Gaussian Process Temporal Difference Learning - Theory and Practice

Yaakov Engel

Collaborators: Shie Mannor, Ron Meir, Peter Szabo,
Dmitry Volkinshtein, Nadav Aharony, Tzachi Zehavi

ALBERTA INGENUITY CENTRE FOR MACHINE LEARNING

UNIVERSITY OF ALBERTA
**Timeline**

- **ICML’03**: Bayes meets Bellman paper – GPTD model for MDPs with deterministic transitions
- **ICML’05**: RL with GPs paper – GPTD model for general MDPs + GPSARSA for learning control
- **NIPS’05**: Learning to control an Octopus Arm – GPTD applied to a high dimensional control problem
- **OPNET’05**: Network association-control with GPSARSA
**Why use GPs in RL?**

- A Bayesian approach to value estimation
- Forces us to make our assumptions explicit
- Non-parametric – priors are placed and inference is performed directly in function space (kernels).
- But, can also be defined parametrically
- Domain knowledge intuitively coded in priors
- Provides full posterior, not just point estimates
- Efficient, on-line implementations, suitable for large problems
THE BAYESIAN APPROACH

- $Z$ – hidden process, $Y$ – observable
- We want to infer $Z$ from measurements of $Y$
- Statistical dependence between $Z$ and $Y$ known: $P(Y|Z)$
- Place prior over $Z$, reflecting our uncertainty: $P(Z)$
- Observe $Y = y$
- Compute posterior: $P(Z|Y = y) = \frac{P(y|Z)P(Z)}{\int dZ' P(y|Z')P(Z')}$
**Gaussian Processes**

**Definition:** “An indexed set of jointly Gaussian random variables”

**Note:** The index set $\mathcal{X}$ may be just about any set.

**Example:** $F(x)$, index is $x \in [0, 1]^n$

$F$’s distribution is specified by its mean and covariance:

$$E[F(x)] = m(x), \quad \text{Cov}[F(x), F(x')] = k(x, x')$$

$m$ is a function $\mathcal{X} \rightarrow \mathbb{R}$, $k$ is a function $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.

**Conditions on $k$:**

Symmetric, positive definite $\Rightarrow k$ is a Mercer kernel
GP REGRESSION

Model equation:

\[ Y(x) = F(x) + N(x) \]

Prior:

\[ F \sim \mathcal{N}(0, k(\cdot, \cdot)) \]

Noise:

\[ N \sim \mathcal{N}(0, \sigma^2 \delta(\cdot - \cdot)) \]

Goal:

Find the posterior distribution of \( F \), given a sample for \( Y \) (via Bayes’ rule)
Example
Markov Decision Processes

\( \mathcal{X} \): state space

\( \mathcal{U} \): action space

\( p: \mathcal{X} \times \mathcal{X} \times \mathcal{U} \rightarrow [0, 1], \quad x_{t+1} \sim p(\cdot| x_t, u_t) \)

\( q: \mathbb{R} \times \mathcal{X} \times \mathcal{U} \rightarrow [0, 1], \quad R(x_t, u_t) \sim q(\cdot| x_t, u_t) \)

A Stationary policy:

\( \mu: \mathcal{U} \times \mathcal{X} \rightarrow [0, 1], \quad u_t \sim \mu(\cdot| x_t) \)

Discounted Return:

\( D^\mu(x) = \sum_{i=0}^{\infty} \gamma^i R(x_i, u_i) | (x_0 = x) \)

Value function:

\( V^\mu(x) = E_\mu[D^\mu(x)] \)

Goal: Find a policy \( \mu^* \) maximizing \( V^\mu(x) \quad \forall x \in \mathcal{X} \)
Bellman’s Equation

For a fixed policy $\mu$:

$$V^\mu(x) = \mathbb{E}_{x',u|x} \left[ R(x, u) + \gamma V^\mu(x') \right]$$

Optimal value and policy:

$$V^*(x) = \max_\mu V^\mu(x), \quad \mu^* = \arg\max_\mu V^\mu(x)$$

How to solve it?
- Methods based on Value Iteration (e.g. Q-learning)
- Methods based on Policy Iteration (e.g. SARSA, OPI, Actor-Critic)
SOLUTION METHOD TAXONOMY

RL Algorithms

Purely Policy based (Policy Gradient)

Value−Function based

Value Iteration type (Q−Learning)

Policy Iteration type (Actor−Critic, OPI, SARSA)

PI methods need a “subroutine” for policy evaluation
**WHAT’S MISSING?**

**Shortcomings of current policy evaluation methods:**

- Some methods can only be applied to small problems
- No probabilistic interpretation - how good is the estimate?
- Only parametric methods are capable of operating on-line
- Non-parametric methods are more flexible but only work off-line
- Small-step-size (stoch. approx.) methods use data inefficiently
- Finite-time solutions lack interpretability, all statements are asymptotic
- Convergence issues
Model Equations:

\[ R(x_i) = V(x_i) - \gamma V(x_{i+1}) + N(x_i, x_{i+1}) \]

Or, in compact form:

\[ R_t = H_{t+1} V_{t+1} + N_t \]

\[ H_t = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & \ldots & 1 & -\gamma \end{bmatrix} \]

Our (Bayesian) goal:

Find the posterior distribution of \( V \),
given a sequence of observed states and rewards.
**Deterministic Dynamics**

**Bellman’s Equation:**

\[ V(x_i) = \tilde{R}(x_i) + \gamma V(x_{i+1}) \]

**Define:**

\[ N(x) = R(x) - \tilde{R}(x) \]

**Assumption:** \( N(x_i) \) are Normal, i.i.d., with variance \( \sigma^2 \).

**Model Equations:**

\[ R(x_i) = V(x_i) - \gamma V(x_{i+1}) + N(x_i) \]

**In compact form:**

\[ R_t = H_{t+1} V_{t+1} + N_t \text{, with } N_t \sim \mathcal{N} \{0, \sigma^2 I\} \]
STOCHASTIC DYNAMICS

The discounted return:

\[ D(x_i) = \mathbb{E}_\mu D(x_i) + (D(x_i) - \mathbb{E}_\mu D(x_i)) = V(x_i) + \Delta V(x_i) \]

For a stationary MDP:

\[ D(x_i) = R(x_i) + \gamma D(x_{i+1}) \quad (\text{where } x_{i+1} \sim p(\cdot | x_i, u_i), \ u_i \sim \mu(\cdot | x_i)) \]

Substitute and rearrange:

\[
\begin{align*}
R(x_i) &= V(x_i) - \gamma V(x_{i+1}) + N(x_i, x_{i+1}) \\
N(x_i, x_{i+1}) \overset{\text{def}}{=} & \Delta V(x_i) - \gamma \Delta V(x_{i+1})
\end{align*}
\]

Assumption: \( \Delta V(x_i) \) are Normal, i.i.d., with variance \( \sigma^2 \).

In compact form:

\[ R_t = H_{t+1} V_{t+1} + N_t \quad \text{with } N_t \sim \mathcal{N} \{0, \sigma^2 H_{t+1} H_{t+1}^\top\} \]
**The Posterior**

**General noise covariance:** \( \text{Cov}[N_t] = \Sigma_t \)

**Joint distribution:**
\[
\begin{bmatrix}
R_{t-1} \\
V(x)
\end{bmatrix} \sim \mathcal{N}
\begin{bmatrix}
0 \\
0
\end{bmatrix},
\begin{bmatrix}
H_t K_t H_t^\top + \Sigma_t & H_t k_t(x) \\
k_t(x)^\top H_t & k(x, x)
\end{bmatrix}
\]

**Invoke Bayes’ Rule:**
\[
\text{E}[V(x)|R_{t-1} = r_{t-1}] = k_t(x)^\top \alpha_t
\]
\[
\text{Cov}[V(x), V(x')|R_{t-1} = r_{t-1}] = k(x, x') - k_t(x)^\top C_t k_t(x')
\]

\( k_t(x) = (k(x_0, x), \ldots, k(x_t, x))^\top \), \( K_t = [k_t(x_0), \ldots, k_t(x_t)] \)

\( \alpha_t = H_t^\top \left( H_t K_t H_t^\top + \Sigma_t \right)^{-1} r_{t-1}, \quad C_t = H_t^\top \left( H_t K_t H_t^\top + \Sigma_t \right)^{-1} H_t. \)
A Parametric Gaussian Process Model

A linear combination of features:

\[ V(x) = \phi(x)^\top W \]

Prior on \( W \): Gaussian, with \( \mathbf{E}[W] = 0, \quad \text{Cov}[W, W] = \mathbf{I} \)

Prior on \( V \): Gaussian, with
\[
\mathbf{E}[V(x)] = 0, \quad \text{Cov}[V(x), V(x')] = \phi(x)^\top \phi(x')
\]
### Comparison of Models

<table>
<thead>
<tr>
<th></th>
<th>Parametric</th>
<th>Nonparametric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parametrization</td>
<td>$V(x) = \phi(x)^\top W$</td>
<td>None, $V$ is $V$</td>
</tr>
<tr>
<td>Prior</td>
<td>$W \sim \mathcal{N}{0, I}$</td>
<td>$V \sim \mathcal{N}{0, k(\cdot, \cdot)}$</td>
</tr>
<tr>
<td>$E[V(x)]$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\text{Cov}[V(x), V(x')]$</td>
<td>$\phi(x)^\top \phi(x')$</td>
<td>$k(x, x')$</td>
</tr>
<tr>
<td>We seek</td>
<td>$W</td>
<td>R_{t-1}$</td>
</tr>
</tbody>
</table>

If we can find a set of basis functions satisfying $\phi(x)^\top \phi(x') = k(x, x')$, the two models become equivalent.

In fact, such a set always exists [Mercer].

However, it may be infinite.
**Relation to Monte-Carlo Estimation**

In the stochastic model: \[ \Sigma_t = \sigma^2 H_{t+1} H_{t+1}^\top \]

Also, let: \[(Y_t)_i = \sum_{j=i}^{t} \gamma^{j-i} R(x_i, u_i)\]

Then:

\[
\mathbb{E}[W| R_t] = \left( \Phi_t \Phi_t^\top + \sigma^2 I \right)^{-1} \Phi_t Y_t \\
\text{Cov}[W| R_t] = \sigma^2 \left( \Phi_t \Phi_t^\top + \sigma^2 I \right)^{-1}
\]

That's the solution to GP regression on Monte-Carlo samples of the discounted return.
Since the posterior is Gaussian:

\[ \hat{w}_{t+1}^{MAP} = \mathbb{E}[W|R_t] = \left( \Phi_t \Phi_t^\top + \sigma^2 I \right)^{-1} \Phi_t Y_t \]

Performing ML inference using the same model we get:

\[ \hat{w}_{t+1}^{ML} = \left( \Phi_t \Phi_t^\top \right)^{-1} \Phi_t Y_t \]

That’s the LSTD(1) (Least-Squares Monte-Carlo) solution.
How can we perform policy improvement?

State values? Not without a transition model (even then tricky).

State-action (Q-) values? Yes!

Idea: Use a state-action value GP

How?

- Define a state-action kernel: \( k((x, u), (x', u')) \)
- Run GPTD on state-action pairs
- Use some semi-greedy action selection rule

We call this GPSARSA.
A SIMPLE EXPERIMENT
THE OCTOPUS ARM

Can bend and twist at any point
Can do this in any direction
Can be elongated and shortened
Can change cross section
Can grab using any part of the arm

Virtually infinitely many DOF
Our Arm Model

- Arm base
- Dorsal side
- Ventral side
- Pair #1
- C<sub>1</sub>
- C<sub>N</sub>
- Arm tip
- Pair #N+1
- Longitudinal muscle
- Transverse muscle

23/30
Each action specifies a set of fixed activations – one for each muscle in the arm.

Base rotation adds duplicates of actions 1, 2, 4 and 5 with positive and negative torques applied to the base.
Starting from a random position, bring {any part, tip} of arm into contact with a goal region, **optimally**.

**Optimality criteria:**

Time, energy, obstacle avoidance

**Constraint:**

We only have access to sampled trajectories

**Our approach:**

Define problem as a MDP  
Solve using a GPTD algorithm
The Task

$\text{t} = 1.38$
MOVIES
ASSOCIATION CONTROL IN WLANs
ASSOCIATION CONTROL IN WLANs

Setting: $n$ users, $m \ll n$ access points (APs),

The problem: Associate users with APs, optimally.

Complications: Users are not the same, they move around, change their behavior over time, what is meant by “optimally”? etc.

Idea: Model the system as a MDP, solve using GPSARSA

Results:

- Tested on simple networks using the OPNET simulator
- Preliminary results look promising
- More work is needed
CHALLENGES

- How to use value uncertainty?
- What’s a disciplined way to select actions?
- What’s the best noise covariance?
- Bias, variance, learning curves
- POMDPs
- More complicated tasks

Questions?