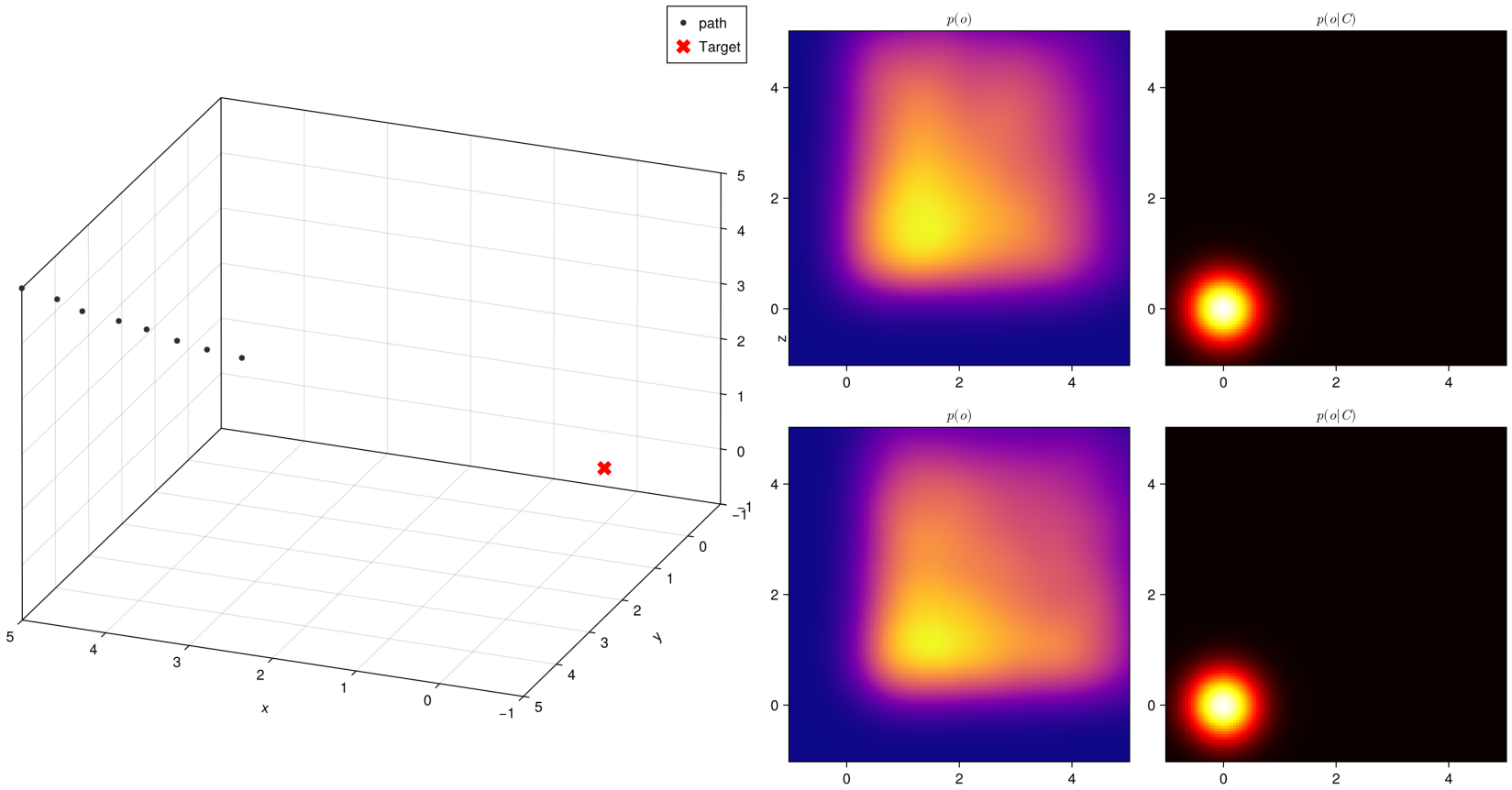
Active Inference

Example: Zero-shot navigation with uncertain dynamics



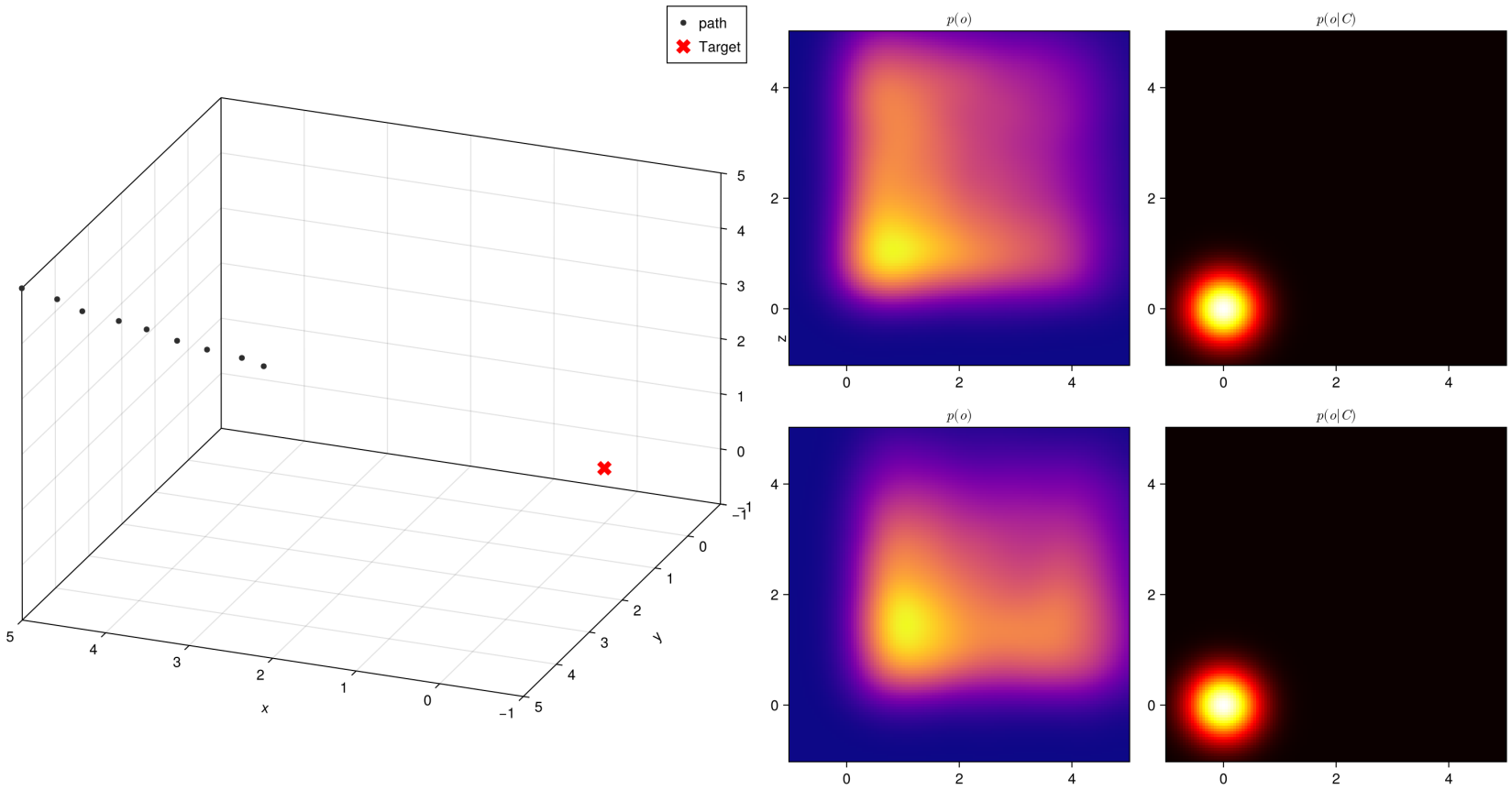
Active Inference

Example: Zero-shot navigation with uncertain dynamics



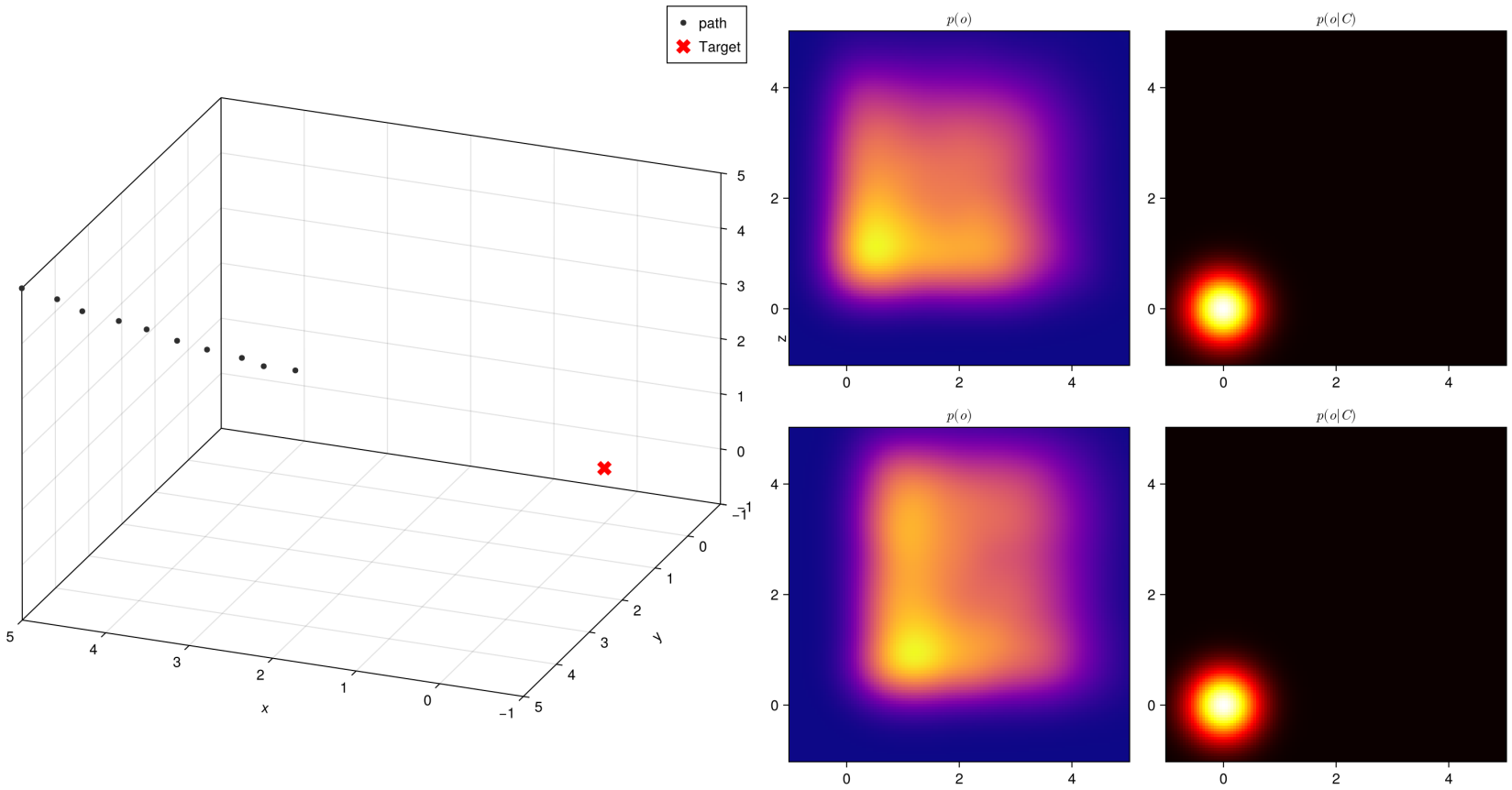
Active Inference

Example: Zero-shot navigation with uncertain dynamics



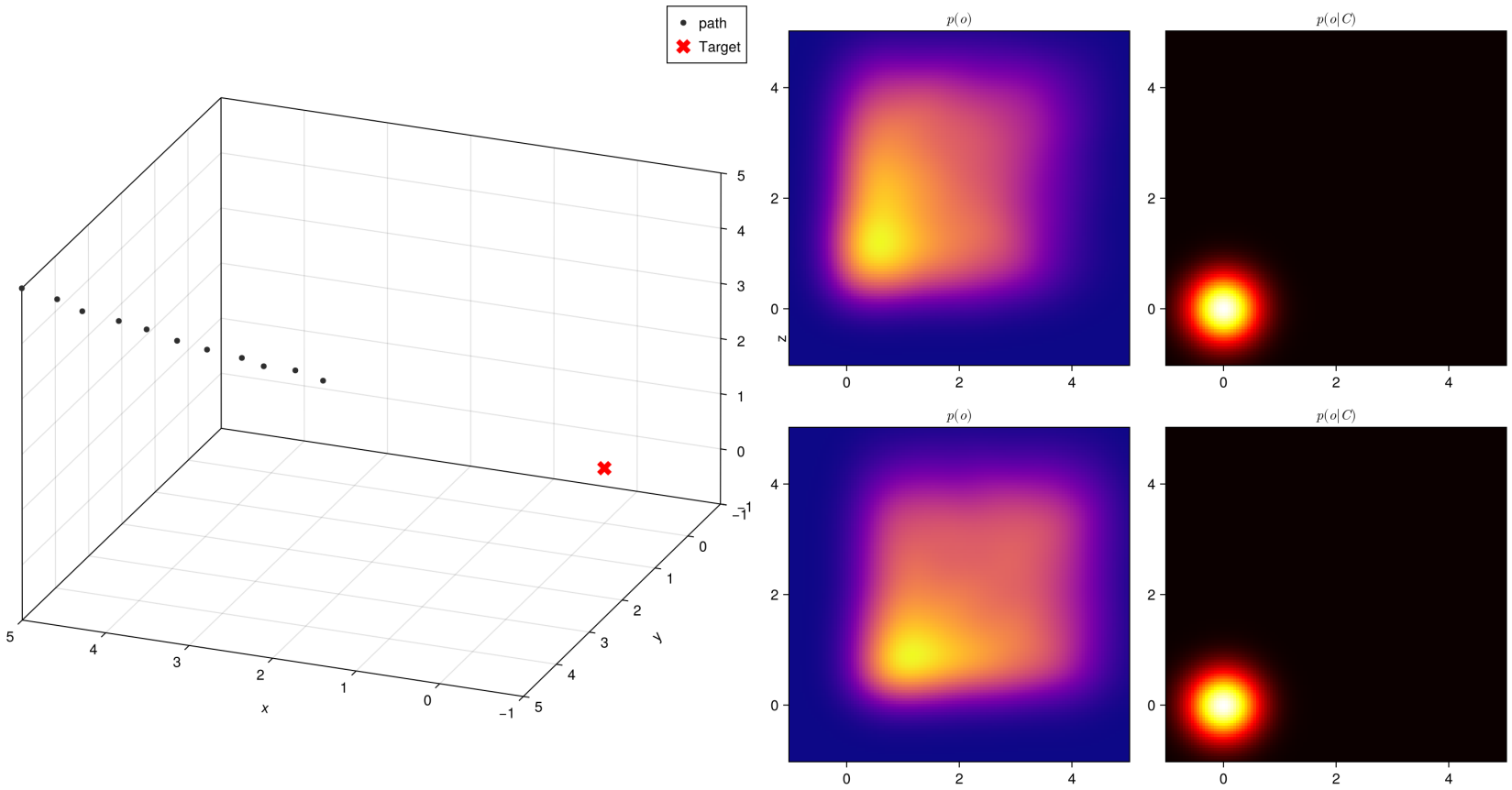
Active Inference

Example: Zero-shot navigation with uncertain dynamics



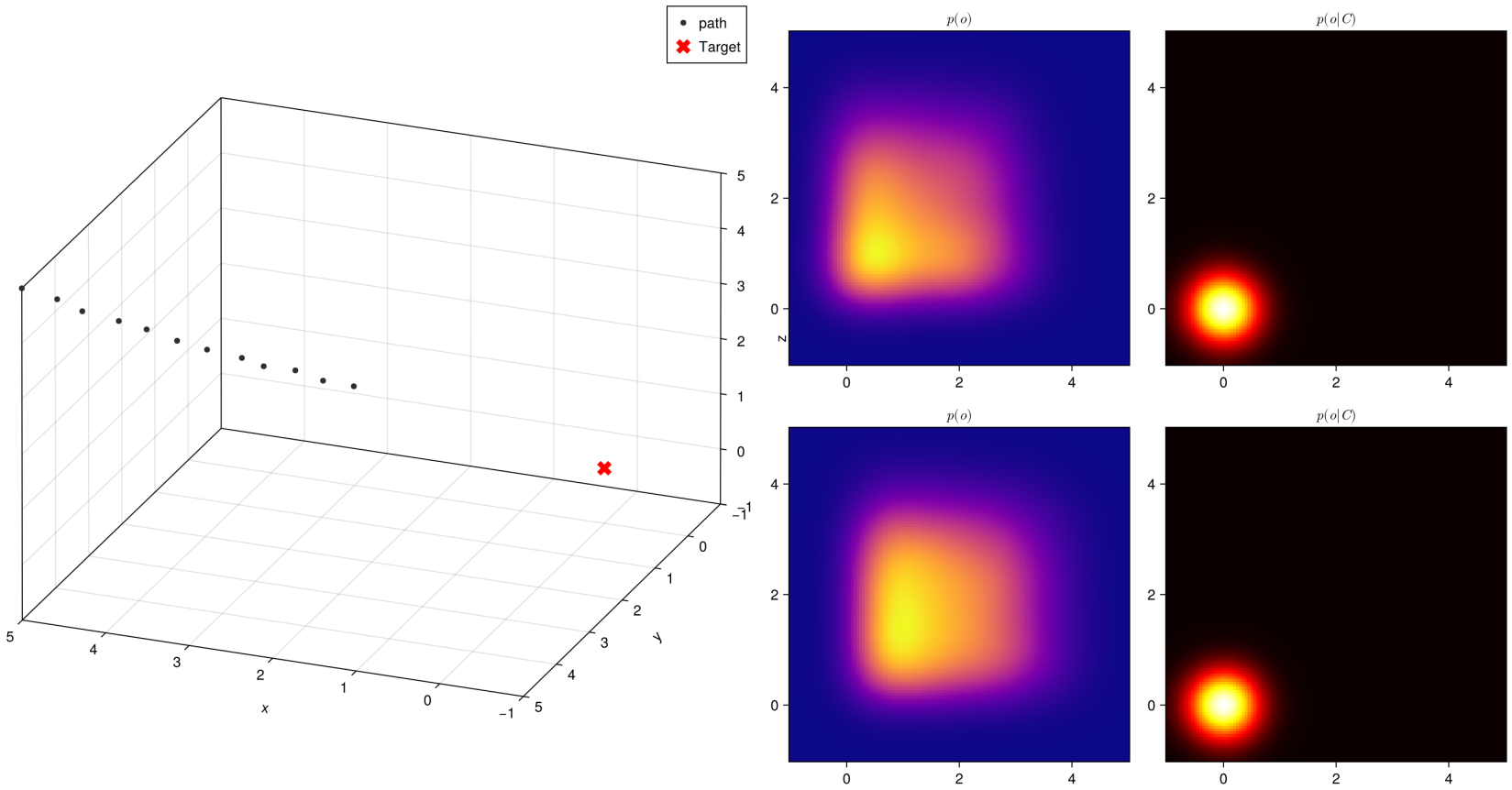
Active Inference

Example: Zero-shot navigation with uncertain dynamics



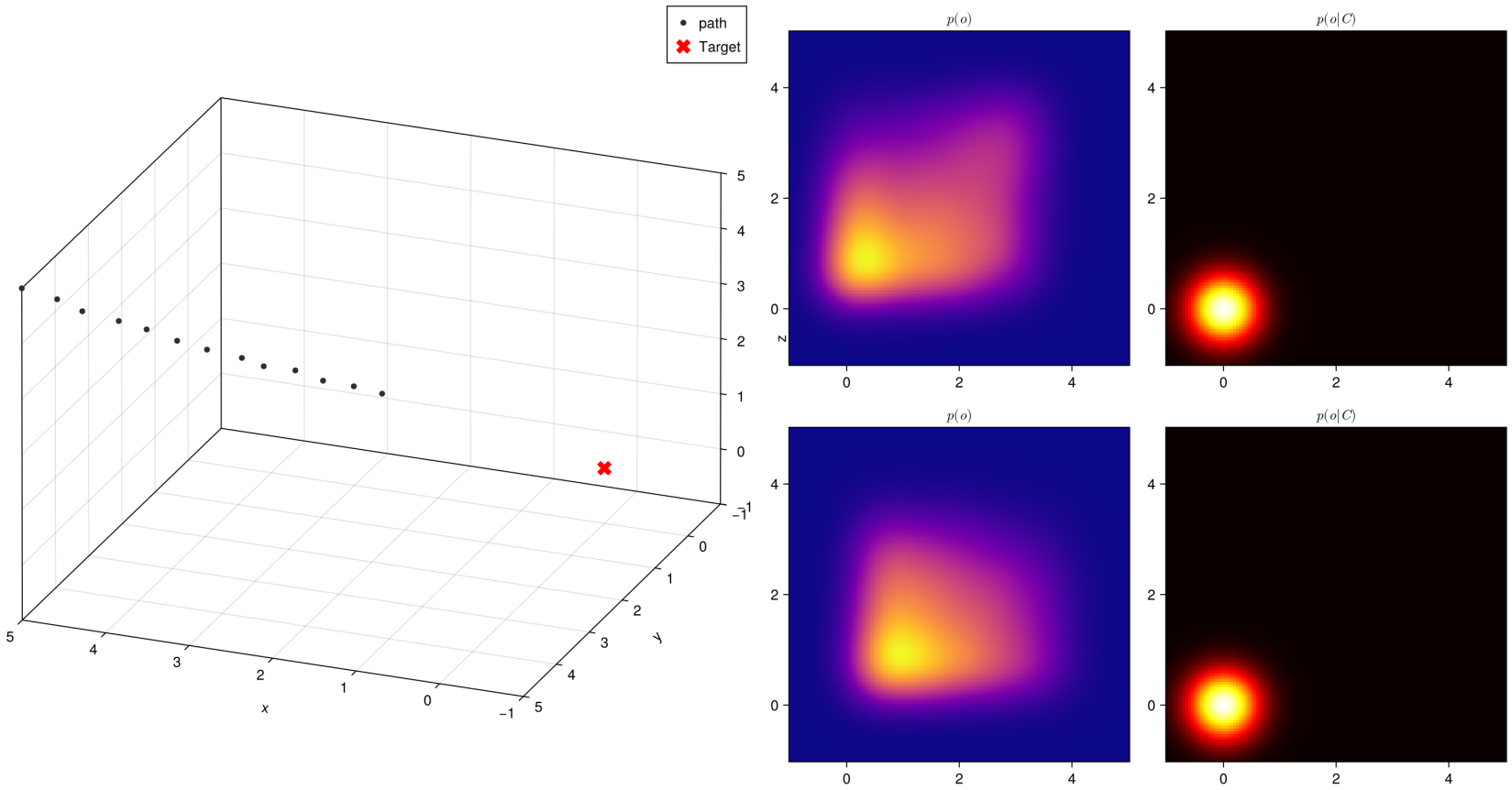
Active Inference

Example: Zero-shot navigation with uncertain dynamics



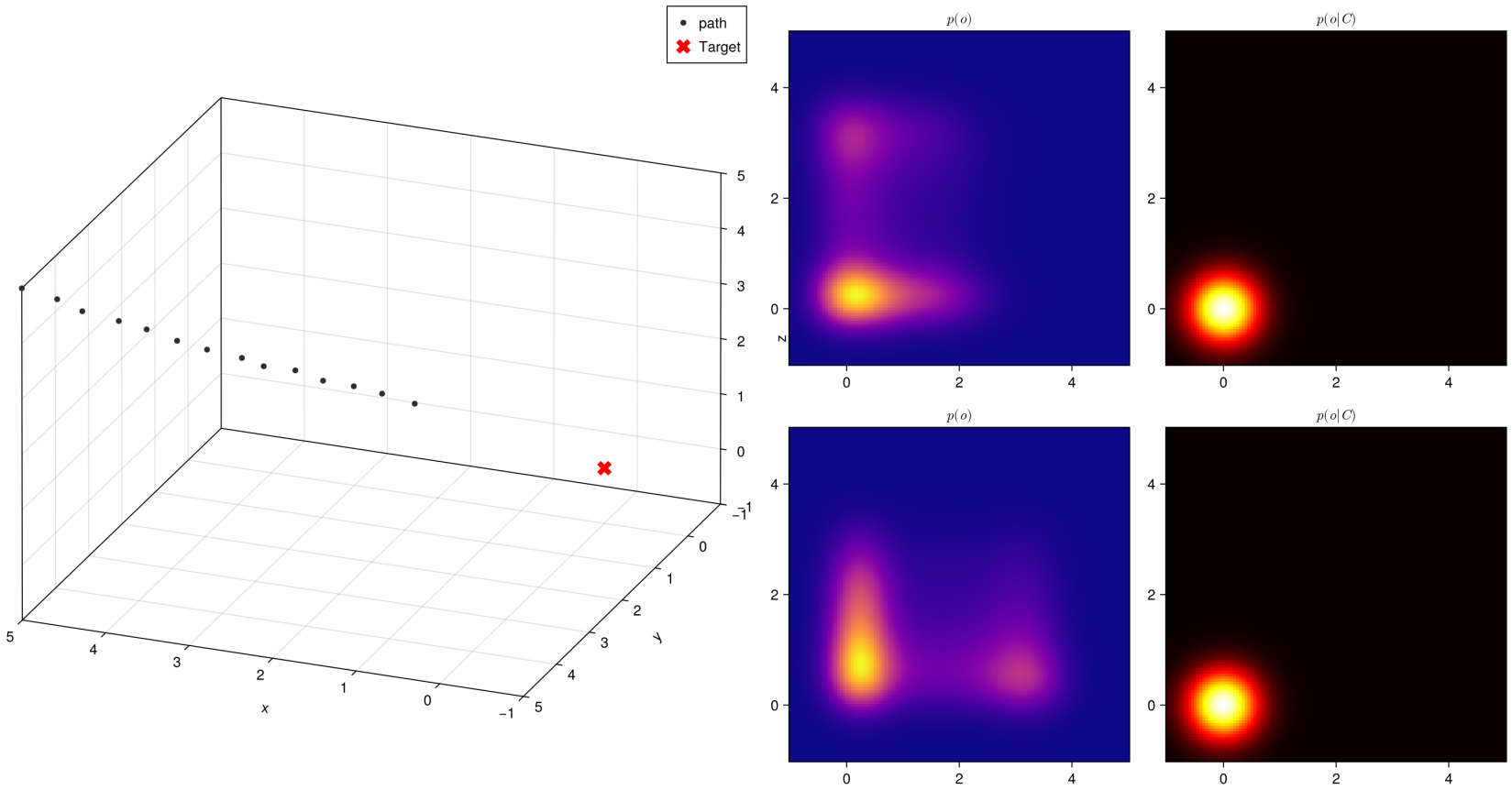
Active Inference

Example: Zero-shot navigation with uncertain dynamics



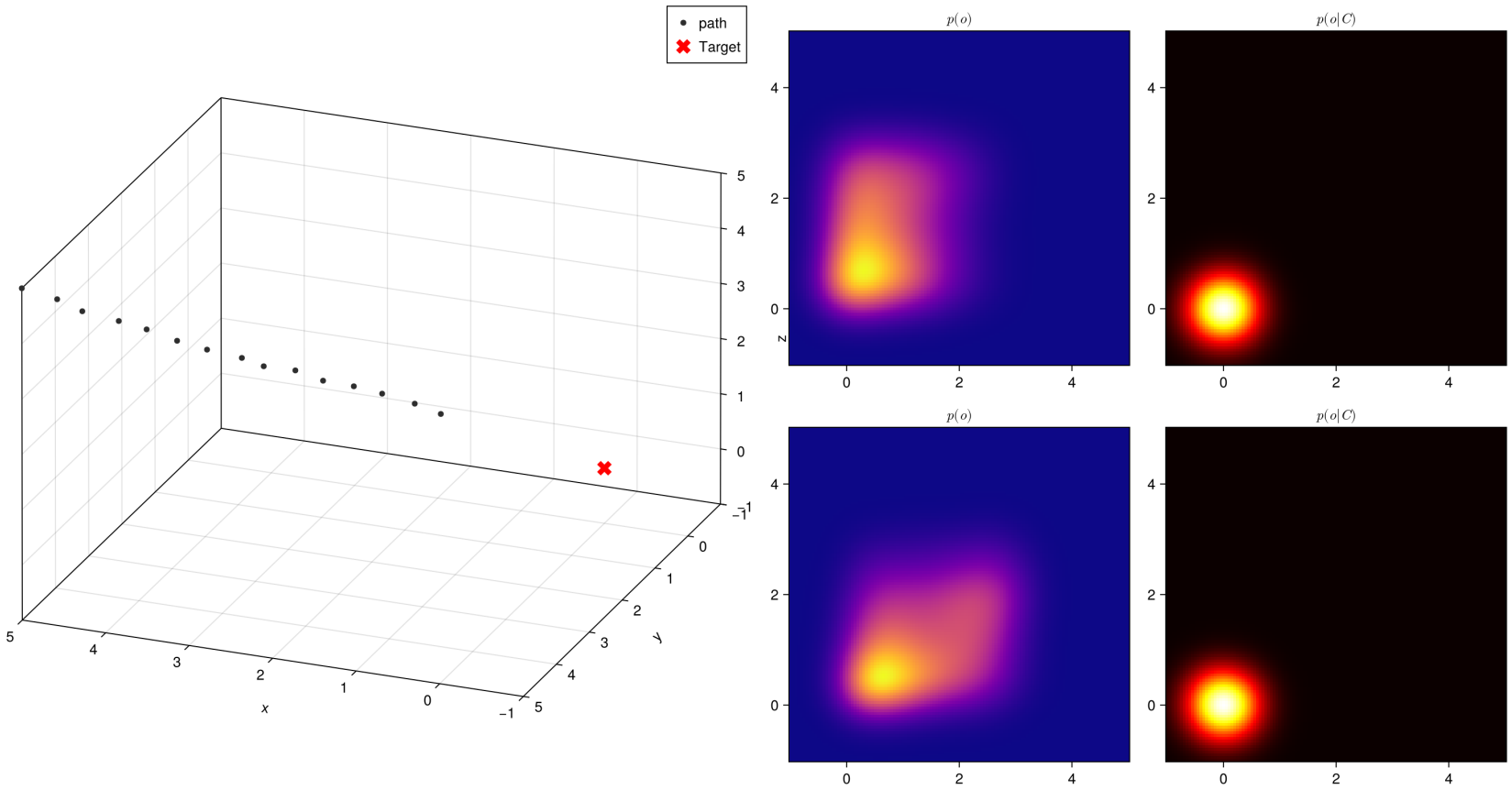
Active Inference

Example: Zero-shot navigation with uncertain dynamics



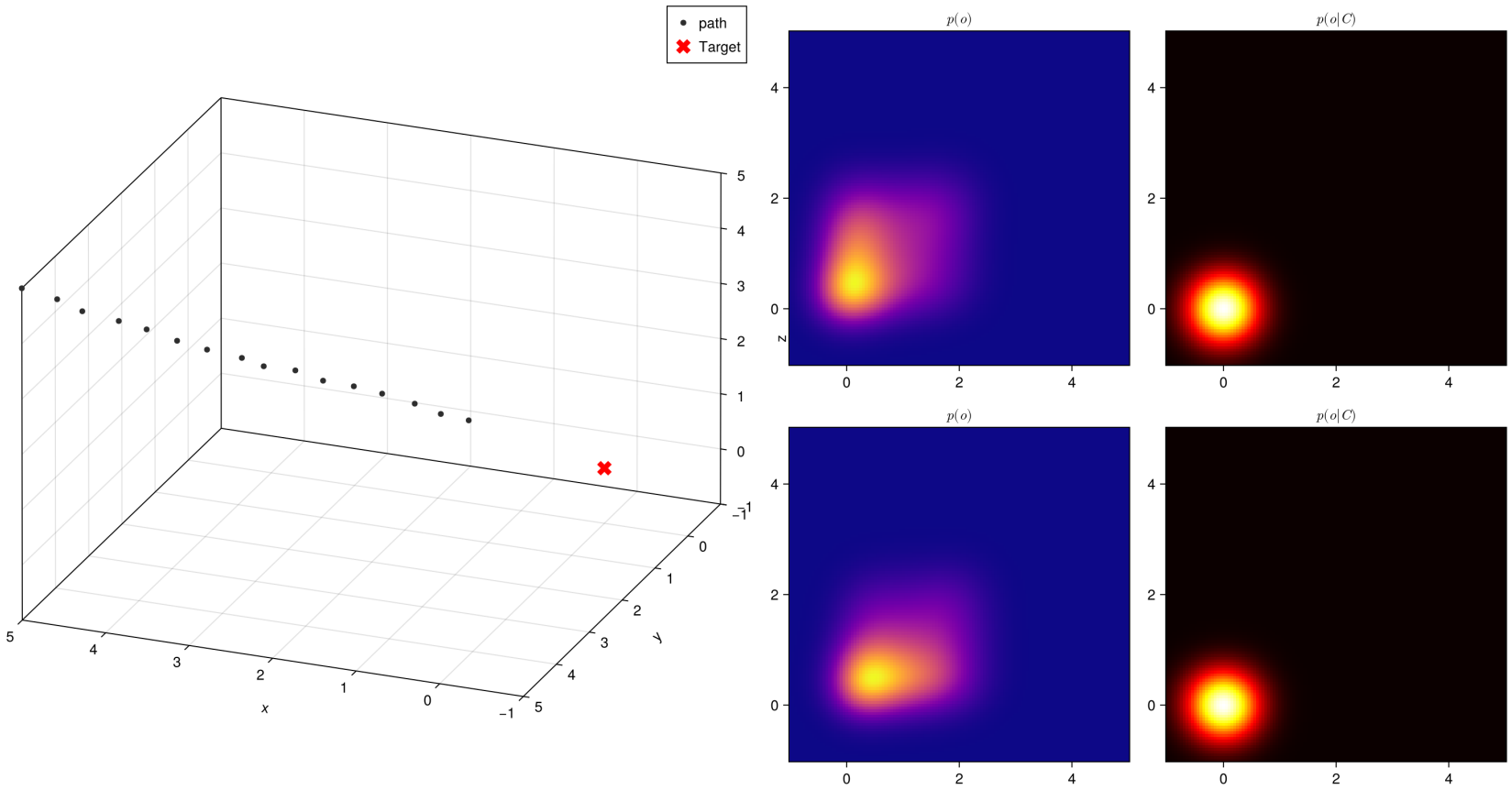
Active Inference

Example: Zero-shot navigation with uncertain dynamics



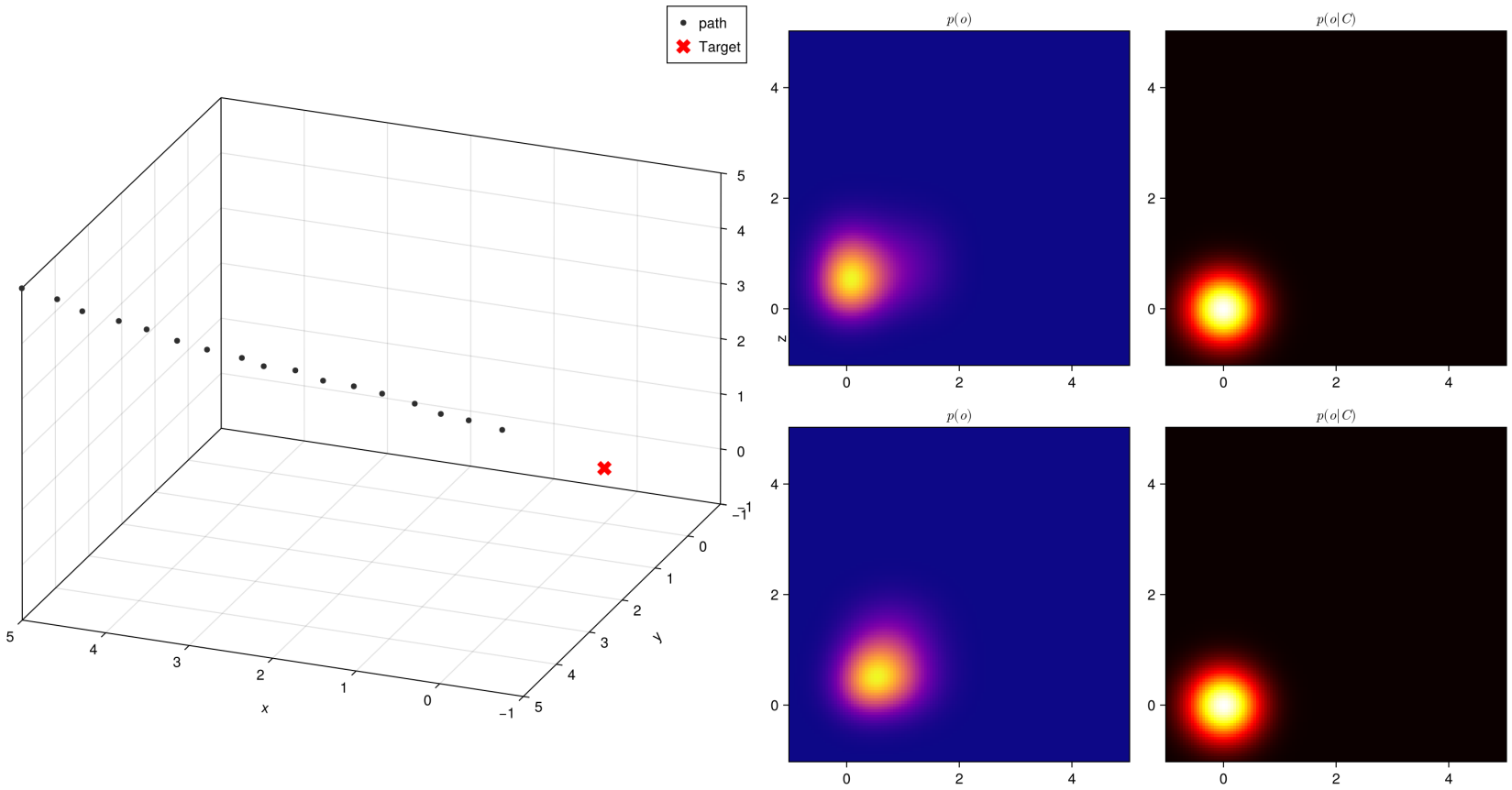
Active Inference

Example: Zero-shot navigation with uncertain dynamics



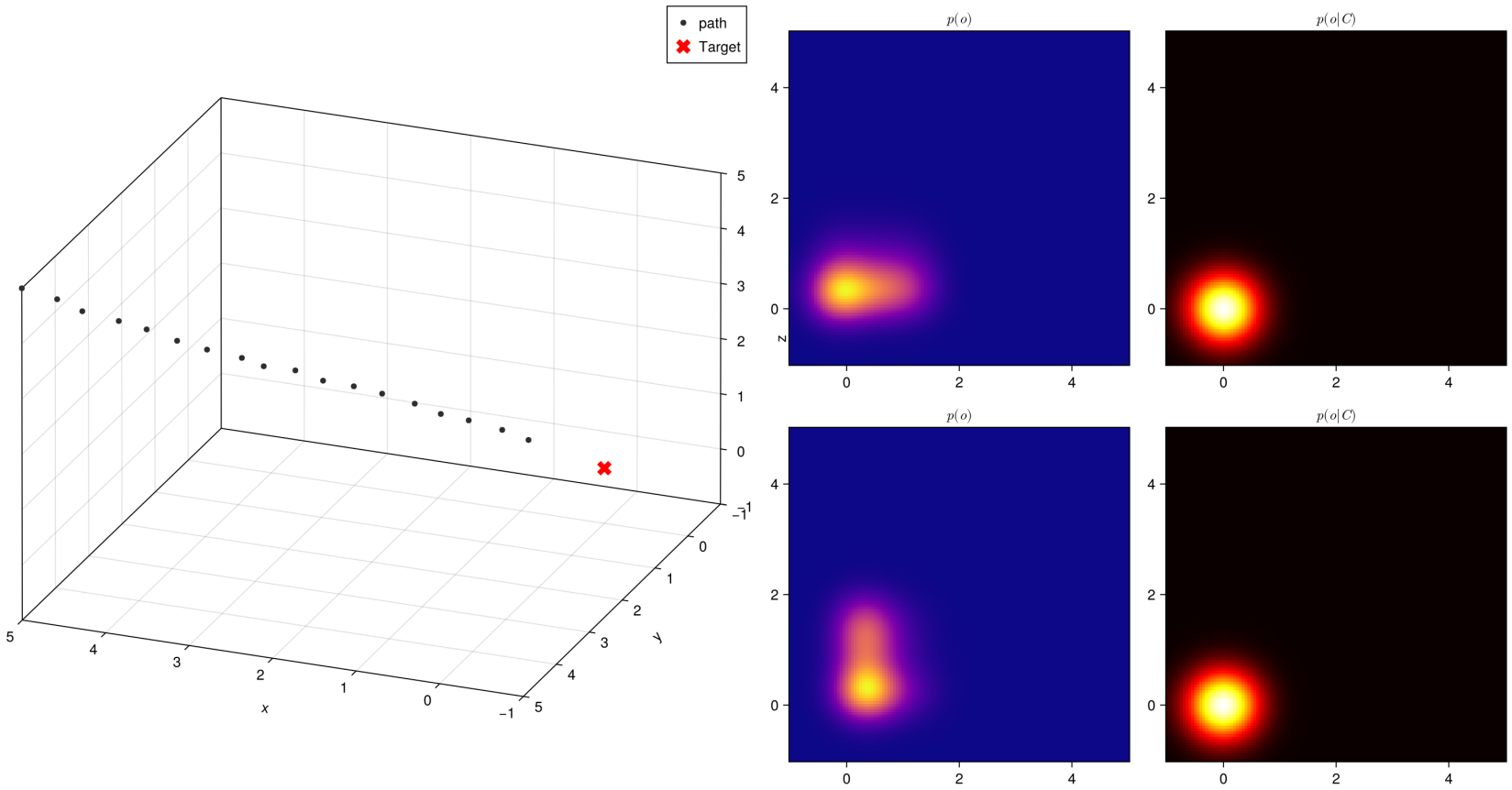
Active Inference

Example: Zero-shot navigation with uncertain dynamics



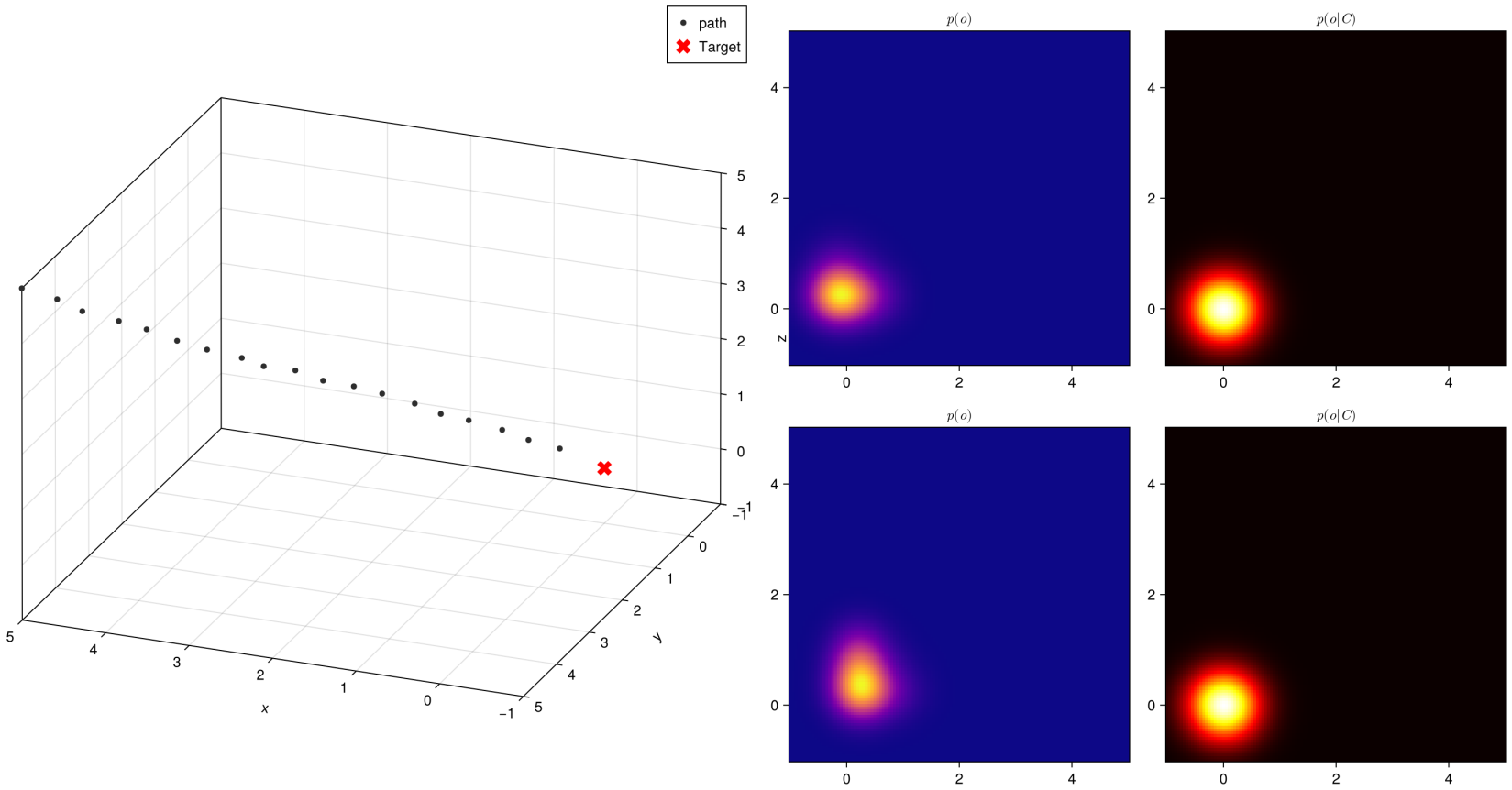
Active Inference

Example: Zero-shot navigation with uncertain dynamics



Active Inference

Example: Zero-shot navigation with uncertain dynamics



Active Inference

Example: Zero-shot navigation with uncertain dynamics