Real-World Application: Uncertainty in Model-based RL

Christopher Mutschler

Outline

- Motivation: why model-based RL?
- What is a model? What are its inputs? What is a good model?
- How can we use a model?
  - Background Planning
    - Environment data augmentation / simulation
    - Sample efficient policy learning
  - Online Planning
    - Discrete Actions
    - Continuous Actions
  - Auxiliary tasks
- **Real-world application**

Real-world application

Problem: Model is never perfect

- Aleatory/process uncertainty (risk): the process itself has noise
- Epistemic/Model uncertainty: our model is not perfect

Solutions:

A. Act under imperfect models: use online re-planning (Model Predictive Control)

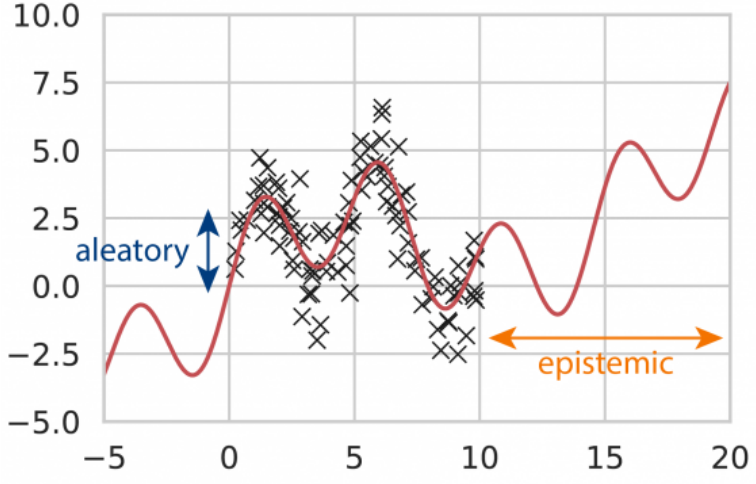
B. Estimate model uncertainty and use it for safe and efficient planning

C. Combine A and B

Model Uncertainty

We will always have model errors

- Aleatory/process uncertainty (risk): the process itself has noise
- Epistemic/Model uncertainty: our model is not perfect



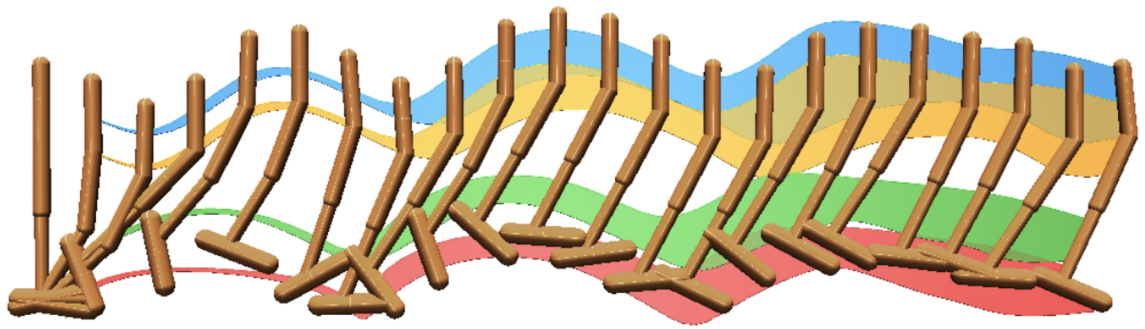
https://www.inovex.de/blog/uncertainty-quantification-deep-learning

Model Uncertainty

We will always have model errors

- Aleatory/process uncertainty (risk): the process itself has noise
- Epistemic/Model uncertainty: our model is not perfect
- Model errors are additive: the prediction error will become larger as we try to predict further into the future

  → Small errors propagate and compound
  - Planner might even exploit such model errors!
  - Longer rollouts are less reliable



Janner et al (2019). When to Trust Your Model: Model-Based Policy Optimization.

Model Uncertainty

Model Calibration

- Big neural networks exhibit superior predictive power
  - But are not well calibrated in practice!
- Calibration affected by
  - depth & width of the network (the bigger the more overconfident)
  - weight decay & batch normalization
- Most calibration methods are post-hoc methods
  - Such methods do not in RL settings!

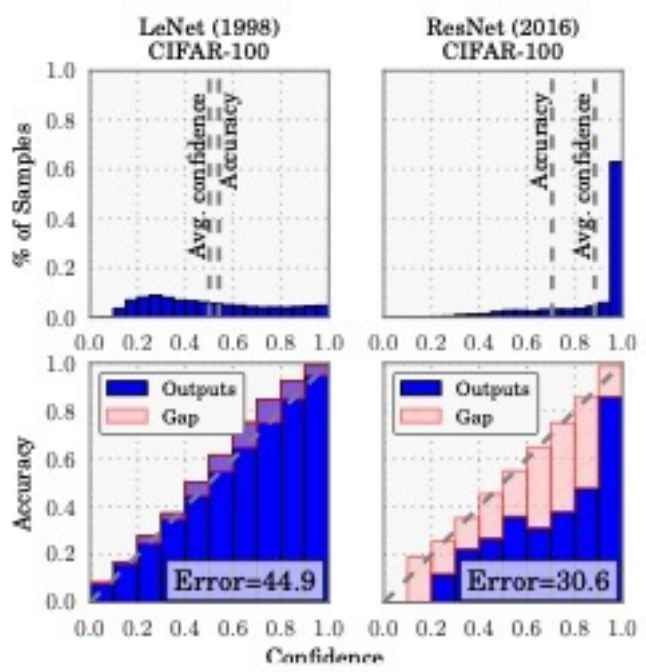


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

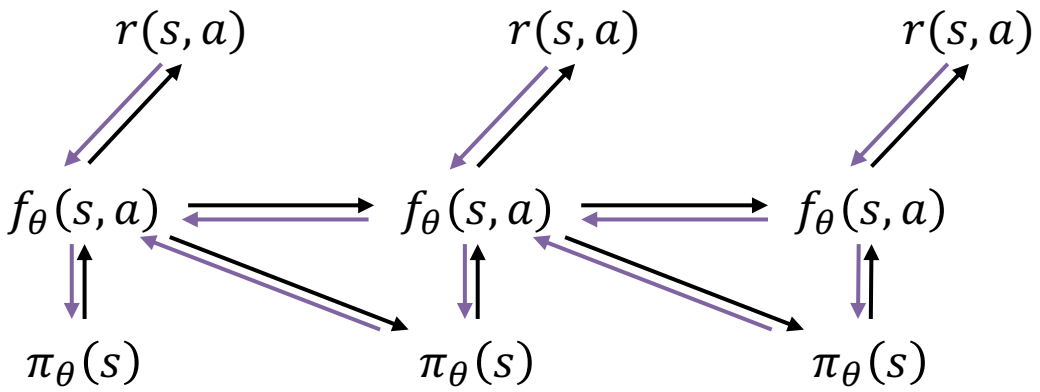
Chuan Guo et al.: On Calibration of Model Neural Networks. ICML. 2017.

Model Uncertainty

Recap: Background Planning with Policy Backpropagation

Better Algorithm:

1. Run a base policy $\pi_0(a_t|s_t)$ (e.g., a random policy) to collect data samples $\mathcal{D}\{(s, a, s')_i\}$
2. Learn a dynamics model $f_\theta(s, a)$ by minimizing $\sum_i \|f_\theta(s_i, a_i) - s_i'\|^2$
3. Backpropagate through $f_\theta(s, a)$ into policy to optimize $\pi_\theta(a_t|s_t)$
4. Run $\pi_\theta(a_t|s_t)$
5. Append visited tuples (s, a, s') to $\mathcal{D}$



Backprop:

$$\max_\theta \sum_t \gamma^t R(s_t, a_t)$$

Remember: we make mistakes!

Model Uncertainty

Recap: ~~Background Planning~~ with Policy Backpropagation

Better Algorithm for online planning:

1. Run a base policy $\pi_0(a_t|s_t)$ (e.g., a random policy) to collect data samples $\mathcal{D}\{(s, a, s')_i\}$
2. Learn a dynamics model $f_\theta(s, a)$ by minimizing $\sum_i \|f_\theta(s_i, a_i) - s'_i\|^2$
3. Plan through $f_\theta(s, a)$ to choose actions
4. Execute those actions
5. Append visited tuples (s, a, s') to $\mathcal{D}$

Backprop:

$$\max_\theta \sum_t \gamma^t R(s_t, a_t)$$

Remember: we make mistakes!

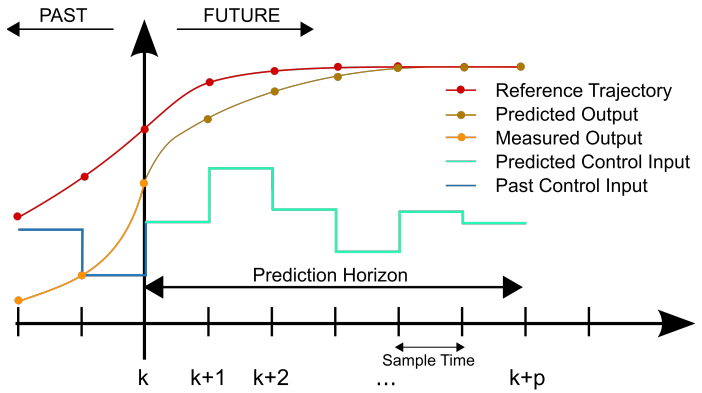
Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)

Every N episodes

1. Run a base policy $\pi_0(a_t|s_t)$ (e.g., a random policy) to collect data samples $\mathcal{D}\{(s, a, s')_i\}$
2. Learn a dynamics model $f_\theta(s, a)$ by minimizing $\sum_i \|f_\theta(s_i, a_i) - s_i'\|^2$
3. Plan until time horizon $H < T$ using the model
4. Apply only first action of the plan
5. Append visited tuples (s, a, s') to $\mathcal{D}$

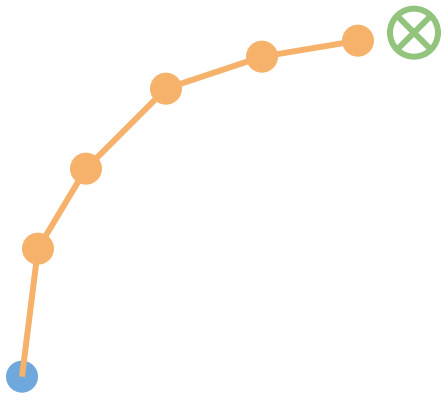


https://de.wikipedia.org/wiki/Model_Predictive_Control

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)



Optimize trajectory

Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

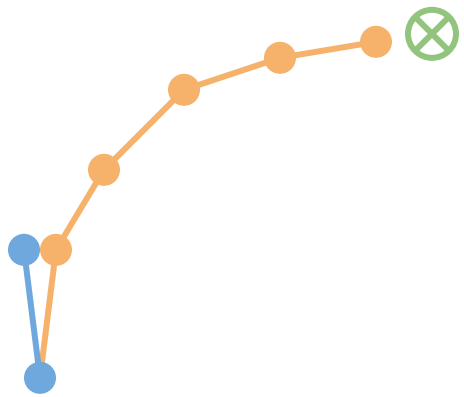
Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)



Apply first optimal action

Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

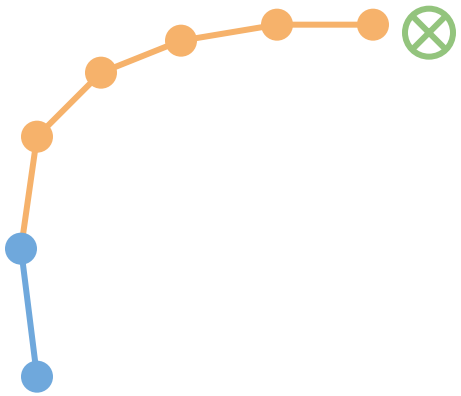
Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)



Optimize trajectory

Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

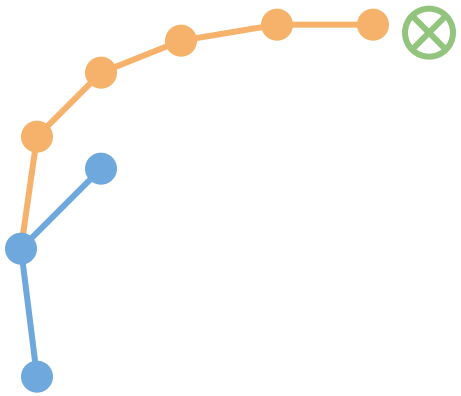
Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)



Apply first optimal action

Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

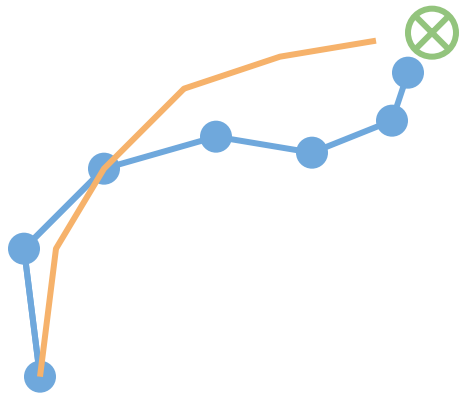
Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)



Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

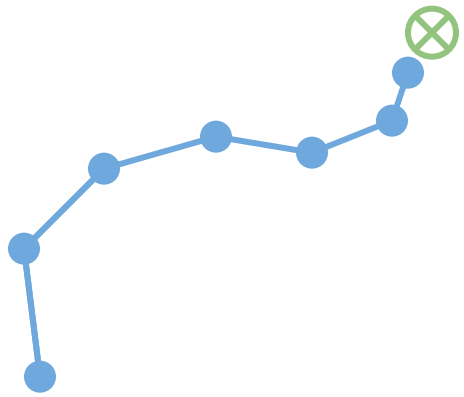
Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)

- Errors don't accumulate
- Don't need a perfect model, just one pointing in the right direction



Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

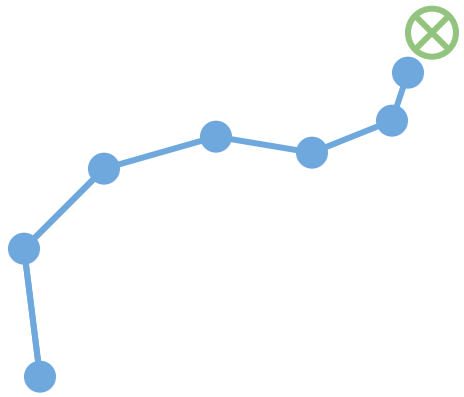
Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)

- Errors don't accumulate
- Don't need a perfect model, just one pointing in the right direction

This sounds expensive?

- Reuse solution from previous step as initial guess for next plan



Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K} \left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)} \right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

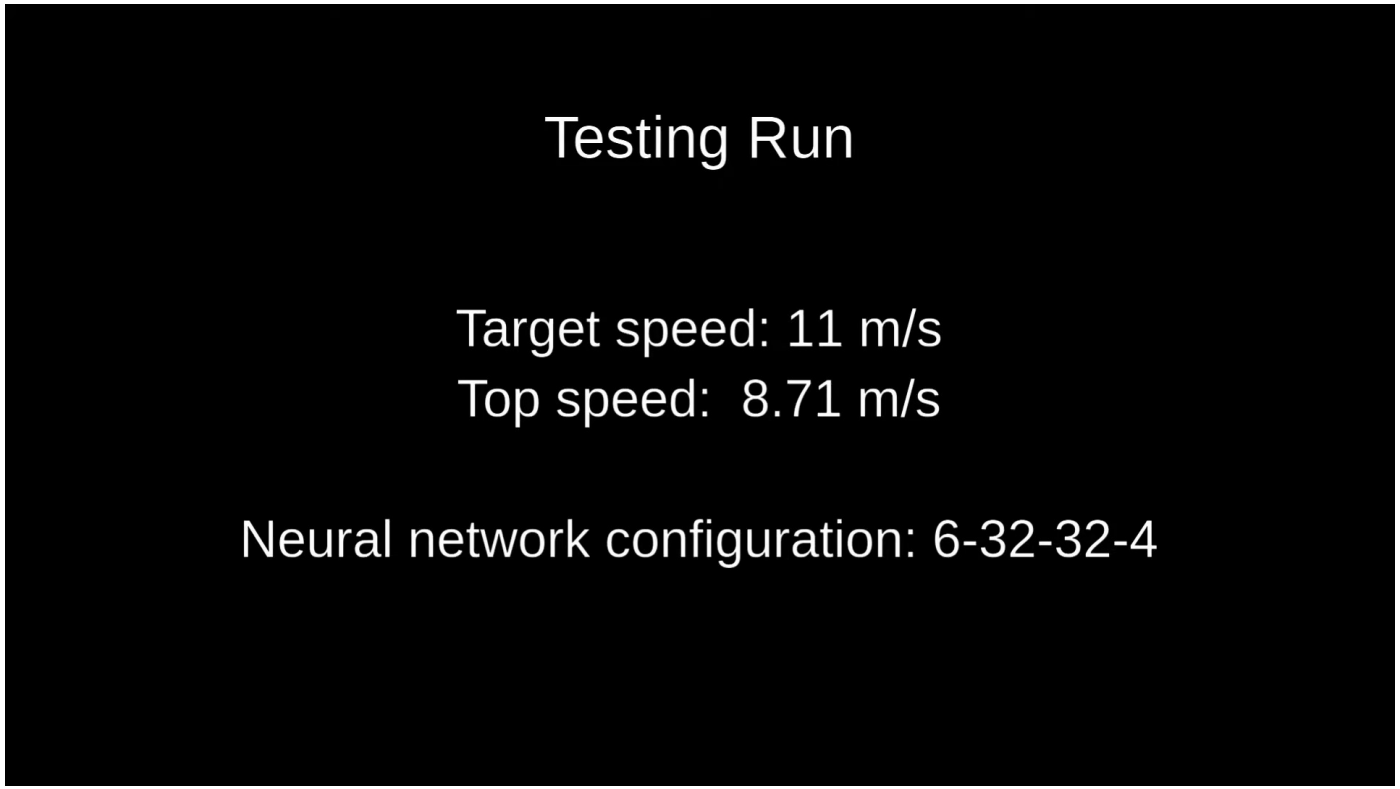
Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Example taken from https://sites.google.com/view/mbrl-tutorial

Model Uncertainty

Idea: don't commit to a plan bullheadedly but re-plan online

Model Predictive Control (MPC)



Williams et al.: Information Theoretic MPC for Model-Based Reinforcement Learning. ICRA. 2017.

Model Uncertainty

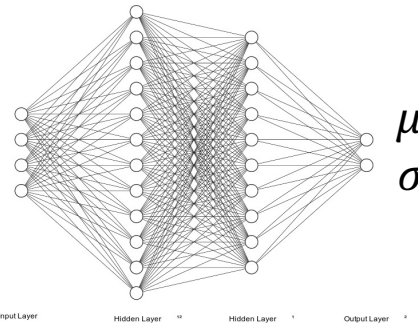
Plan Conservatively: Model Uncertainty Propagation

- Estimate/Quantify Model Uncertainty
  - Gaussian Processes
  - Monte Carlo Dropout
  - Probabilistic Neural Networks
  - Model Ensembles
- Propagate for multiple steps in the future

Uncertainty Estimation

Probabilistic Neural Networks

- Capture Aleatoric/Process Uncertainty

(s_t, a_t)  μ_{t+1} σ^2_{t+1}

- Loss function for training
  - Negative log prediction probability
  - Assume: dataset $\mathcal{D}$ with pairs of example data $\{(s_i, a_i), s_{i+1}\}$

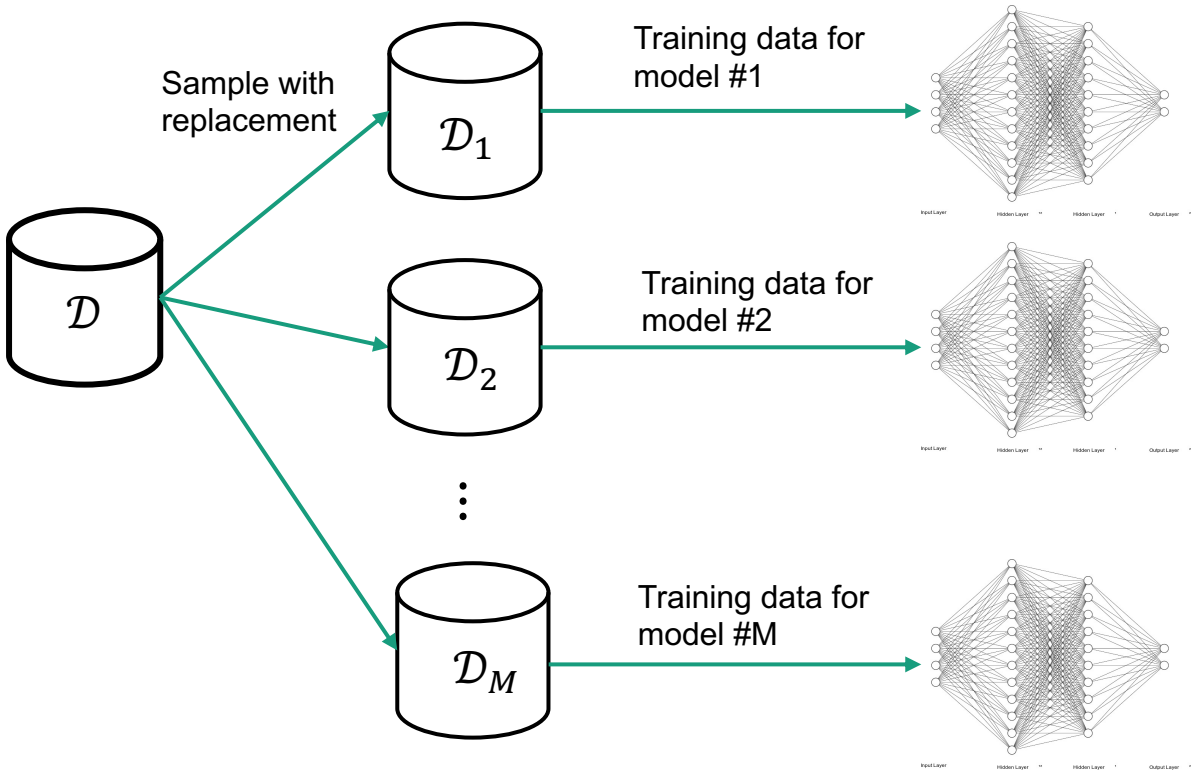
$$\mathcal{L}(w) = -\sum_{i=1}^{n} \log \hat{f}(s_{i+1} | s_i, a_i; w)$$

$$\mathcal{L}_{Gauss}(w) = \sum_{i=1}^{n} [\mu(s_i, a_i) - s_{i+1}]\, \Sigma^{-1}(s_i, a_i) [\mu(s_i, a_i) - s_{i+1}] + \log|\Sigma(s_i, a_i)|$$

Uncertainty Estimation

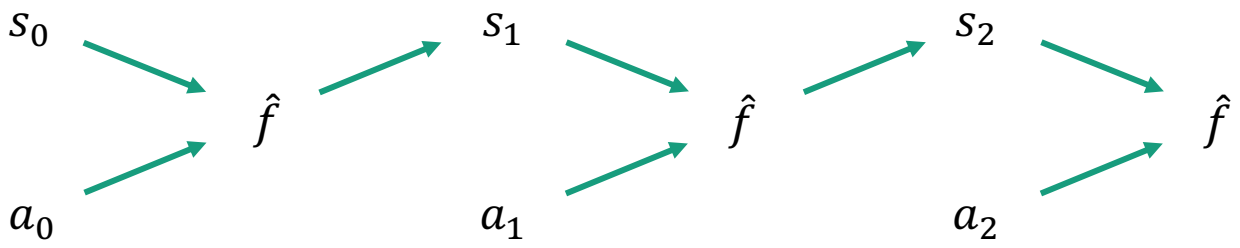
Ensembles of Probabilistic Neural Networks

- Capture Epistemic/model Uncertainty



Uncertainty Propagation

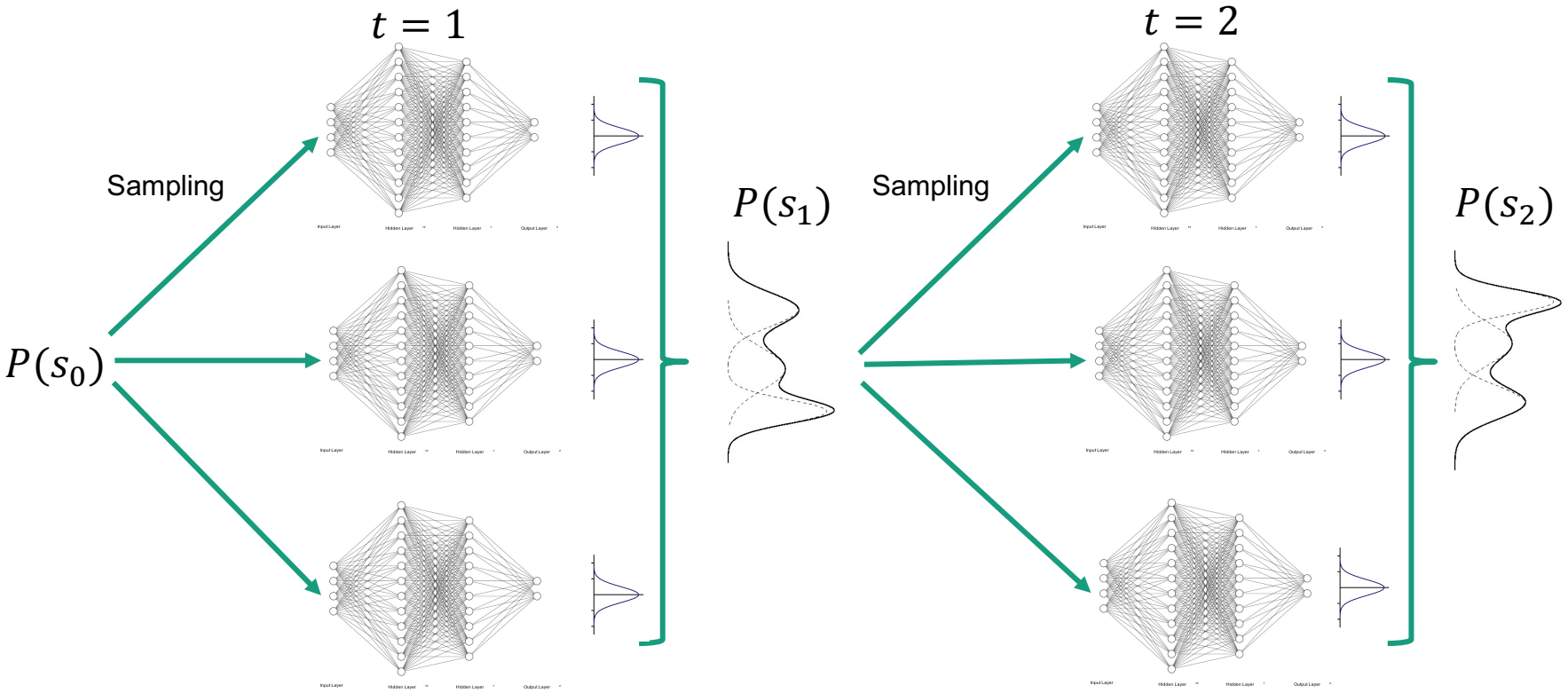
How do we propagate uncertainty to several future time-steps?



Uncertainty Propagation

How do we propagate uncertainty to several future time-steps?

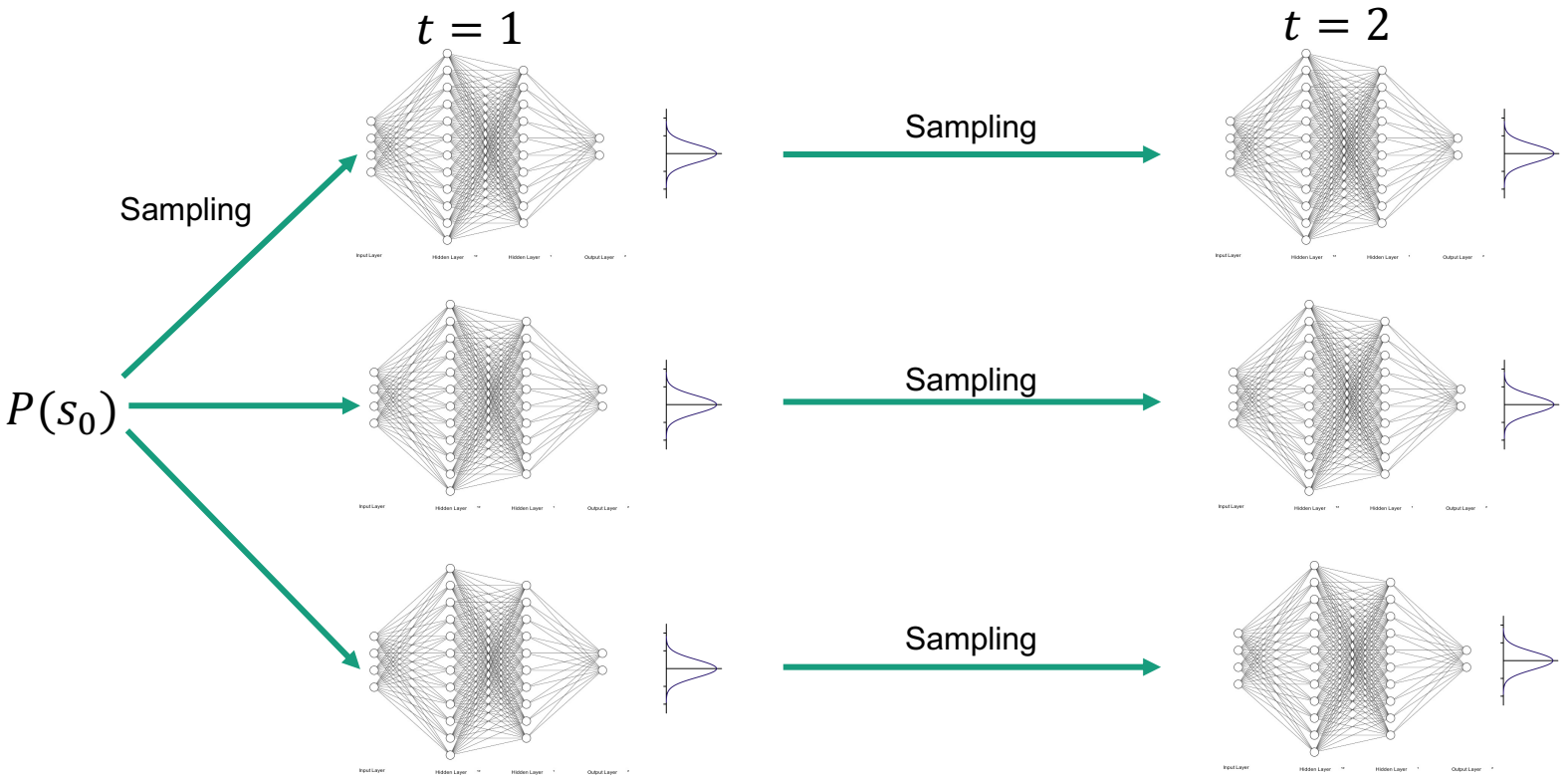
- Moment Matching



Uncertainty Propagation

How do we propagate uncertainty to several future time-steps?

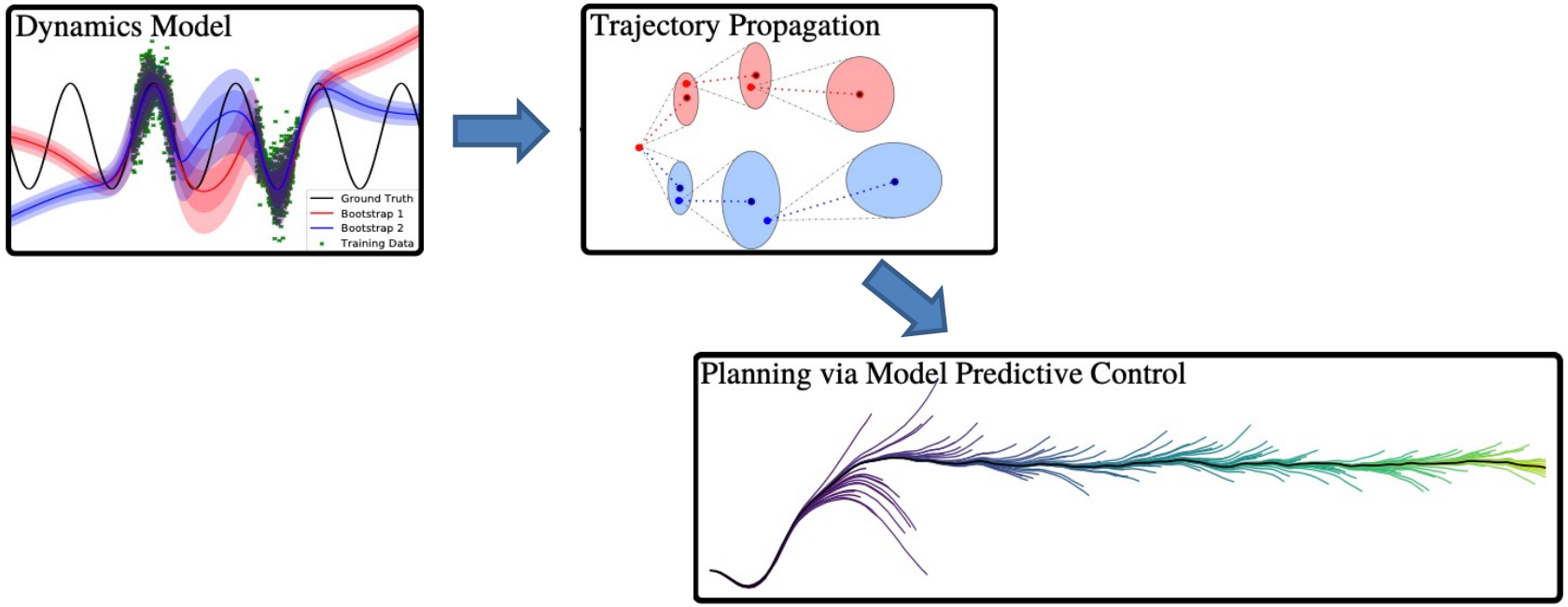
- Trajectory Sampling



Uncertainty Estimation

Ensembles of Probabilistic Neural Networks

- Capture Epistemic/model Uncertainty
- Sample Trajectories
- Plan via MPC



Kurtland Chua et al.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. NIPS 2018.

Use Uncertainty information

Plan conservatively:
Closed-Loop control with Model Uncertainty Propagation

- PILCO *(probabilistic inference for learning control)*
  - Background planning
  - Gaussian Processes as state-space model
  - Moment-matching for posterior (next state) distribution (uncertainty propagation)

Marc Deisenroth et al.: PILCO: A model-based and data-efficient approach to policy search. ICML. 2011.

Use Uncertainty information

Plan conservatively:
Closed-Loop control with Model Uncertainty Propagation

- PILCO *(probabilistic inference for learning control)*
  - Background planning
  - Gaussian Processes as state-space model
  - Moment-matching for posterior (next state) distribution (uncertainty propagation)
- Why GP?
  1. GPs have low extrapolation error by design (they fall back to prior distribution)

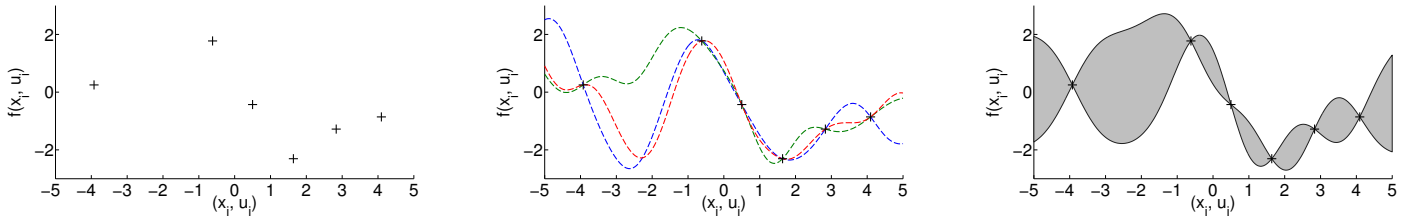


Figure 1. Small data set of observed transitions (left), multiple plausible deterministic function approximators (center), probabilistic function approximator (right). The probabilistic approximator models uncertainty about the latent function.

Marc Deisenroth et al.: PILCO: A model-based and data-efficient approach to policy search. ICML. 2011.

Use Uncertainty information

Plan conservatively:
Closed-Loop control with Model Uncertainty Propagation

- PILCO *(probabilistic inference for learning control)*
  - Background planning
  - Gaussian Processes as state-space model
  - Moment-matching for posterior (next state) distribution (uncertainty propagation)
- Why GP?
  1. GPs have low extrapolation error by design (they fall back to prior distribution)
  2. Moment-matching for RBF kernel can be calculated analytically

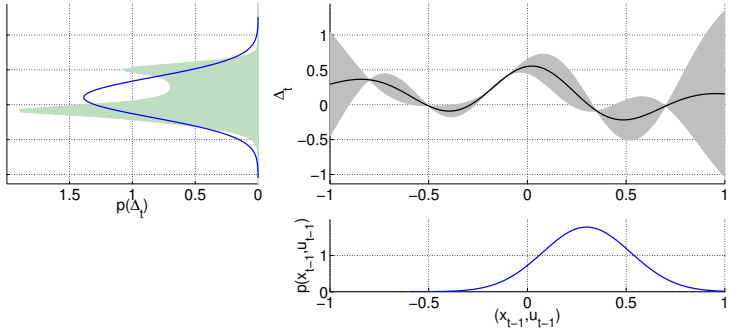


Figure 2. GP prediction at an uncertain input. The input distribution $p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ is assumed Gaussian (lower right panel). When propagating it through the GP model (upper right panel), we obtain the shaded distribution $p(\Delta_t)$, upper left panel. We approximate $p(\Delta_t)$ by a Gaussian with the exact mean and variance (upper left panel).

Marc Deisenroth et al.: PILCO: A model-based and data-efficient approach to policy search. ICML. 2011.

Use Uncertainty information

Plan conservatively:
Closed-Loop control with Model Uncertainty Propagation

- PILCO *(probabilistic inference for learning control)*
  - Background planning
  - Gaussian Processes as state-space model
  - Moment-matching for posterior (next state) distribution (uncertainty propagation)

Algorithm 1 PILCO

1: **init:** Sample controller parameters $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Apply random control signals and record data.
2: **repeat**
3: Learn probabilistic (GP) dynamics model, see Sec. 2.1, using all data.
4: Model-based policy search, see Sec. 2.2–2.3.
5: **repeat**
6: Approximate inference for policy evaluation, see Sec. 2.2: get $J^\pi(\theta)$, Eqs. (10)–(12), (24).
7: Gradient-based policy improvement, see Sec. 2.3: get $\mathrm{d}J^\pi(\theta)/\mathrm{d}\theta$, Eqs. (26)–(30).
8: Update parameters θ (e.g., CG or L-BFGS).
9: **until** convergence; **return** θ^*
10: Set $\pi^* \leftarrow \pi(\theta^*)$.
11: Apply π^* to system (single trial/episode) and record data.
12: **until** task learned

https://www.youtube.com/watch?v=XiigTGKZfks

Marc Deisenroth et al.: PILCO: A model-based and data-efficient approach to policy search. ICML. 2011.

Use Uncertainty information

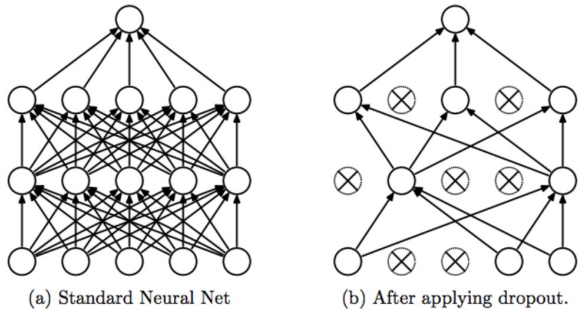
Plan conservatively:
Closed-Loop control with Model Uncertainty Propagation

- PILCO *(probabilistic inference for learning control)*
  - Background planning
  - Gaussian Processes as state-space model
  - Moment-matching for posterior (next state) distribution (uncertainty propagation)
- Advantages:
  + Unprecedent sample efficiency
  + Robust to small model errors
- Shortcomings:
  - GPs scale cubically with the number of training data samples
  - Only specific classes of policies and reward functions are supported
  - Assumes/requires problems with very smooth dynamics

Use Uncertainty information

Plan conservatively:
Closed-Loop control with Model Uncertainty Propagation

- Deep PILCO
  - Background planning
  - Gaussian Processes as state-space model
  - Moment-matching for posterior (next state) distribution (uncertainty propagation)
  - **Addresses all PILCO shortcomings**
- Advantages:
  + Scaling to large datasets + leveraging GPU processing
  + Very flexible policies
  + Robust to small model errors
- Shortcomings:
  - Still high execution times (e.g., for 40 learning iterations in Cartpole PILCO needed 20.7 hours (!!!) and Deep PILCO (GPU) needed 5.8 hours)


(a) Standard Neural Net (b) After applying dropout.

Nitish Srivastava et al.: Dropout: a simple way to prevent neural networks from overfitting. JMLR. 2014.