A Deeper Look At Planning as Learning from Replay

Harm van Seijen
Rich Sutton

Barbados Workshop, 2015

Outline

1. What are model-free / model-based methods.

2. Show that there exist model-free and model-based methods that compute, at each time-step, exactly the same values.

3. Use obtained insight to derive a new multi-step model-based method.

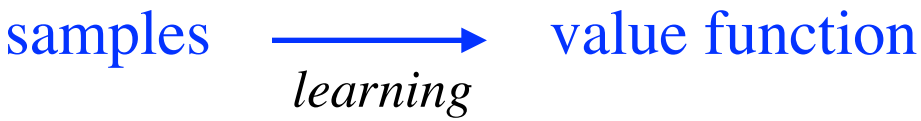
Motivation

"Learned transition models are ill-suited for planning, because modelling errors compound when reasoning over long time horizons."

- quote from some recent AAAI paper

model-free vs model-based

model-free:



- computationally cheap
- data inefficient

model-based:

- computationally expensive
- data efficient

RL methods

model-free:	*model-based:*
TD(λ)	R-max
Sarsa(λ)	Prioritized Sweeping

LSTD

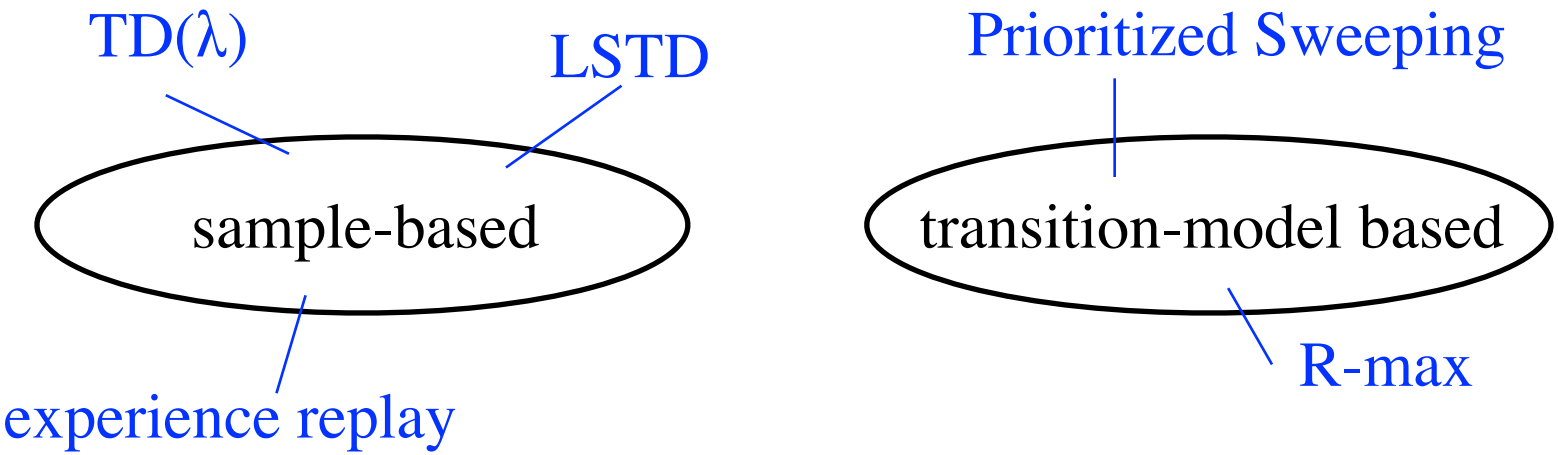
experience replay

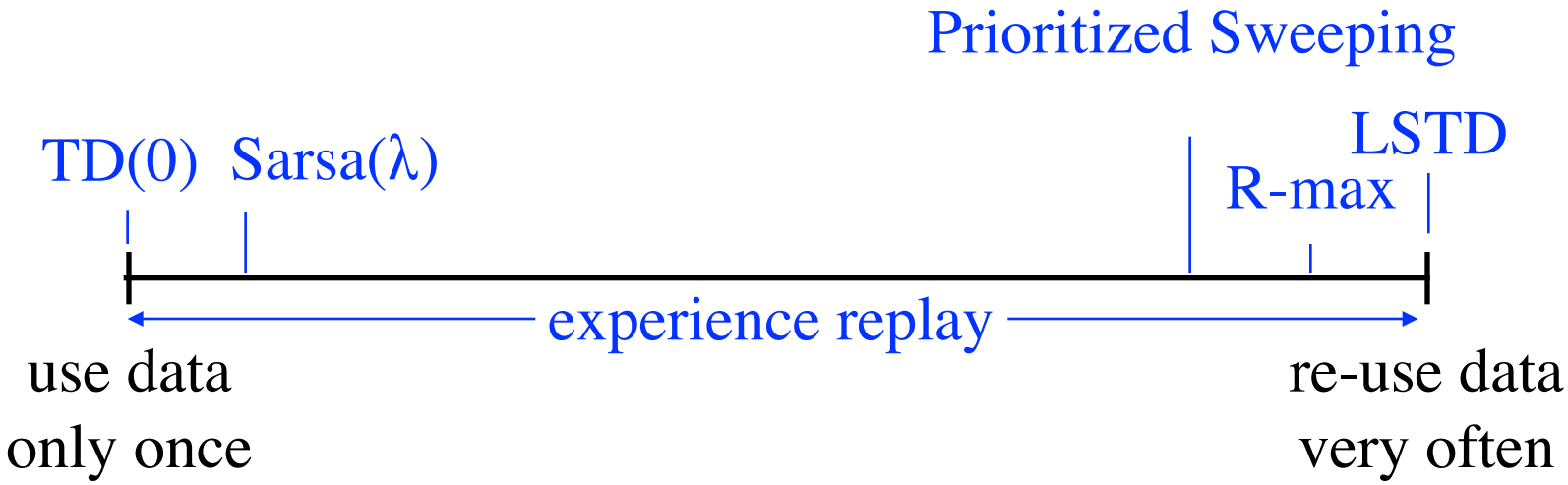
- based on samples

… but:

- computationally expensive
- data efficient

model-free vs model-based





sample-based vs transition-model based

"LSTD solution of a set of samples is equal to the fixed point of a linear, least-squares model"

Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. Dyna-style planning with linear function approximation and prioritized sweeping. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 528–536, 2008.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M.L. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pp. 752–759, 2008.

…however, in practise:

- only approximation of least-squares model is learned
- due to bounded computation the iterative model-based planning process terminates before convergence

Our result

- Exact equivalence between a sample-based method and a model-based method *at each moment in time*.
- Methods involved:
  - sample-based: linear TD(0) + experience replay
  - model-based: linear Dyna without model-free updates

Algorithm 1: Replaying TD(0) updates

INPUT: $\alpha, K, \boldsymbol{\theta}_{init}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$
$\mathcal{V} \leftarrow \emptyset$ *($\mathcal{V}$ is an ordered set)*
obtain initial $\boldsymbol{\phi}$
Loop:
  obtain next feature vector $\boldsymbol{\phi}'$, γ and reward R
  add $(\boldsymbol{\phi}, R, \gamma, \boldsymbol{\phi}')$ to $\mathcal{V}$
  Repeat K times:
    $\boldsymbol{\theta}_{target} \leftarrow \boldsymbol{\theta}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$
    For all $(\boldsymbol{\varphi}, r, \bar{\gamma}, \boldsymbol{\varphi}')$ in $\mathcal{V}$ (from oldest to newest):
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[r + \bar{\gamma}\,\boldsymbol{\theta}_{target}^\top \boldsymbol{\varphi}' - \boldsymbol{\theta}^\top \boldsymbol{\varphi}\big]\boldsymbol{\varphi}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$

Algorithm 2: Planning with a learned linear model

INPUT: $\alpha, K, \boldsymbol{\theta}_{init}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}$
$\boldsymbol{b} \leftarrow \boldsymbol{\theta}_{init}$, $F \leftarrow \mathbf{0}$
obtain initial $\boldsymbol{\phi}$
Loop:
  obtain next feature vector $\boldsymbol{\phi}'$, γ and reward R
  $F \leftarrow F + \alpha\big[\gamma\boldsymbol{\phi}' - F\boldsymbol{\phi}\big]\boldsymbol{\phi}^\top$
  $\boldsymbol{b} \leftarrow \boldsymbol{b} + \alpha(R - \boldsymbol{b}^\top\boldsymbol{\phi})\boldsymbol{\phi}$
  Repeat K times:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{b} + F^\top\boldsymbol{\theta}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$

Theorem: Given the same sequence of samples and parameter settings, Algorithm 1 and Algorithm 2 compute the same weight vectors at each time step.

Consequences

- It deepens our understanding:
  Sample-based approaches are much more related to transition-model based approached then assumed up to now.
- It provides a clear strategy for constructing new methods.
- A well-grounded multi-step model-based method can be constructed by combining the replay strategy with TD(λ).

Why a multi-step model?

Theoretical result for TD(λ) with linear function approximation (Peter Dayan '92):

- For $\lambda = 1$, fixed point solution is equal to the LMS solution
- For $\lambda < 1$, in general solution quality is worse.

"Learned transition models (forward models) are ill-suited for planning, because modelling errors compound when reasoning over long time horizons."

=> Examining the basis of this claim revealed that the observed phenomenon could be fully attributed to the limitations of one-step predictions.

Multi-step models - previous work

- Rich Sutton, *TD models: modeling the world at a mixture of time scales* (ICML '95)
  - Tabular multi-step models are used on problems with state aggregation.

- H. Yao et. al. : *Multi-step linear Dyna-style planning* (NIPS '09)
  - Extends multi-step models to domains with linear function approximation.
  - However, at its core it learns a one-step model.
    Hence, it does not improve the asymptotic performance.

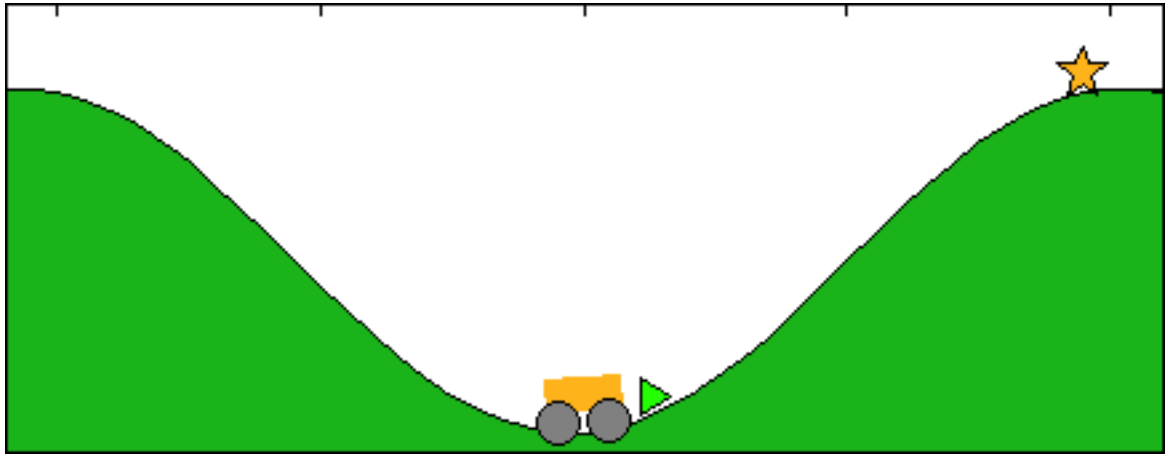
Our multi-step method

- Uses a linear multi-step model based on λ.
- Computes same weight vectors as true-online TD(λ) with experience replay.
- For $\lambda = 0$ (and a particular initialization), the method reduces to linear Dyna (without model-free updates).
- Method can be interpreted as a multi-step version of linear Dyna, as well as a forgetful version of LSTD(λ).
- We call the method: *forgetful LSTD(λ)*.

Algorithm 3: Forgetful LSTD(λ)

INPUT: $\alpha, \beta, \lambda, K, \boldsymbol{\theta}_{init}, \boldsymbol{d}_{init}, A_{init}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{init}, \;\; \boldsymbol{d} \leftarrow \boldsymbol{d}_{init}, \;\; A \leftarrow A_{init}$
obtain initial $\boldsymbol{\phi}$
$\boldsymbol{e} \leftarrow \mathbf{0}$
Loop:
    obtain next feature vector $\boldsymbol{\phi}'$,γ and reward R
    $\boldsymbol{e} \leftarrow (\mathcal{I} - \beta \boldsymbol{\phi}\boldsymbol{\phi}^\top)\boldsymbol{e} + \boldsymbol{\phi}$
    $A \leftarrow (\mathcal{I} - \beta \boldsymbol{\phi}\boldsymbol{\phi}^\top)A + \boldsymbol{e}(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')^\top$
    $\boldsymbol{d} \leftarrow (\mathcal{I} - \beta \boldsymbol{\phi}\boldsymbol{\phi}^\top)\boldsymbol{d} + \boldsymbol{e}R$
    $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e}$
    Repeat K times:
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\boldsymbol{d} - A\boldsymbol{\theta})$
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$

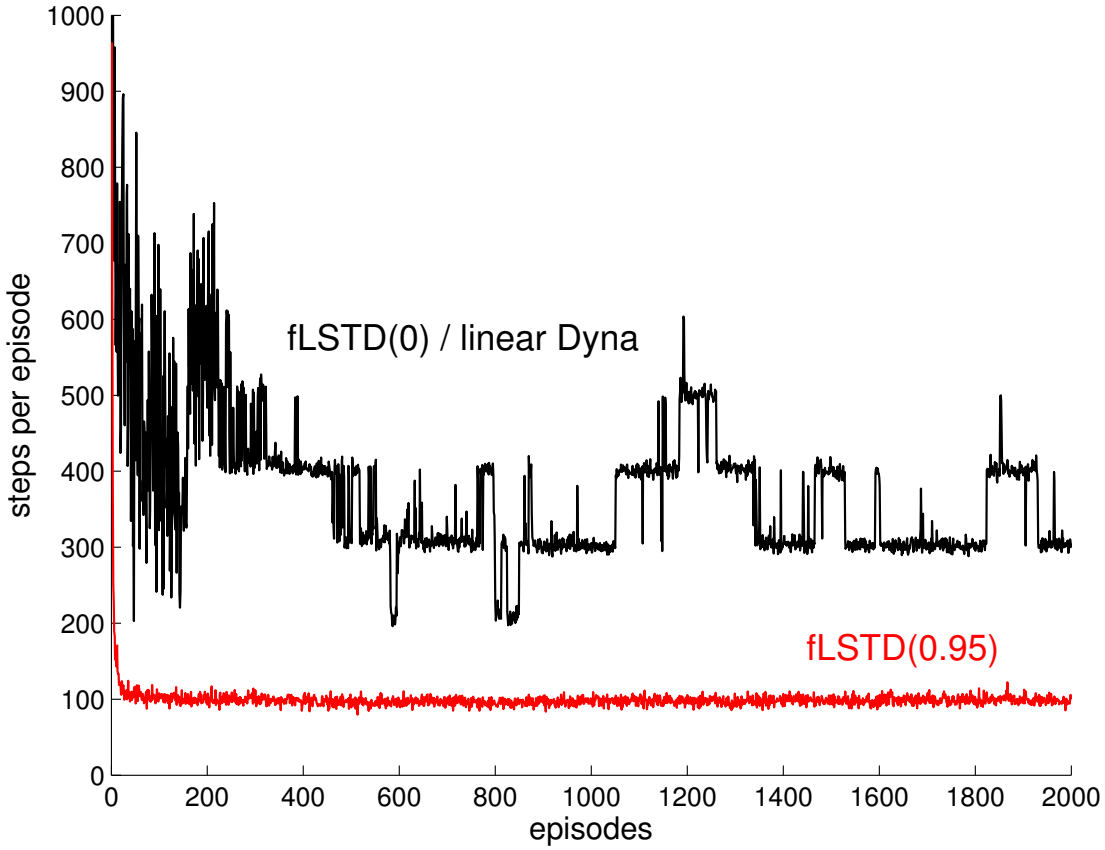
Experiment: Mountain Car task



- underpowered cart has to move up a hill
- 3 actions: {left, right, do nothing}
- state-space: (position, velocity)
- ~~default representation of state-space:~~
  ~~10 tilings consisting of 10 x 10 tiles~~

  3 tilings consisting of 3 x 3 tiles

one-step vs multi-step performance



Summary

- Sample-based methods are much more related to transition-model based methods than assumed up to now.
- Specifically, TD(0) enhanced by replay computes, at each time step, the same values as a method based on a learned linear model.
- Our strategy to prove this can also be used to derive new model-based methods.
- We introduced *forgetful LSTD(λ)*, a multi-step method that can be used for control in tasks with significant function approximation.

Thanks!

For more information:

- ICML'15 — full paper (not currently online)
- RLDM'15 — extended abstract (online at my homepage)