

# AA203

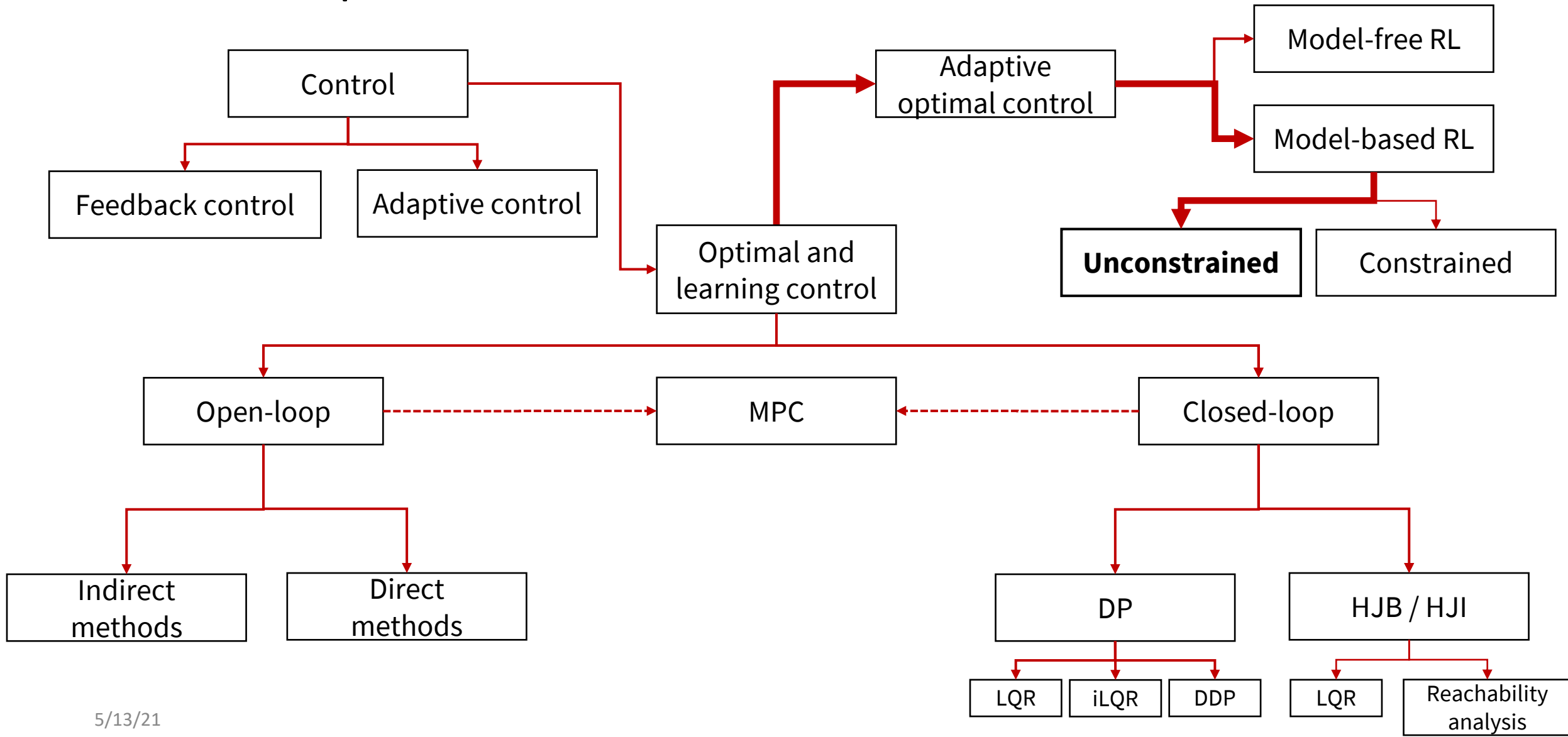
# Optimal and Learning-based Control

Model-based reinforcement learning

# Logistics

- Grading midterm reports
- Homework 3 out now: start early!
- 2/3 term survey out today, due Sunday

# Roadmap

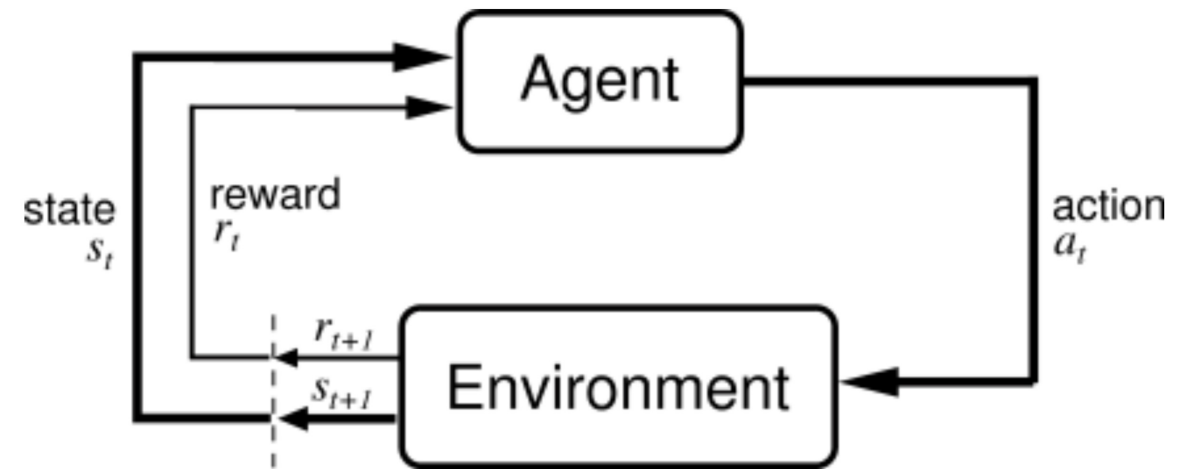


# Agenda

- Reviewing the reinforcement learning problem statement
- Tabular model-based RL
- Continuous model-based RL
- Readings:
  - M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, *Bayesian Reinforcement Learning: A Survey*, Foundations and Trends in ML, 2016.
  - R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*, 2018.
  - M. Kochenderfer. *Decision Making Under Uncertainty*, 2015.
  - T. M. Moerland, J. Broekens, C. M. Jonker. *Model-based Reinforcement Learning: A Survey*, 2020
  - K. Chua, R. Calandra, R. McAllister, and S. Levine. *Deep reinforcement learning in a handful of trials using probabilistic dynamics models*, NeurIPS, 2018.

# Reinforcement learning assumptions and information patterns

- Previously:
  - Intro model-free RL: Q-learning, SARSA
    - Multiple episodes, interleave data collection and policy improvement
    - Tabular (exact)
  - System Identification
    - Batch, offline data collection
    - Primarily linear dynamics
  - Adaptive Control
    - Intra-episode/online adaptation
    - Primarily linear dynamics



# How are these different?

- Short answer: different historical developments, thus different standard assumptions
- Typically not clearly stated
- Thus difficult to:
  - connect similar ideas in different fields
  - know what works, and when
  - know what you should use for your problem
- Currently, all are increasingly overlapping

# Core assumptions

- Linear vs. nonlinear dynamics
- Known vs. unknown cost function
- Episodic interaction vs. single episode (online) vs. batch offline data
  - Related: which policy was used to collect data (Offline vs online)?

# Breaking down assumptions

	System Identification	Adaptive Control	Model-based RL	Model-free RL
<b>Dynamics</b>	Usually linear	Linear or nonlinear, usually control affine	Discrete or nonlinear continuous	Discrete or nonlinear continuous
<b>Reward knowledge?</b>	N/A	Designed (thus known)	Typically assumed known (not always)	Typically assumed unknown, provided by environment
<b>Data collection/ episodic structure</b>	Dataset provided	One episode	Typically, repeated episodes	Typically, repeated episodes
<b>What do we learn?</b>	Dynamics model	Usually policy (MRAC) or model (MIAC)	Dynamics model, sometimes reward model, sometimes policy	Policy (or Q function)

Caveat: there are exceptions to **all** of the above.



# Unconstrained stochastic control problem

$$J_0^*(\mathbf{x}_0) = \min_{\boldsymbol{\pi}_0, \dots, \boldsymbol{\pi}_{T-1}} \mathbb{E} \left[ p(\mathbf{x}_T) + \sum_{k=0}^{T-1} c(\mathbf{x}_k, \boldsymbol{\pi}_k(\mathbf{x}_k)) \right]$$

subject to  $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \boldsymbol{\pi}_k(\mathbf{x}_k), \mathbf{w}_k, \boldsymbol{\theta}) \quad k = 0, \dots, N - 1$

$\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$

$\mathbf{w}_k \sim p(\mathbf{w})$  iid,  $k = 0, \dots, N - 1$

# Generalization and exploration

- LTI dynamics: if dataset generated with sufficient excitation, gives **global** knowledge
- Nonlinear dynamics: extrapolation is difficult and can be misleading
  - As AC/RL moves to more complex systems, have to consider uncertainty, exploration, and data collection process



# Tabular model-based RL

- Discrete state/action space with stochastic transitions
- If model is known, can use value iteration/policy iteration/etc.
- Model unknown: want to build approximate model from observed transitions

# Tabular MBRL outline

- Assume initial policy
- Loop forever:
  - Take some number of actions, resulting in transition/reward data
  - Improve dynamics model
  - Choose actions/policy
- Approaches for action selection:
  - Dynamic programming/VI/DP on approximate model
    - Expensive, gives optimal policy for model
  - Plan suboptimal sequence of actions via online control optimization

# Dynamic programming for action selection

- Given an updated model, can perform value iteration/DP to yield new policy.
  - Can be very expensive for large MDPs!
  - Effect of local model changes (often) has minor impact on far away states.

# Local methods for action selection

- Tree search methods:
  - Similar idea to MPC: continuously generate short plans to approximate closed-loop policy.
  - For example, Monte Carlo tree search (MCTS):
    - Sample random action sequences
    - Choose best sequence and execute first action

# Combining local and global methods

- Many ways to combine local search (e.g. MCTS) with global/dynamic programming methods
  - Can use (possibly old) running value estimate as tail value in search
  - Forward search gives a TD update for value

# Learning a tabular model from data

- States ( $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ )
- Actions ( $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m$ )
- Want to learn  $p(\mathbf{x}_i | \mathbf{x}_j, \mathbf{u}_k)$  for all  $i, j, k$
  
- We will discuss both max likelihood point estimation and fully Bayesian approaches



# Max likelihood for tabular MBRL

- Categorical likelihood:  $p(\mathbf{x}_i | \mathbf{x}_j, \mathbf{u}_k, \boldsymbol{\theta}) = \boldsymbol{\theta}_{ijk}$
- Assume data  $D = \{(\mathbf{x}, \mathbf{u}, \mathbf{x}')\}_{i=1}^d$

- Max likelihood:

$$\max_{\boldsymbol{\theta} \in \Theta} \sum_D \log p(\mathbf{x}' | \mathbf{x}, \mathbf{u}, \boldsymbol{\theta})$$

- Gives MLE  $\boldsymbol{\theta}_{ijk} = N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i) / N(\mathbf{x}_j, \mathbf{u}_k)$   
where  $N(\cdot, \cdot)$  is the empirical count

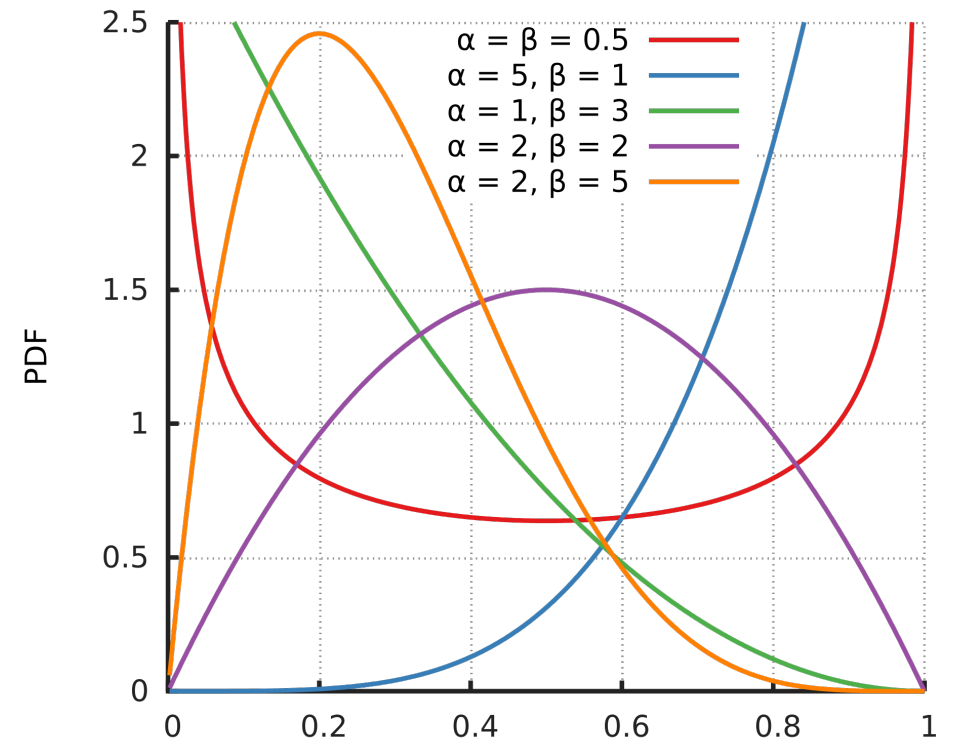
# Example: coin toss

# Max likelihood for tabular MBRL

- $\theta_{ijk} = N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i) / N(\mathbf{x}_j, \mathbf{u}_k)$
- Problem: what if  $N(\mathbf{x}_j, \mathbf{u}_k) = 0$ ?
  - For example, if we are starting with zero information, this model estimation scheme breaks
- Simple solution:
  - Store  $N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i)$ ; note that  $N(\mathbf{x}_j, \mathbf{u}_k) = \sum_{\mathbf{x}_i} N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i)$
  - Replace  $N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i)$  with  $N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i) + 1$
  - Gives  $\theta_{ijk} = (N(\mathbf{x}_j, \mathbf{u}_k, \mathbf{x}_i) + 1) / (N(\mathbf{x}_j, \mathbf{u}_k) + n)$
- We will see: corresponds to weak prior over transition mass function

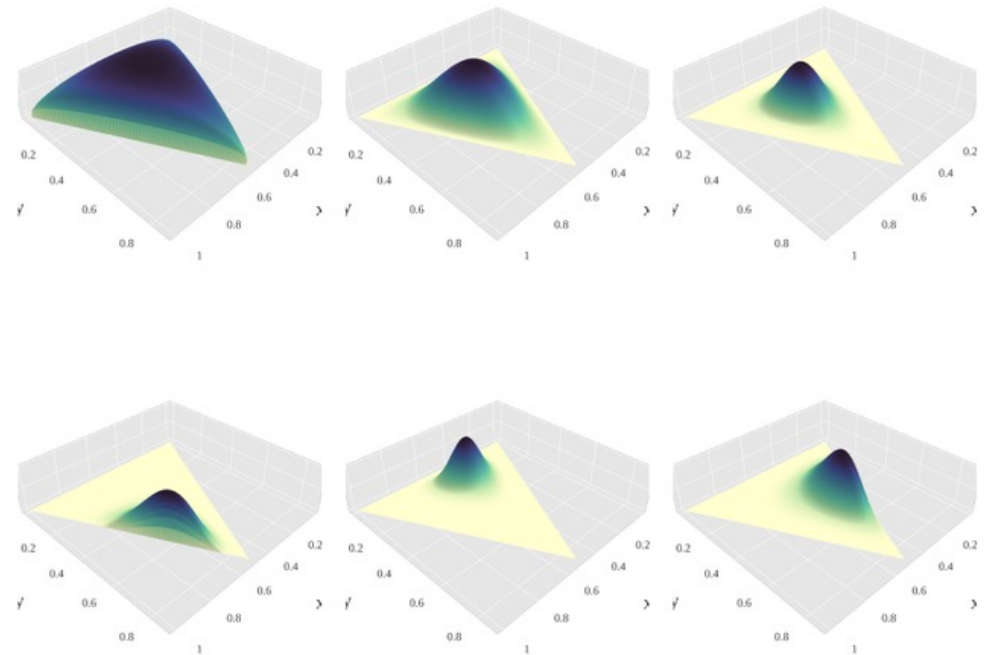
# Bayesian inference for categorical distribution

- Beta distribution



# Bayesian inference of transition probabilities

- Fix *Dirichlet distribution* prior
  - Corresponds to a probability distribution over discrete probability distributions
  - Write  $\text{Dir}(X, \alpha) = p(\mathbf{x}_1, \dots, \mathbf{x}_n | \alpha_1, \dots, \alpha_n)$
- Dirichlet is *conjugate* with categorical distribution:
  - Dirichlet prior with parameters  $\alpha_1, \dots, \alpha_n$ , plus Categorical distribution gives Dirichlet posterior  $\text{Dir}(X, \alpha + c)$
  - $c = (c_1, \dots, c_n)$  is counts of data



For details on derivation of posterior, see: *The Dirichlet-Multinomial and Dirichlet-Categorical models for Bayesian inference*, Stephen Tu (available online).

# Bayesian posterior

- Prior parameter  $\alpha$  corresponds to number of prior observations
- For  $(\mathbf{x}_1, \dots, \mathbf{x}_n) \sim Dir(\alpha)$ ,  $E[\mathbf{x}_i] = \alpha_i / \sum_j \alpha_j$ 
  - Posterior predictive is  $p(\mathbf{x}' | \mathbf{x}, \mathbf{u}, D, \alpha) = \frac{N(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \alpha'(\mathbf{x}, \mathbf{u}, \mathbf{x}')}{\sum_{\mathbf{x}'} N(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \alpha'(\mathbf{x}, \mathbf{u}, \mathbf{x}' )}$
- Choose  $\alpha = (1, \dots, 1)$  gives our previous correction
- But, we have more than just the point estimate of our model. How can we use this?

# Bayes-adaptive MDPs

- Have model parameters  $\theta$ , which summarizes “counts” for Dirichlet posterior for every state/action/next state combination.
- *Bayes-adaptive* MDP, with hyperstate  $(\mathbf{x}, \theta)$ 
  - So have transition dynamics  $p(\mathbf{x}', \theta' | \mathbf{x}, \mathbf{u}, \theta)$
  - Factor as  $p(\mathbf{x}' | \mathbf{x}, \mathbf{u}, \theta) p(\theta' | \mathbf{x}', \mathbf{x}, \mathbf{u}, \theta)$ 
    - $p(\mathbf{x}' | \mathbf{x}, \mathbf{u}, \theta) = \frac{N(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \alpha'(\mathbf{x}, \mathbf{u}, \mathbf{x}')}{\sum_{x'} N(\mathbf{x}, \mathbf{u}, x') + \alpha'(\mathbf{x}, \mathbf{u}, x')}$
    - Model parameter count increases by 1 for corresponding transition count
- Problem: state space grows infinitely, so can not do dynamic programming

# Exploration heuristics: Thompson sampling

- Even in tabular and linear MDPs, dual control/Bayes-adaptive MDP intractable
- So turn to heuristics to explore
  - Epsilon greedy, noise addition (persistent excitation)
- Simple approach using posterior over models: Thompson sampling
  - Sample MDP from posterior
  - Act optimally w.r.t. this MDP for episode
  - Update model posterior and loop



# Continuous MBRL

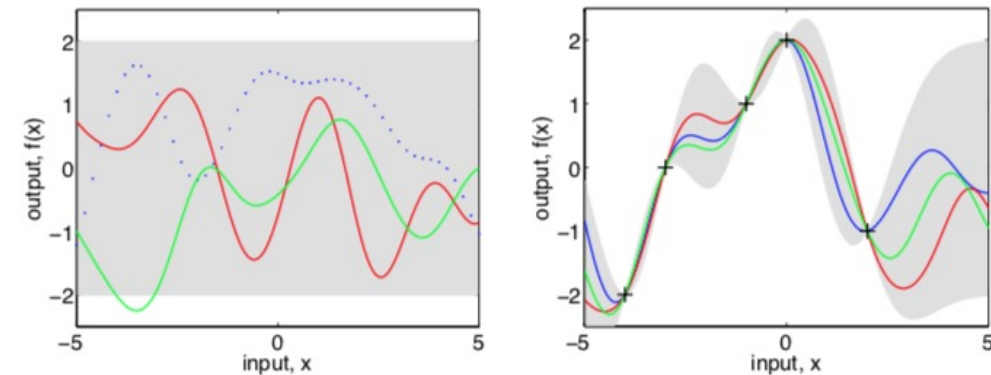
- Will now consider general non-LTI continuous models
- Many possible model choices:
  - Nonlinear features in linear regression
  - Time varying linear dynamics
  - Gaussian processes
  - Neural networks
- Many possible control choices:
  - MPC (gradient-based vs. sampling, with/without final cost)
  - Direct methods (e.g. iLQR/DDP)
  - Directly optimize policy (next week)



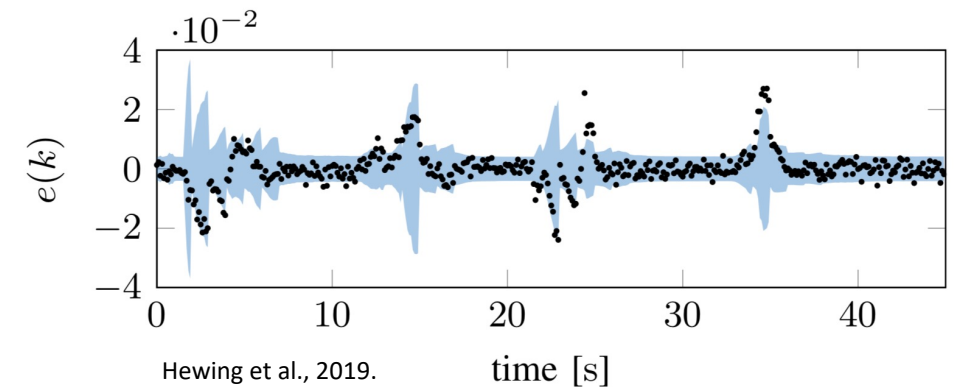
Abbeel et al., NeurIPS 2008

# Gaussian process models

- Place prior over dynamics,  $f \sim GP(m(\cdot), k(\cdot, \cdot))$ 
  - Corresponds to infinite dimensional gaussian distribution, prior over functions
- Strengths
  - Data efficient
  - Exact posterior
  - Predictable behavior via kernel choice
- Weaknesses
  - High computational complexity
  - Assume Gaussian measurement error
  - Can not learn expressive features



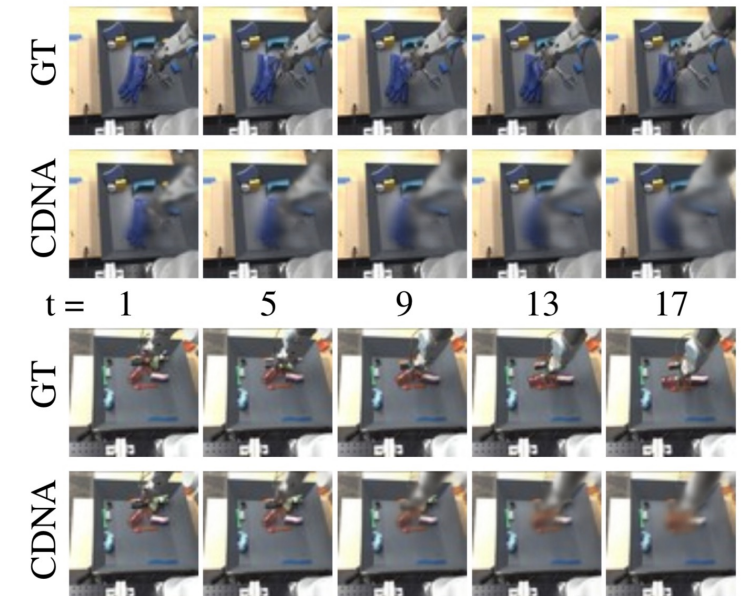
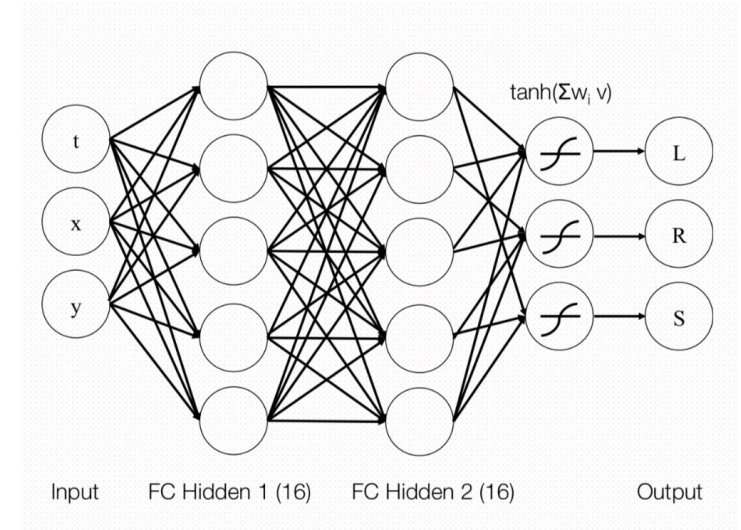
Rasmussen and Williams, 2006.



Hewing et al., 2019.

# Neural network models

- Parameterize model using neural network
- Strengths
  - Can learn complex, expressive features
  - Can be paired with arbitrary loss functions
- Weaknesses
  - Data inefficient
  - Difficult to represent uncertainty
  - Unpredictable extrapolation



Finn et al., 2017.

# Case study: PETS

- Key idea:
  - Use *ensemble* (collection) of NNs to approximate posterior over model
  - Incorporate model uncertainty into control
- Ensembling:
  - Initialize several networks with different weights
  - Will agree where there is a lot of data, disagree elsewhere

---

## Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

---

Kurtland Chua

Roberto Calandra

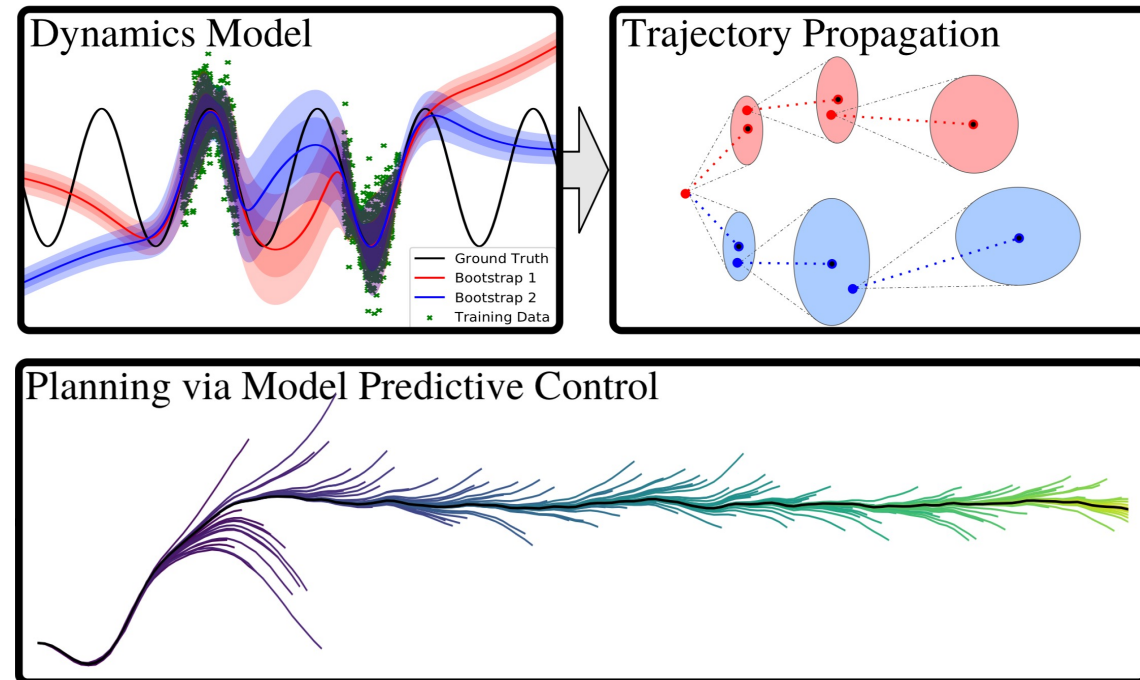
Rowan McAllister

Sergey Levine

Berkeley Artificial Intelligence Research

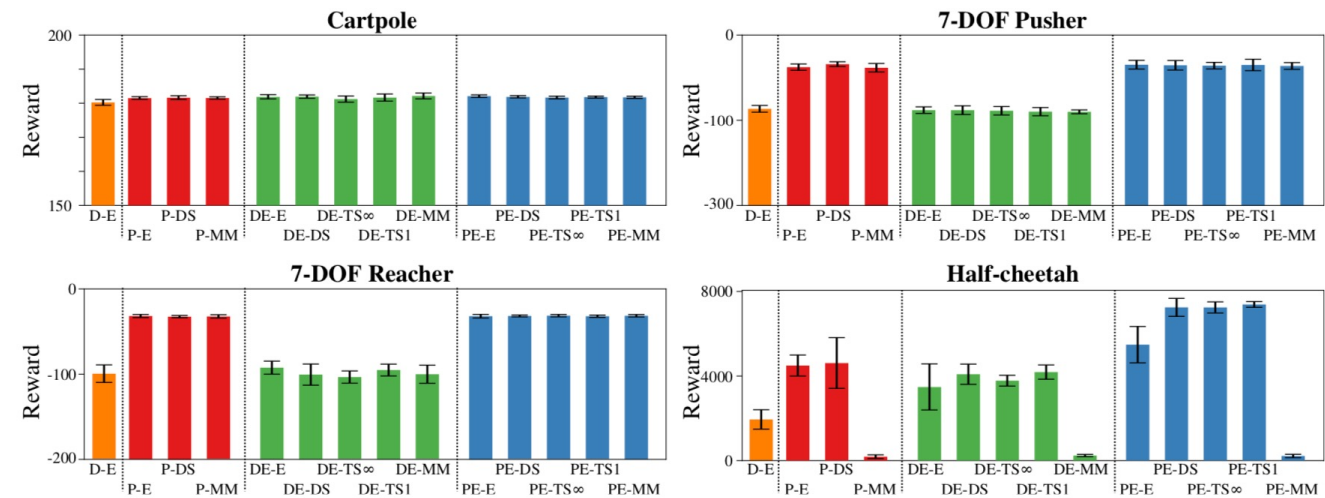
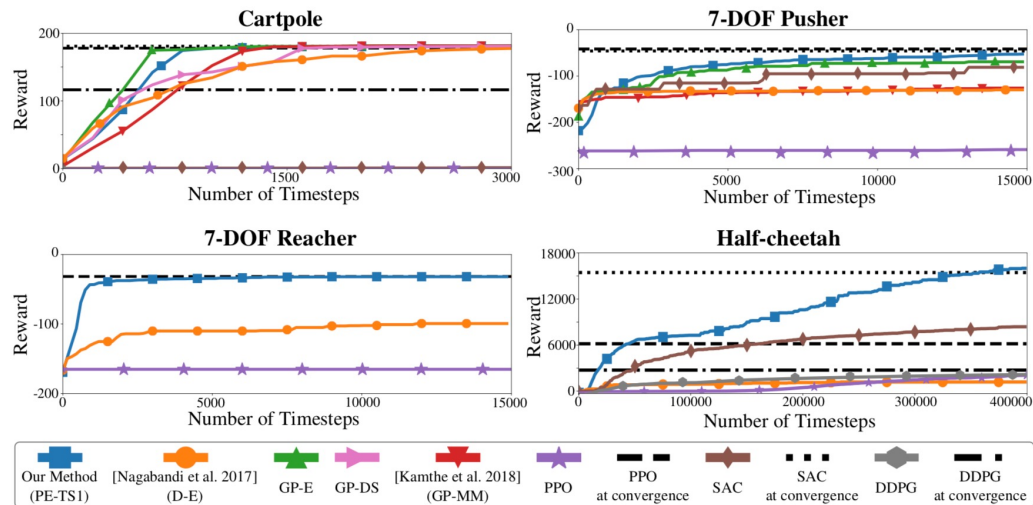
University of California, Berkeley

{kchua, roberto.calandra, rmcallister, svlevine}@berkeley.edu



# PETS findings

- Consider both probabilistic network (outputs mean + variance) and deterministic
- Use particle-based MPC controller (random action sampling)
- Either re-sample dynamics at each time, or keep fixed



# Why model-based?

- Advantages
  - Transitions give strong signal
  - Data efficiency, improved multi-task performance, generalization
- Weaknesses
  - Optimizing the wrong objective
  - May be very difficult/intractable for systems with high dimensional observations/states

# Next time

- Model-free RL: policy gradient, variance reduction, actor-critic.