

Reinforcement learning with kernels and Gaussian processes

Yaki Engel
University of
Alberta

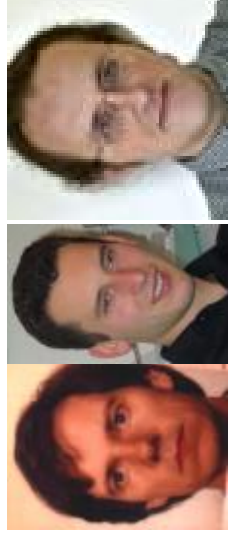
Shie Mannor
McGill University

Ron Meir
Technion

Give me a lever long enough and a fulcrum
on which to place it, and I shall move the
world



Give me a good kernel and a convex
optimization problem, and I shall learn the
world (ML folklore)



Some shortcomings of (most) RL algorithms

- Designed for Euclidean state spaces
- No confidence estimates
 - No stopping conditions
 - Do not guide exploration
- Function Approximation has to be predefined
 - Many functions needed (CMAC, NN)
 - Understanding solutions is hard
- Embarrassingly **not** parallel
 - More CPUs will not help
 - Hard to share information

Our approach

Look at the discounted reward from state \mathbf{x} (policy is fixed)

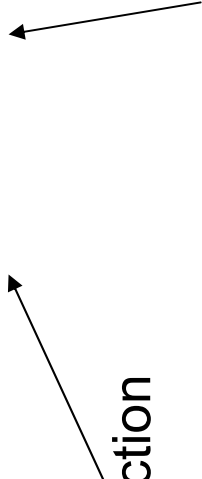
$$D(\mathbf{x}) = \sum_{i=0}^{\infty} \gamma^i R(\mathbf{x}_i) | \mathbf{x}_0 =, \text{ with } \mathbf{x}_{i+1} \sim p^\mu(\cdot | \mathbf{x}_i).$$

Also:

$$D(\mathbf{x}) = R(\mathbf{x}) + \gamma D(\mathbf{x}'), \text{ with } \mathbf{x}' \sim p^\mu(\cdot | \mathbf{x})$$

Classical approach: $D(\mathbf{x}) = v(\mathbf{x}) + \Delta V(\mathbf{x})$

Deterministic value function



Our approach (cont')

$$D(\mathbf{x}) = V(\mathbf{x}) + \Delta V(\mathbf{x})$$



Random value variable

Zero-mean **random** disturbance

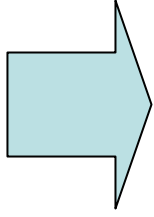
Interpretation:

- Model is unknown (that's why we do learning)
- Bayesian value variable
- Extrinsic vs. Intrinsic uncertainty
 - Known model
 - Deterministic model

Building the model

- Policy is fixed

$$D(\mathbf{x}) = V(\mathbf{x}) + \Delta V(\mathbf{x}) \qquad D(\mathbf{x}) = R(\mathbf{x}) + \gamma D(\mathbf{x}')$$



$$R(\mathbf{x}) = V(\mathbf{x}) - \gamma V(\mathbf{x}') + N(\mathbf{x}, \mathbf{x}'), \quad \mathbf{x}' \sim p^\mu(\cdot | \mathbf{x})$$

where

$$N(\mathbf{x}, \mathbf{x}') := \Delta V(\mathbf{x}) - \gamma \Delta V(\mathbf{x}')$$

If we observe a sequence of states x_0, x_1, x_2, \dots and rewards $R(x_0), \dots$

Generative model


$$R(\mathbf{x}_i) = V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i, \mathbf{x}_{i+1}), \quad i = 0, \dots, t-1$$

The prior

$$R(\mathbf{x}_i) = V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i, \mathbf{x}_{i+1})$$

- Gaussian process for V and ΔV
 - Simplest assumption
- ΔV is white: $V \sim \mathcal{N}(0, \sigma(x)^2 \delta(x - x))$
- V has a meaningful prior: $V \sim \mathcal{N}(0, k(\cdot, \cdot))$
 - i.e. $\mathbf{E}[V(\mathbf{x})V(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$
 - k is a kernel (known a-priori)
 - Happy coincidence: Mercer \rightarrow covariance
 - Choosing/designing k via domain knowledge

The posterior

- Can obtain posterior via “simple” algebra
- For a new point \mathbf{x} :  posterior mean

$$\hat{\nu}_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^\top \alpha_t,$$

$$\hat{\rho}_t(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t(\mathbf{x})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{x}),$$

posterior variance 

where

α_t = length t vector

\mathbf{C}_t = $t \times t$ matrix

$$\mathbf{k}_t(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \dots, k(\mathbf{x}, \mathbf{x}_t))$$

- So we are done if t is small!

Making the algorithm on-line

- But t is growing (Memory, computation $\sim t^2$)
- Idea:
 - Maintain a (hopefully) small dictionary of input points
 - Given a new point:
 - Check if contribution can be approximated by dictionary points
 - Yes: keep dictionary the same
 - No: Add to dictionary
 - Update posterior based only on dictionary points
- Feature space interpretation – is point in the approximate span of the dictionary?
- Memory = (size of dictionary)²
- # Computations per point = (size of dictionary)²

Regularization

- Keeping the dictionary small
- Procedure has some PCA properties
- Starting from a “flattish” prior
- Prior value not important, prior shape is
- Assuming disturbance variance is large
- Maximum likelihood interpretation

Changing the policy

- Optimistic policy iteration
 - Works well in applications
 - Can take variance into account
- SARSA based approach
 - Learn state-action values
 - Look at an MDP $\mathcal{X}=\mathcal{X} \times \mathcal{U}$
 - For a given μ : $p'(x', u'|x, u) = p(x'|x, u) \mu(u'|x')$
- Use component kernel:
 - k_1 kernel on \mathcal{X} , k_2 kernel on \mathcal{U}
 - $k((x, u), (x', u')) = k_1(x, x') k_2(u, u')$ is a kernel
 - Design kernel to make the problem of finding an optimal action easy (quadratic/linear)

Finding kernels

- Is finding kernels easier than finding basis functions?
 - The paper count & bag of tricks
 - Kernels (can) represent similarity, basis functions the structure of the value function
- Kernels act as “adaptive” basis functions
 - Non-parametric estimation
- Can enable RL in non-standard new state spaces
 - Examples: Fisher kernels, tree kernels, string kernels

A good open question: learning “RL” kernels

Parallelizing

- Objective:
 - Specialization of agents
 - Speed-up
- Basic equations are linear estimation equations
 - Standard fusion of linear estimators
 - Variance is taken into account
- Slight complication:
 - Run each separately → dictionaries, α , C
 - Obtain two dictionaries
 - Merge dictionaries somehow (tricky)

Does it work?

- Works well in continuous state and action maze worlds ...
- Some experiments with an octopus arm – high dimensional (88) input control task, polynomial kernel
(see <http://www.cs.ualberta.ca/~yaki/> + movies!)
- Jury is still out

Wrap-up

- Bayesian
- Flexible approach (any state space)
- Design kernel, not basis function
 - Rich representations
 - Decomposable/hierarchical kernels
- Parallelizable
- Gives you extrinsic variance → guide exploration
- Inherently policy based, can be made SARSA like

Related references

- Engel, M. and Meir:
 - [The Kernel Recursive Least Squares Algorithm](#) (IEEE Trans. on Signal Processing, 52(8):2275–2285, 2004)
 - [Sparse Online Greedy Support Vector Regression](#) (ECML 2002)
- [Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning](#) (ICML 2003),
 - [Reinforcement learning with Gaussian Processes](#) (ICML 2005)

Sparisification

RL with GPs

Tuesday @ 11

Reinforcement learning with kernels and Gaussian processes

Yaki Engel
University of
Alberta

Shie Mannor
McGill University

Ron Meir
Technion

Thank you