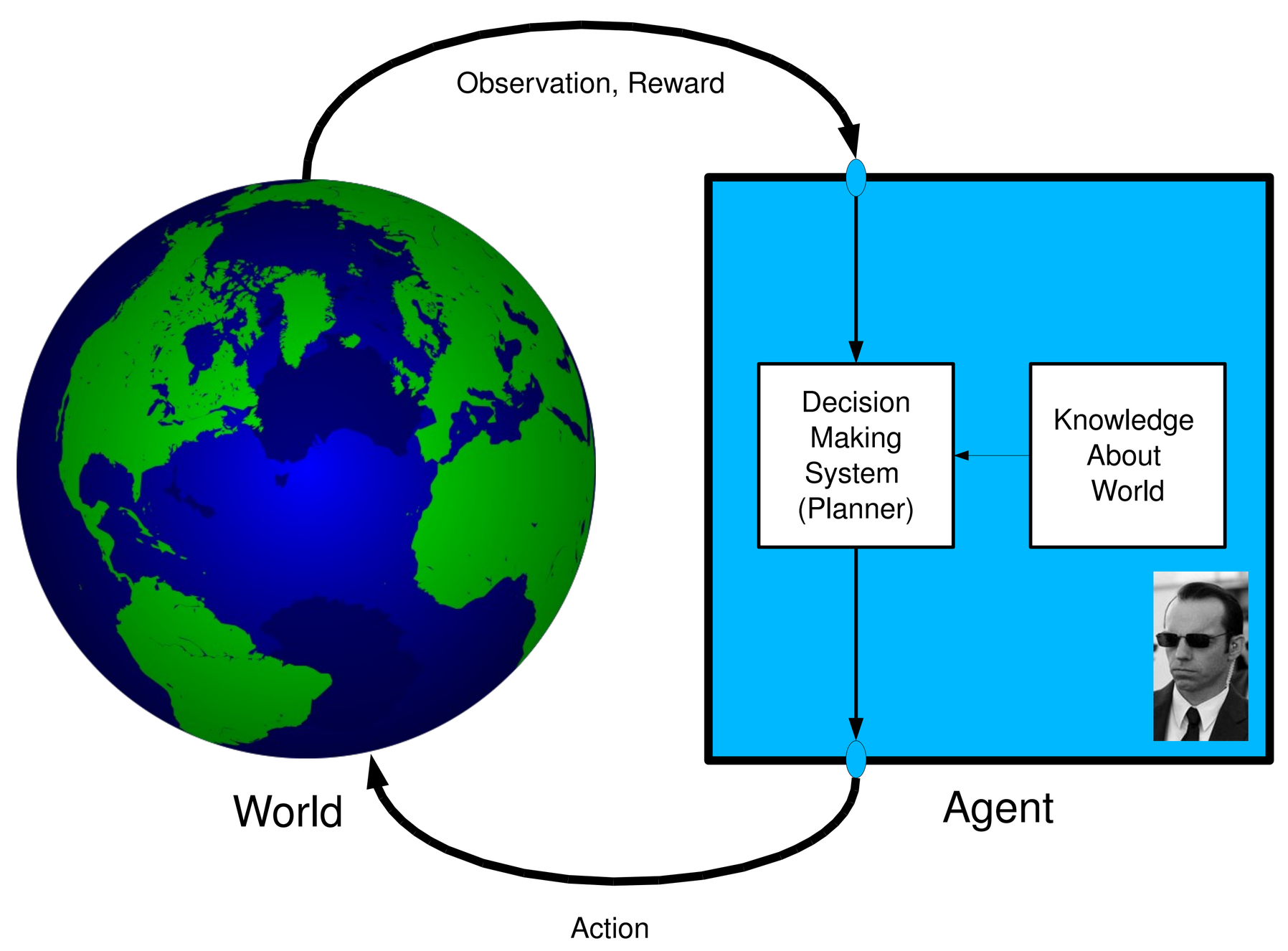
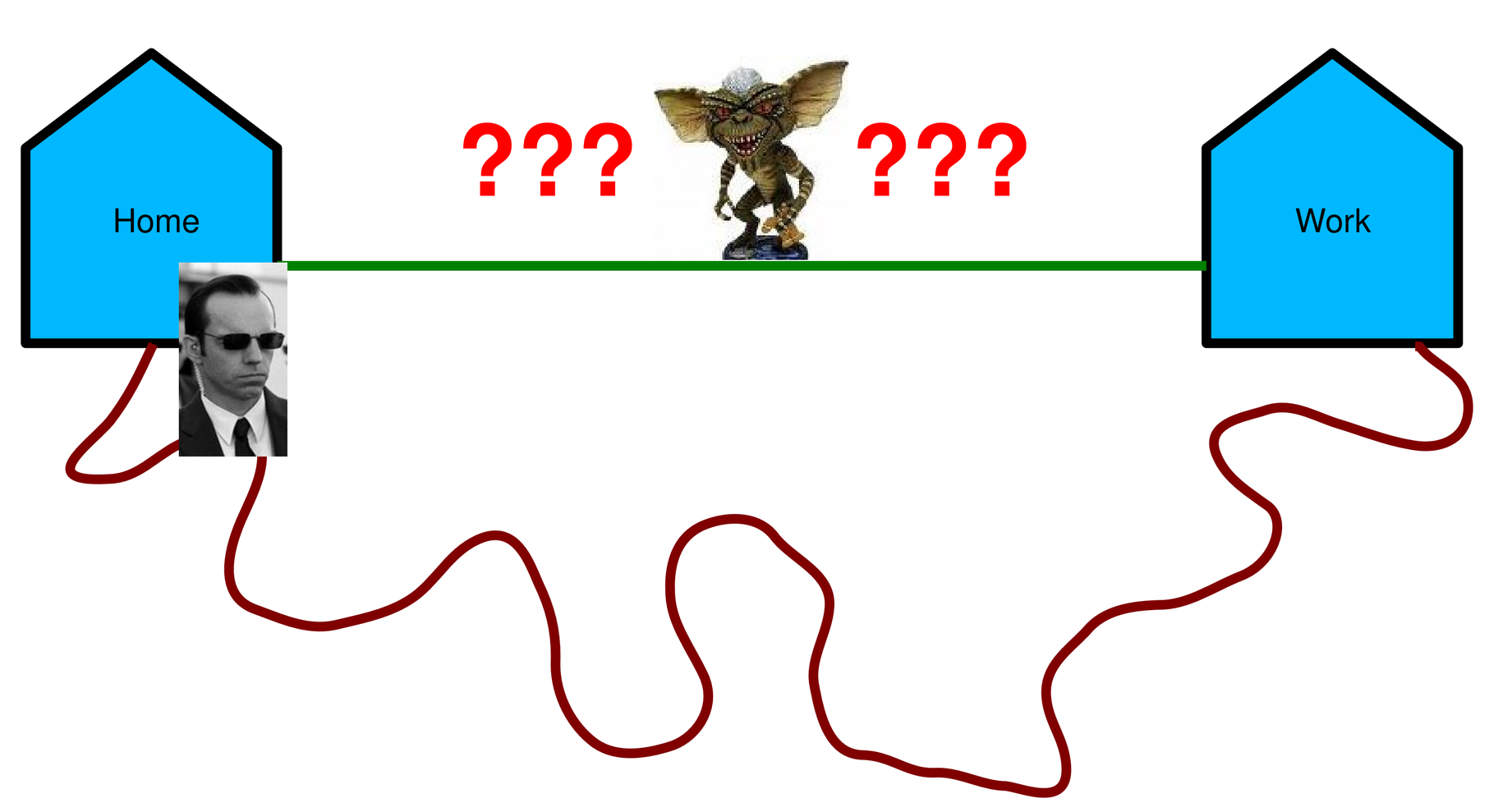
Reinforcement Learning

Computerized agents—robots, automated telephone systems, cell-phone and document word-completion managers—are all around us. Somehow, these agents must reason about the world and learn to interact with the environment, people, and other systems in a sensible manner. In reinforcement learning (RL), the agent observes some properties of the world, and based on its own internal model of how the world behaves, decides what actions are most likely to maximize its expected long-term reward.

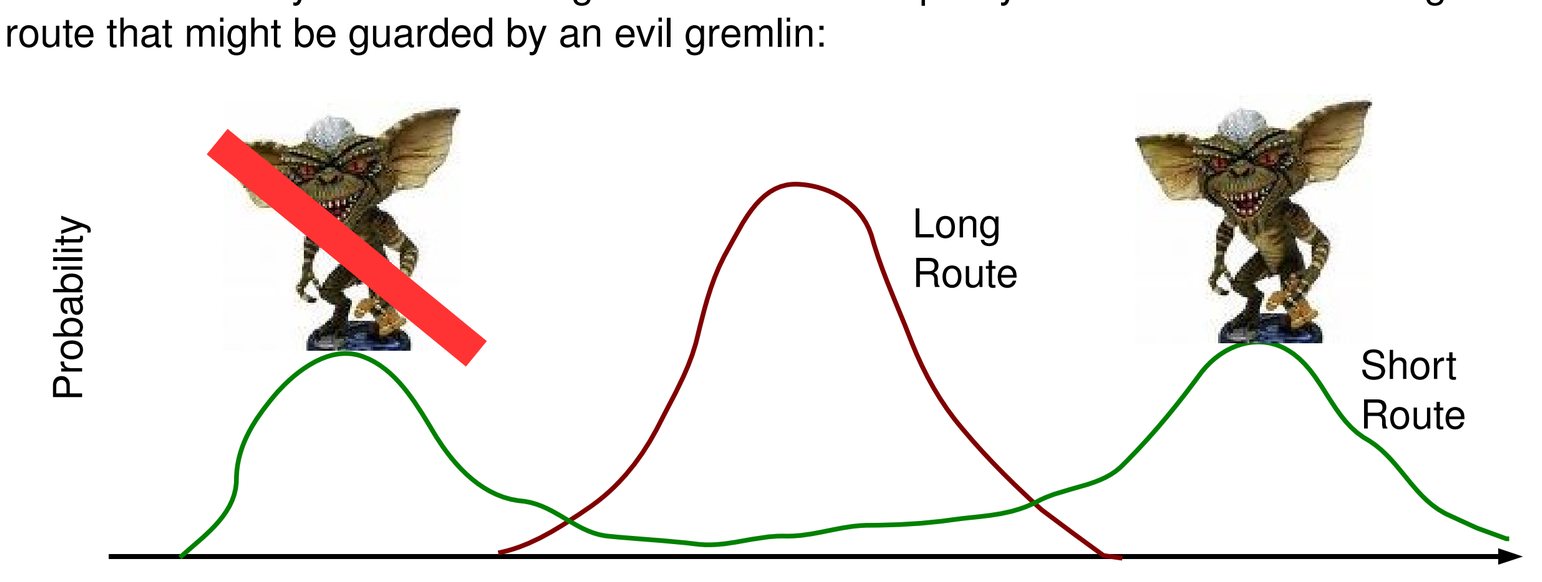


Issues

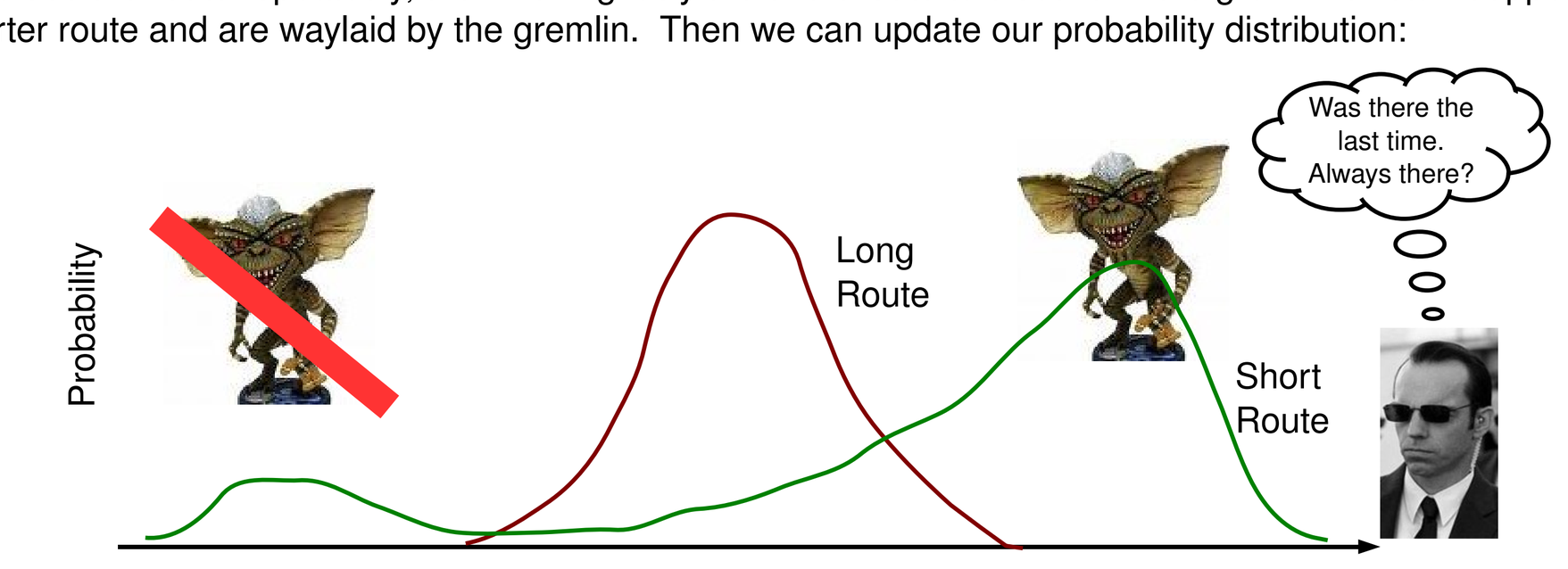
Exploration vs. Exploitation:
- explore: sacrifice short term reward, gain knowledge thus potentially getting more reward in the long run
- exploit: go for rewards now, potentially missing out on better opportunities



Suppose there are two routes you can take to get to work: we are pretty sure that there is a long but reliable route and a shorter route that might be guarded by an evil gremlin:



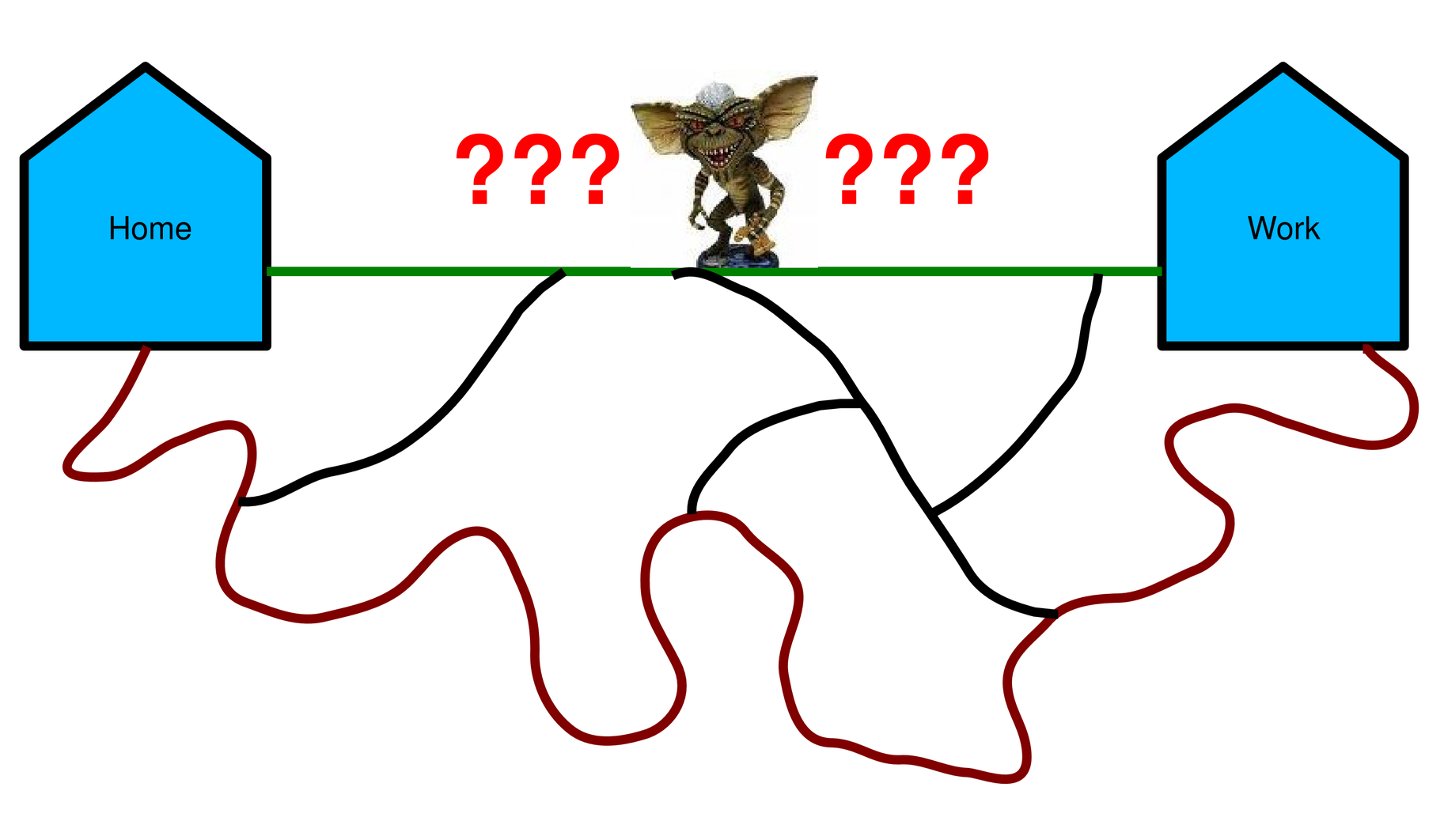
What should we do? If we are only taking the route once, we might go the more reliable but longer way. However, if we are going to use the route repeatedly, then we might try the short route to discover if the gremlin exists. Suppose we try the shorter route and are waylaid by the gremlin. Then we can update our probability distribution:



Through repeated trials we continuously update our beliefs about the world.

Temporal Credit Assignment: The effect of a decision may not become apparent until long after it has been made.

Suppose now that the paths are much more complicated:



Assume we eventually find our way to work. How do we know what decisions helped us get there, and what should we change to get there faster next time?

Reinforcement Learning in Computational and Biological Systems

Optimistic Active Learning

Marc Deisenroth (CUED), Carl Rasmussen (CUED), Jan Peters (MPI)

Decision-making under uncertainty

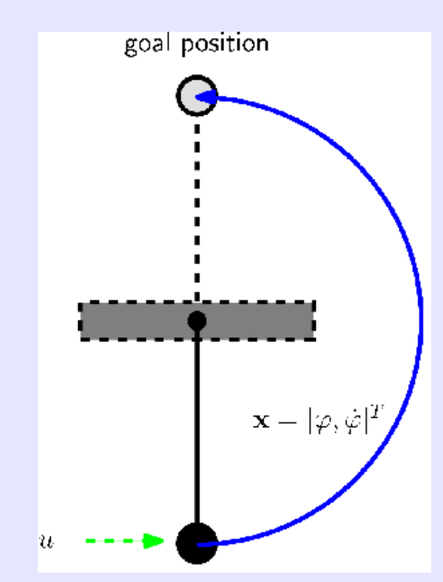
- unknown model of the world
- uncertain rewards

→ solution by combining **Bayesian methods** and **Bellman's optimality criterion**

goal of RL is identical to the one of **optimal control**: maximize long-term reward

→ application of RL-based controllers to dynamical systems possible

Control application: underpowered pendulum swing up



Task:
(1) pendulum hangs down
(2) swing it up
(3) balance it in inverted position
- system is **underpowered**: direct swing up impossible

Initial approach
- **known** dynamics
- **uncertain** rewards
- **continuous** state and action domains
  → function approximation by Gaussian processes
- support points **randomly** distributed in state space
- learn **global** model of close-to-optimal policy

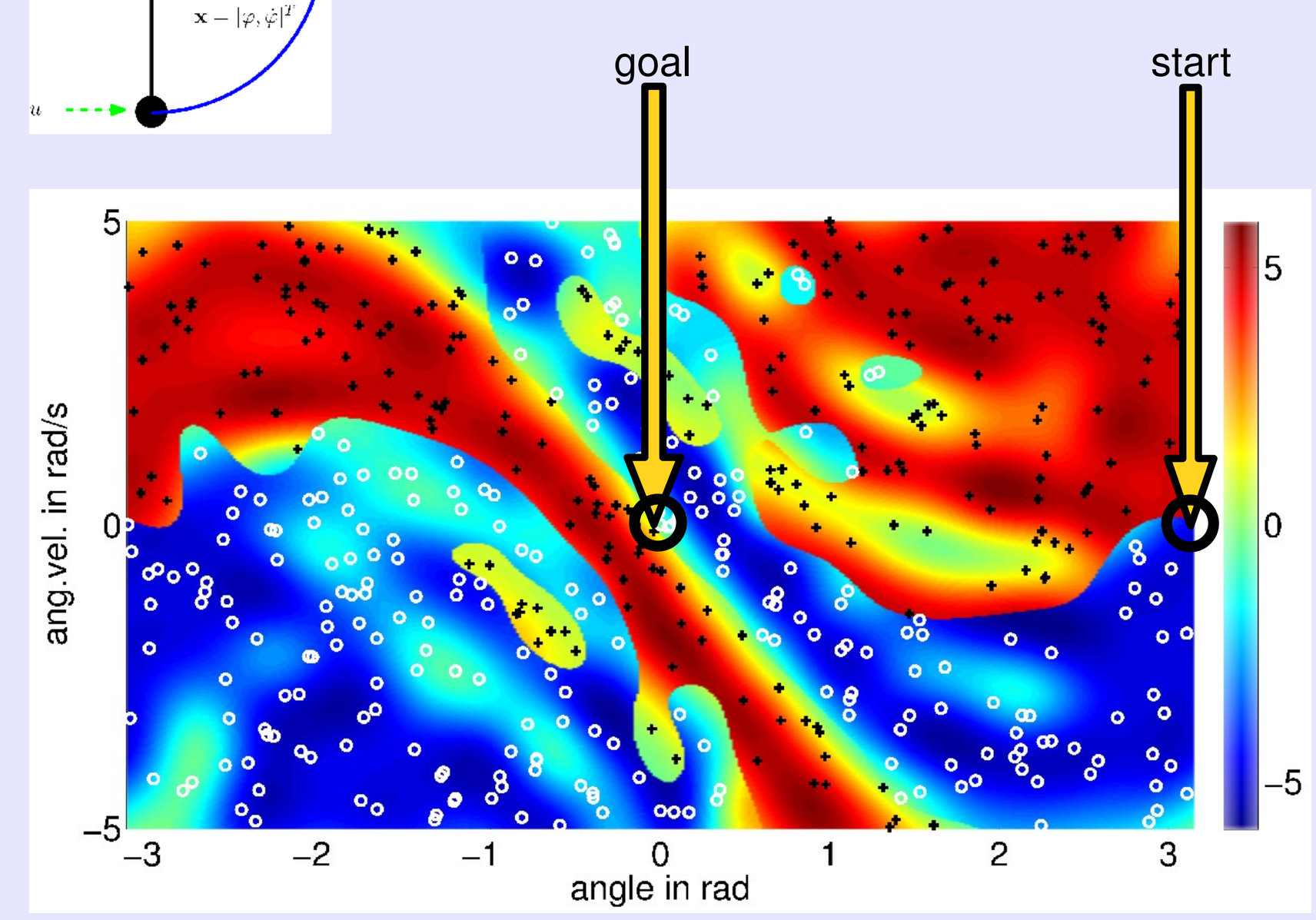


Figure: Support points and mean function of a Gaussian process for a globally optimal policy

- **random** placement of points in state space
- **global** (close-to) optimal solution

→ we can use **less points** if we are interested in **local** solutions

Relaxing the constraints

- **unknown** dynamics
- learn optimal policy **locally** on dynamically relevant manifold (where is this?)
- **explore** the world and **update** the models incrementally

Algorithmic idea:

(1) **model building (offline):** learn data-dependent model of unknown dynamics based on sampled trajectories
(2) **Bayesian active learning scheme** within initial approach: from few initial states around start point incrementally add new states in an optimistic manner
(3) **explore** paths from initial state to goal state
(4) start **exploitation**

Optimistic Bayesian active learning

objective:
- find a dynamically relevant manifold in state space
- build value function model (based on Gaussian processes)

add points in regions with
- expected **high reward**
- expected high **information gain**

→ **optimistic Bayesian active learning**

Approach:

- **sequentially** add states that maximize expected reward and expected information gain
- discover **later**, whether the added states were really good (**optimistic** approach: hope the best)
- **automatically** decide
  - where to explore
  - when to exploit
- automatically **detect** dynamically relevant manifold
- more **efficient** use of data than initial global solution

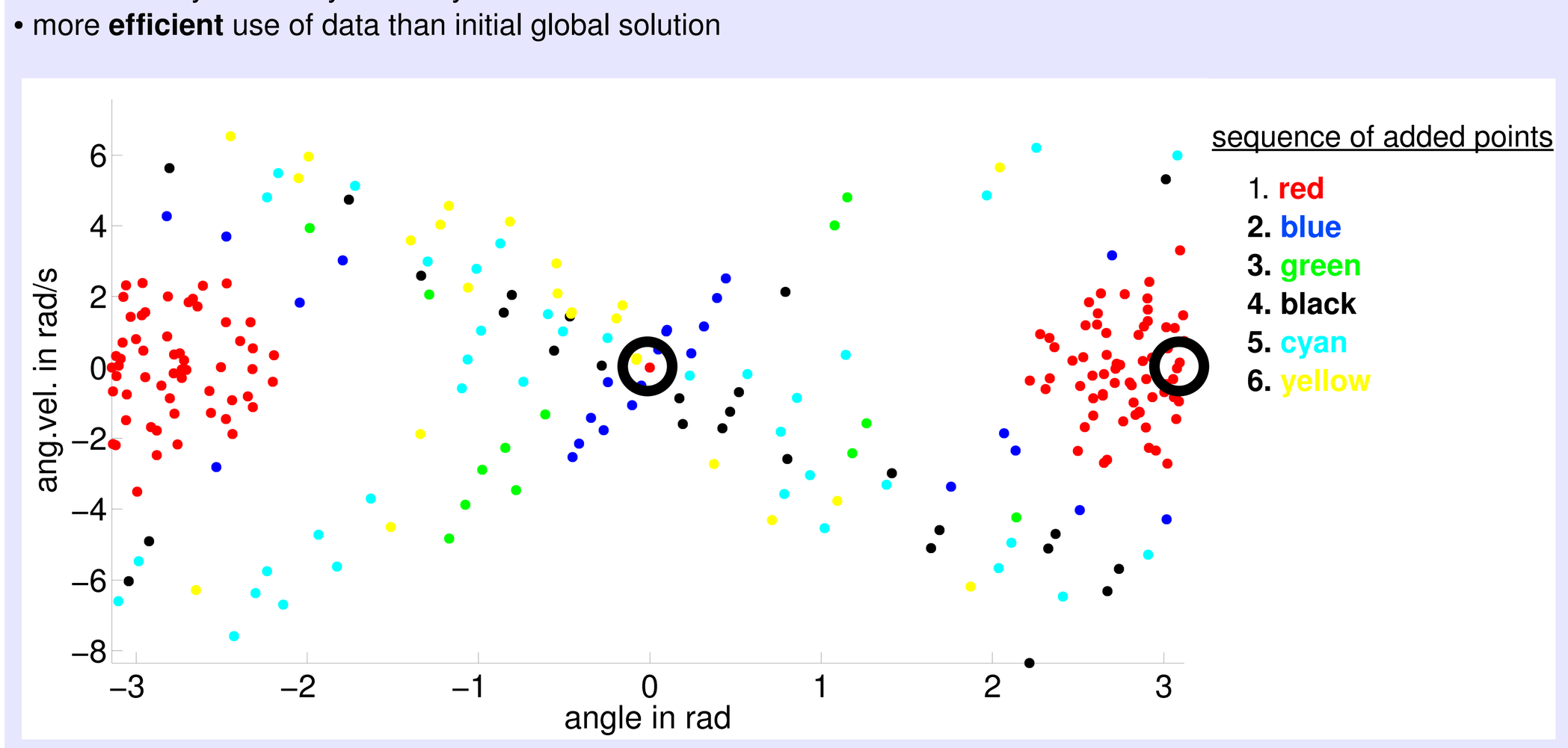


Figure: Sequentially added points using the Bayesian active learning scheme.

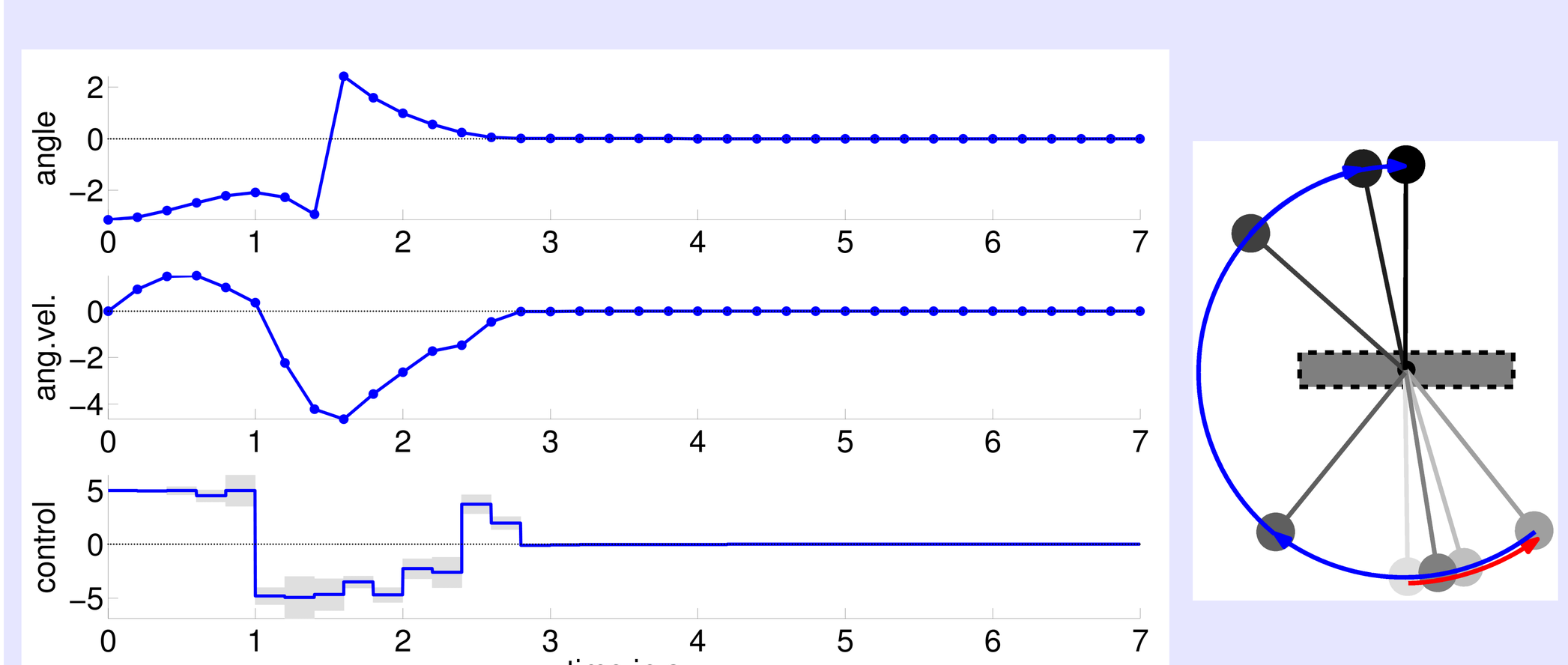


Figure: System trajectories when applying a Gaussian process controller based on the active learning scheme

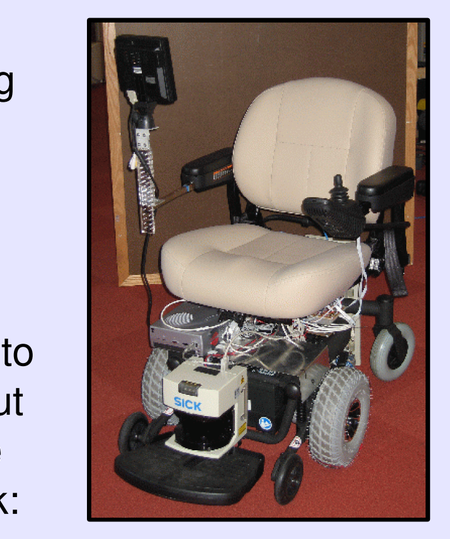
Summary

- reinforcement learning generalizes optimal control problem by considering unknown structures
- exploit sources of uncertainty to learn more about the world
- optimistic Bayesian active learning scheme automatically balances between exploration and exploitation
- locally optimal solution requires less data (more efficient) than globally optimal solution

Robust Active Learning

Finale Doshi (CUED), Nicholas Roy (MIT)

Sometimes the agent can't afford to make mistakes, even to learn—for example, consider a controller for a robotic wheelchair (right). We consider a scenario in which the agent needs to discover where the wheelchair user needs go using spoken commands, keeping in mind that taking someone to the wrong place can be quite annoying and disconcerting.



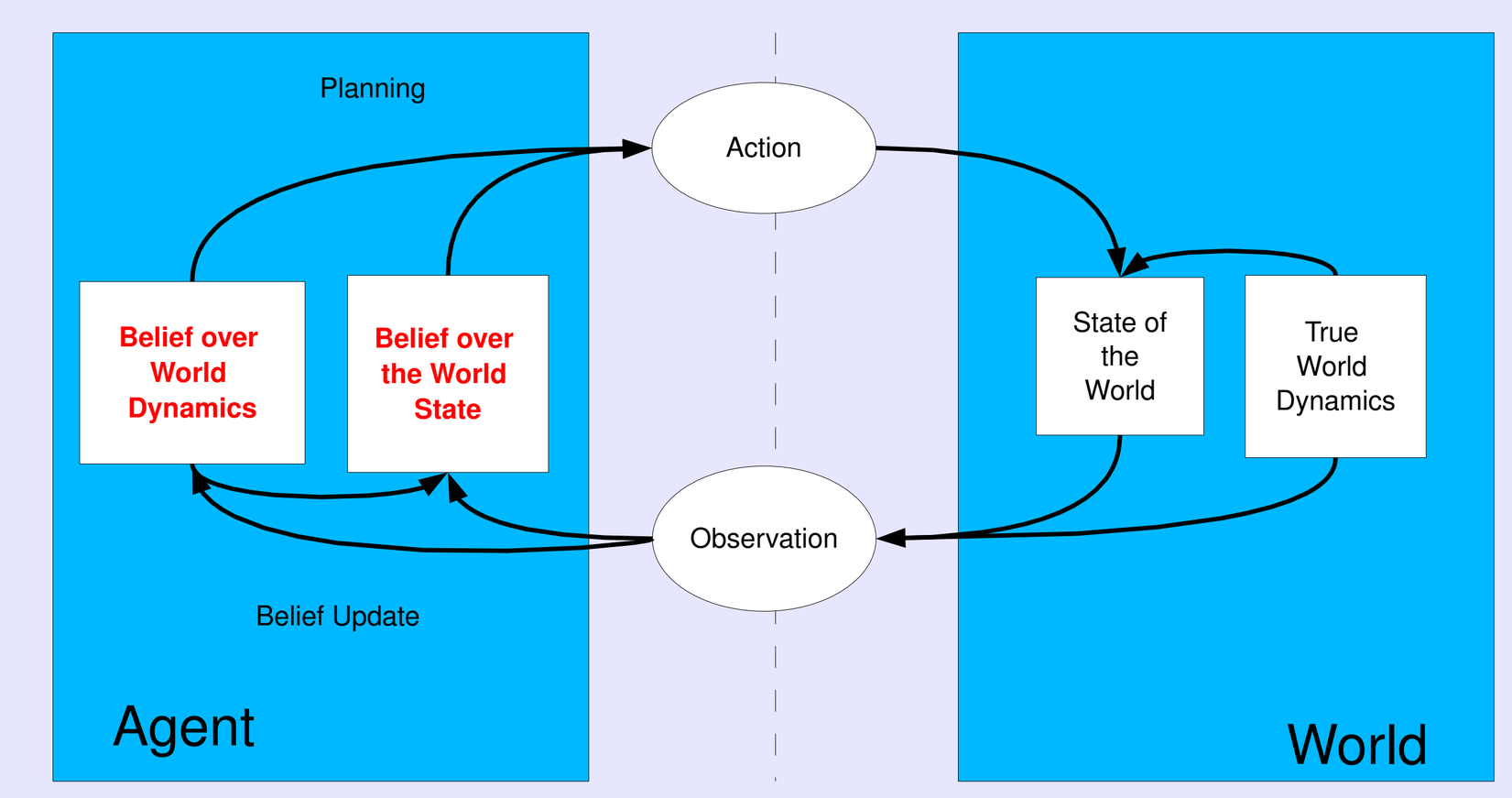
However, the agent can often ask for help. In most scenarios where robustness is needed, we can assume that there will be experts on hand to provide advice if needed. We assume this advice is policy information, or information about how the agent should act, because in many situations, it may be easier to specify what the agent should do rather than how it should be reasoning. For example, the agent might ask:

"I think you might want to go to the kitchen. Should I take you there?"

The model uses a planning framework known as partially observable Markov Decision Processes (POMDPs). POMDPs consist of:

- States: the true state of the world is hidden from the agent
- Observations: what the agent actually measures, which reflect the state of the world
- Actions: how the agent can interact with the world
- Probability distributions that govern the effects of taking each action in each state.

For the robotic wheelchair, the state of the "world" is where the user wishes to go, and it must be inferred from noisy utterances from the voice recognition system. The dynamics of the world, that is, the noise rate and the user's habits and preferences, must also be inferred. Accurate numerical reward feedback is often difficult to elicit from a user, so the agent must gather this information from the policy queries.



Algorithmic approach begins with a probability distribution over all possible states of the world. Reasoning takes place in two steps:
- Action Selection:
  - Finding the optimal action is intractable, so the agent considers the least risky action.
  - If the risk is too large, the agent asks for help.
- Belief Update: once the agent takes an action:
  - It receives an observation that changes its belief about what the user wants.
  - If it asked for help, it must incorporate this information into its model of the world.

Bayes' risk action selection: Let Q(b,a) the value—or expected long-term reward—of taking action a if the agent believes b. Then the loss of action a is defined to be:

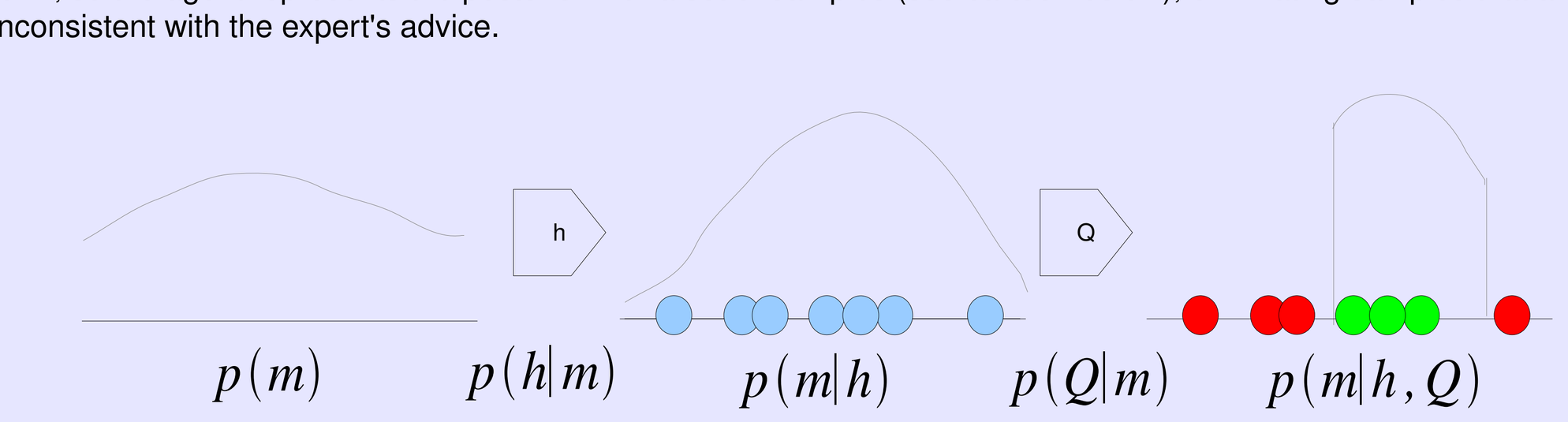
$$L(a)=Q(b,a)-Q(b,a')$$

where a' is the best action to take. The equation above assumes that the agent knows the value of taking a particular action. If instead the agent only has a probability distribution over models m (each with a different value function), then the risk of an action is its expected loss, and the least risky action is:

$$a=argmin_{a\in A}\sum_{M}(Q_m(b_m,a)-Q_m(b_m,a'_m))\,p(m)$$

If the risk is too large, the agent asks for help.

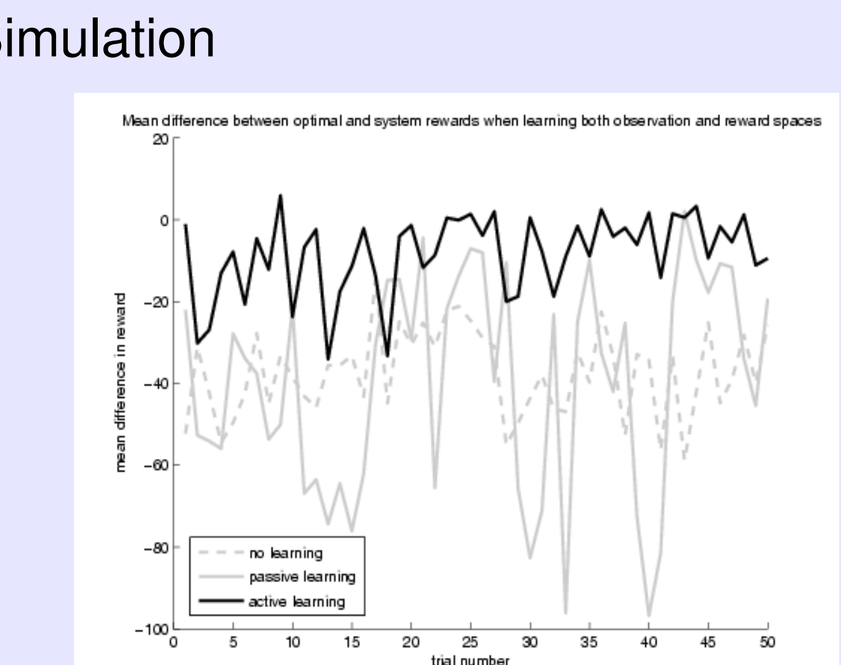
Belief update: the agent has two sources of information which it can use to update its belief over possible models of the world dynamics. First, it has a history h of actions and observations (for examples, the utterances that it hears from a voice recognition system). Under the right conditions, the agent can incorporate this information over world models. The second source of information is the set of queries Q. These cannot be incorporated in closed form; so the agent represents the posterior with a set of samples (see cartoon below), eliminating samples that are inconsistent with the expert's advice.



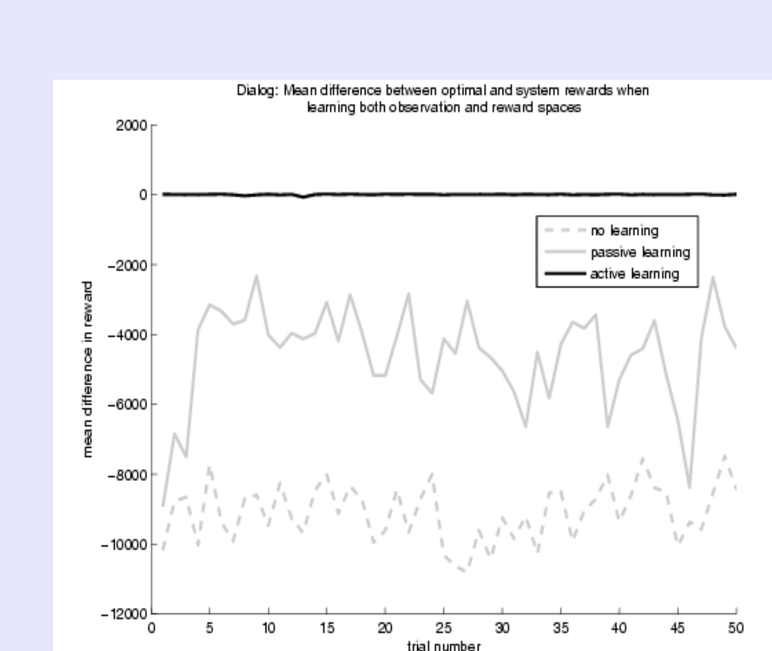
$p(m)$ $p(h|m)$ $p(m|h)$ $p(Q|m)$ $p(m|h,Q)$

Results. Below we show a simulation result that shows how our approach is robust from the very start—by asking questions, the agent avoids the learning curve. The tables below shows a sample dialogue with the robotic wheelchair, in which the agent learns the meaning of a new word in the presence of noisy voice recognition software.

Simulation



Performance of an agent that already has a good model of the world.



Performance of an agent that begins with only a very vague world model.

User study

Early conversation:
User: Give me the forecast.
Robot: I'm confused. What action should I take now?
<User indicates that the robot should provide the weather forecast>
Robot: Showers

Later conversation:
User: What's the forecast for today?
Robot: Do you want the weather?
User: Yup.
Robot: Showers

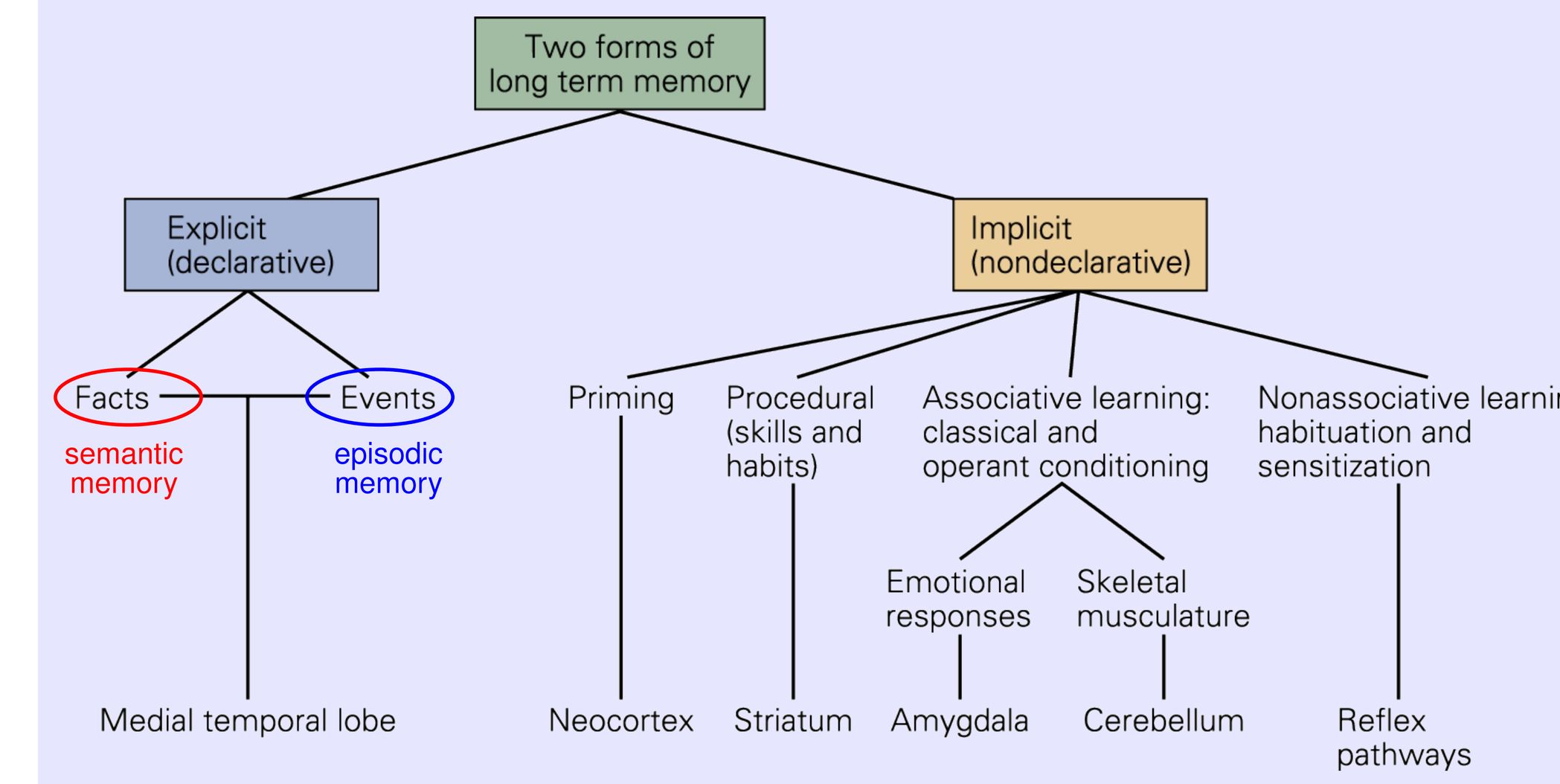
Early conversation:
User: I want some tea from the kitchen.
Robot: How can I help you?
User: The tea is in the kitchen near the coffee machine.
<The system hears "copy" instead of "coffee" and is confused about "tea" and "kitchen." >
Robot: How can I help you?
User: Go to the kitchen.
Robot: I'm confused. What action should I take now?
<The user indicates "Confirm the coffee machine area" is the correct action. >
Robot: Do you want to go to the coffee machine area?
User: Yup.
...
<The system asks a few more (non-meta) questions>
Robot: Do you want to go to the coffee machine area?
User: Yup.
Robot: Going to the coffee machine area.

Later conversation:
User: Take me to the kitchen. *<The system hears "Nick" and "kitchen.">*
Robot: How can I help you?
User: I want some coffee. *<The system hears "copy" instead of "coffee.">*
Robot: Do you want to go to the coffee machine area?
User: Yup.
Robot: Going to the coffee machine area.

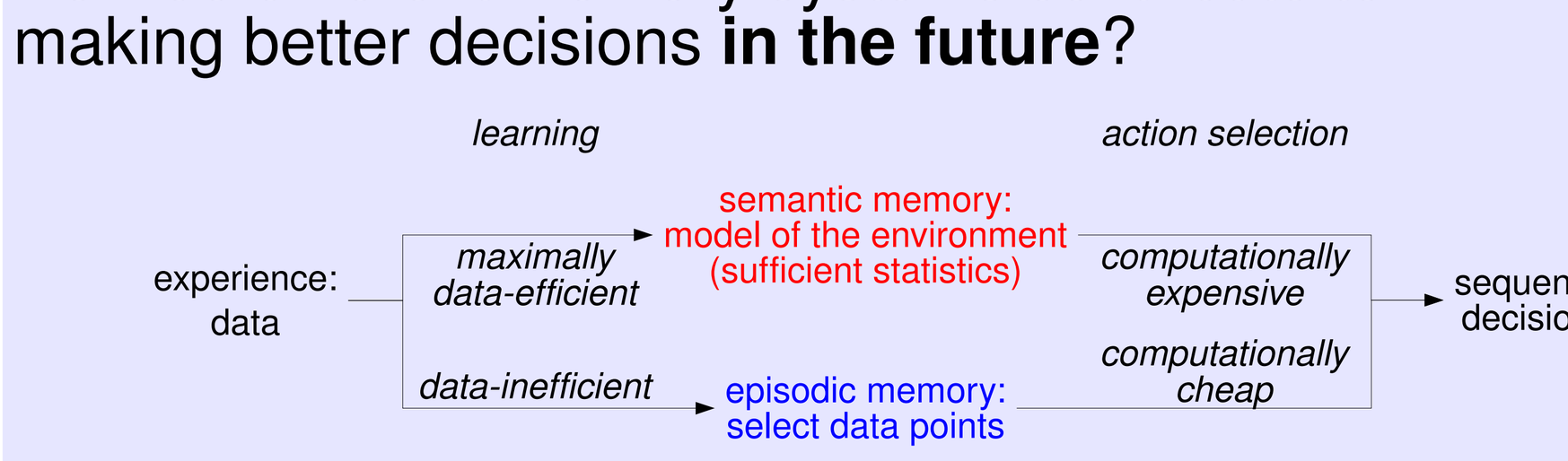
Memories for Decisions

Máté Lengyel (CUED), Peter Dayan (Gatsby Unit, UCL)

Biological memory: a multitude of systems



How do different memory systems contribute to making better decisions **in the future**?



Q: why store episodes *even if semantic learning is perfect*?
A: useful for sequential decision making with computational noise.

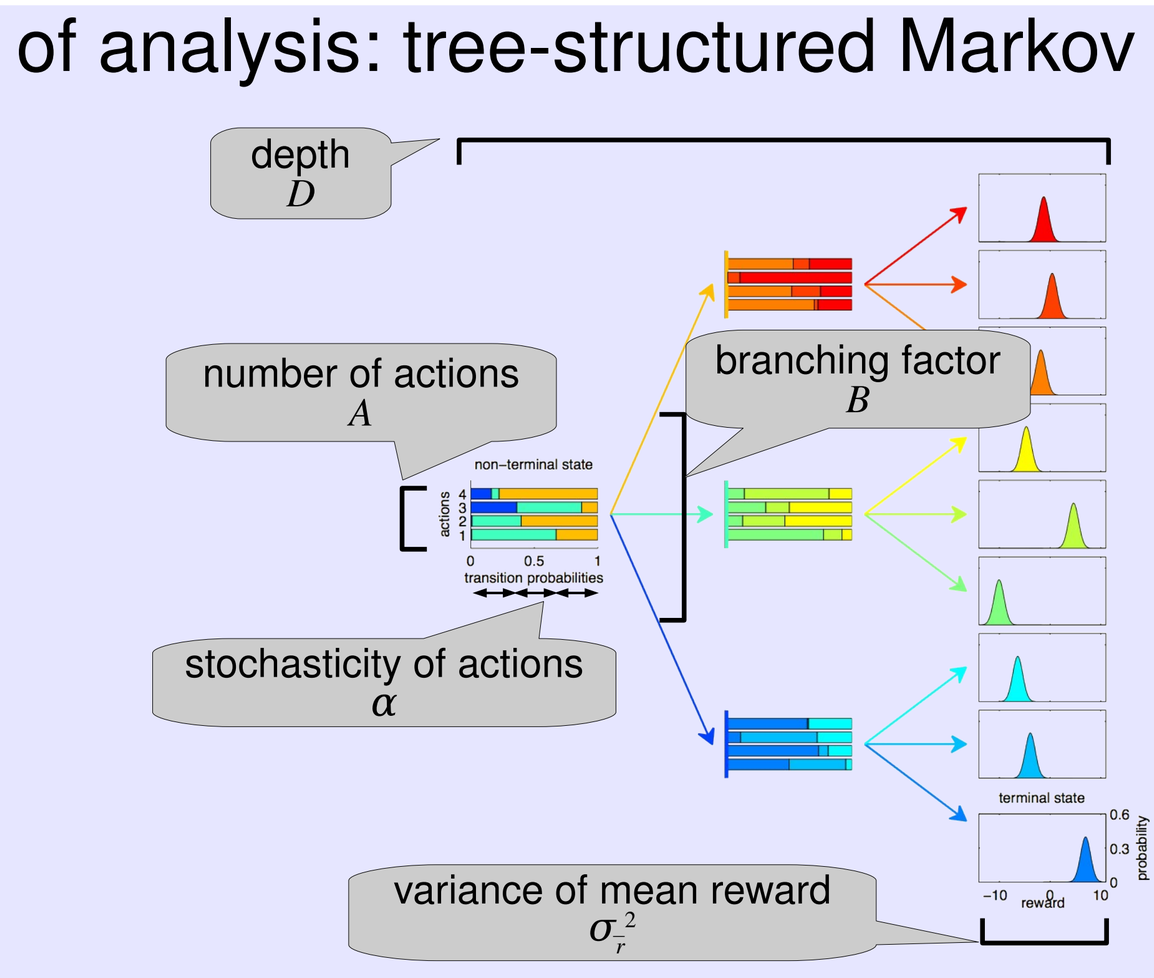
Semantic memory: model-based control

- learns a model of the environment (Bayesian updating of transition and reward probabilities)
- selects actions by **recursive** 'mental simulation' (online DP)
  - → tree search with **combinatorial explosion**
  - → **approximations** are necessary
    ↓
    computational noise!

Episodic memory: off-policy Monte Carlo control

- stores specific episode **retrospectively** (state-action-...-reward sequence), when reward exceeds expectations
- selects action that yielded maximal reward from given states in past episodes

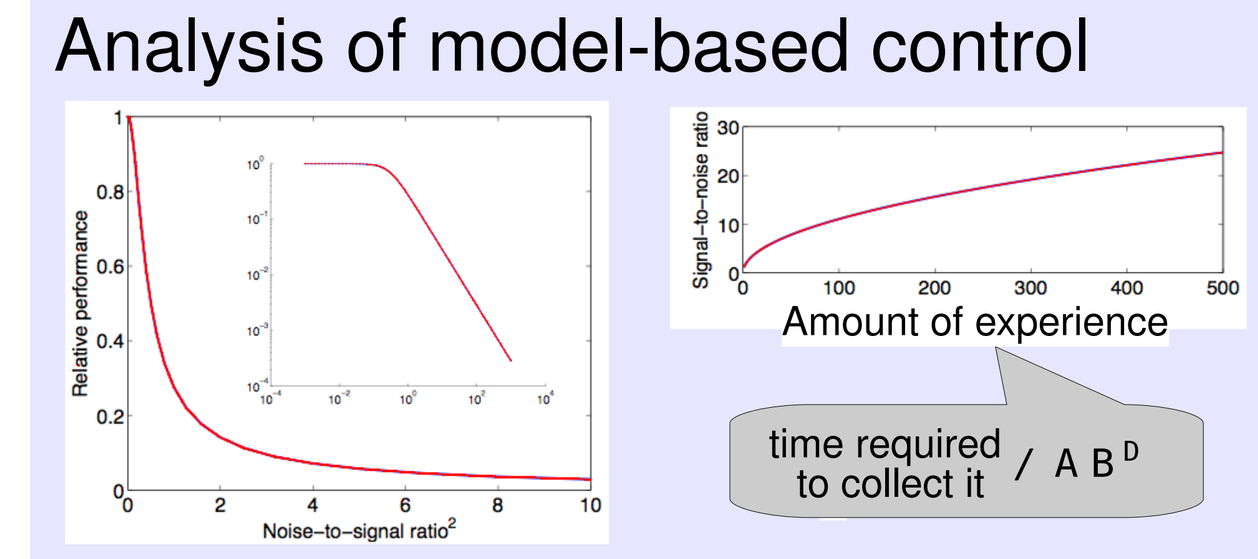
Domain of analysis: tree-structured Markov decision processes



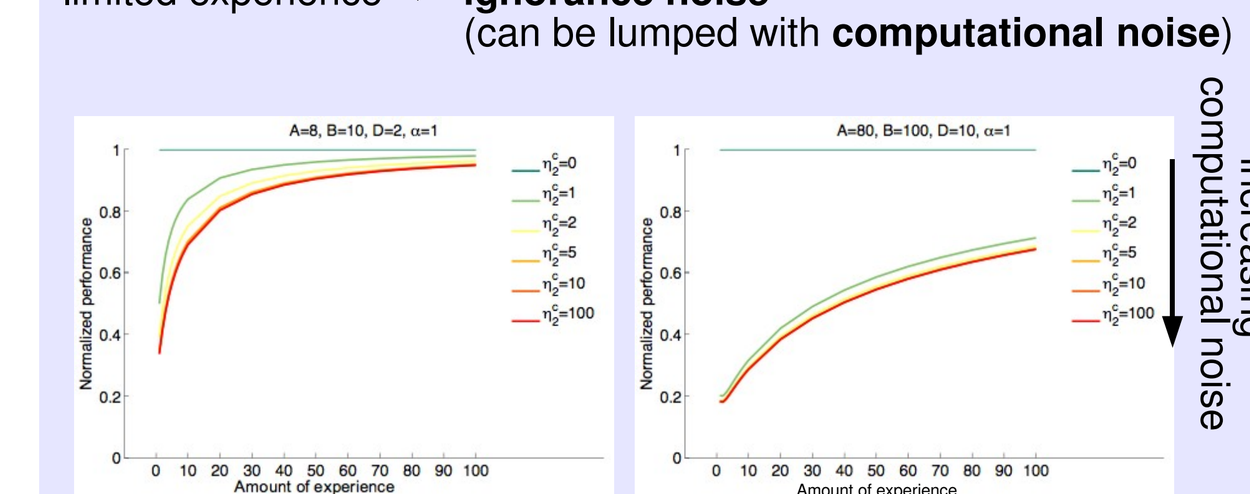
Episodic vs. semantic memory-based control

- vary complexity of the environment (through hyper parameters)
- vary amount of experience available to semantic and episodic memory systems
- compute average performance of three controllers (over distribution of environments)
  1. perfect model-based control → theoretical upper bound
  2. noisy model-based controller → semantic memory
  3. off-policy Monte Carlo controller → episodic memory

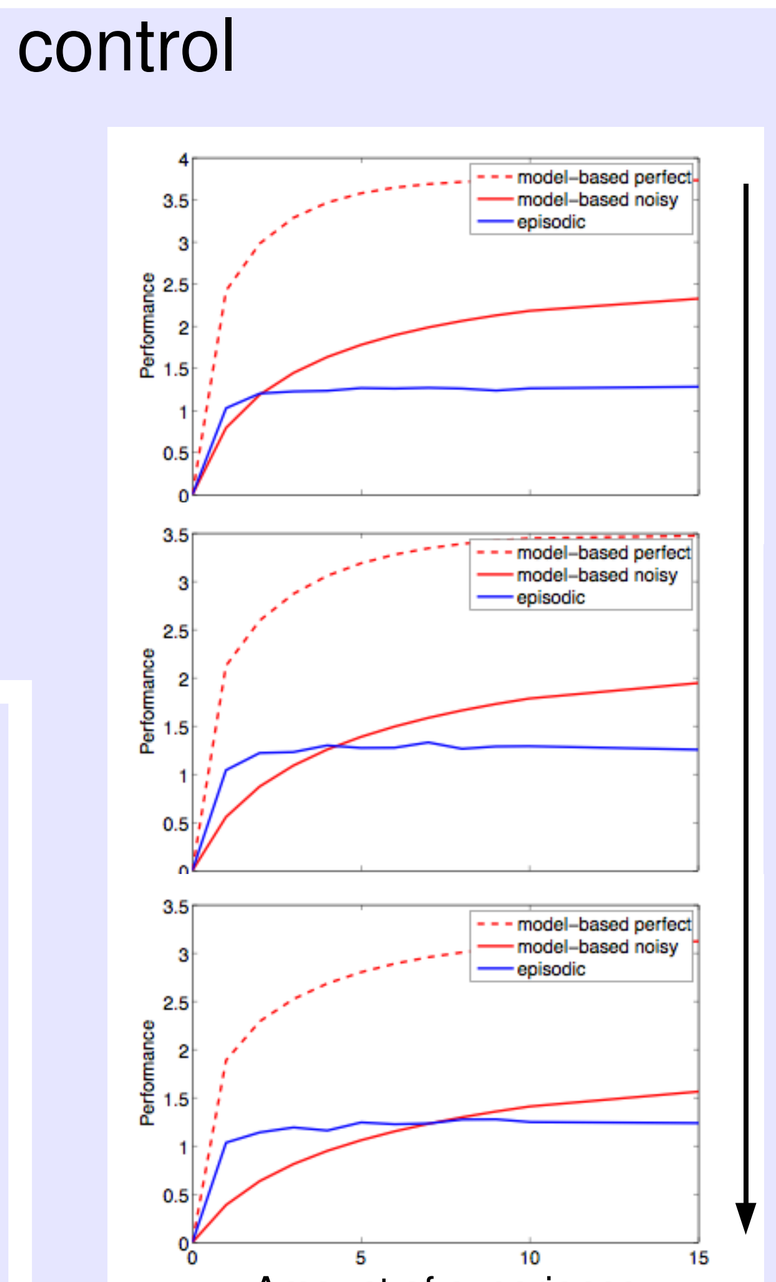
Analysis of model-based control



limited experience → **ignorance noise** (can be lumped with **computational noise**)



- **computational noise more adverse in learning**



- **episodic advantage early in learning**
- **lasts longer for more complex environments**

Summary