Composable Learning with Sparse Kernel Representations

Ekaterina Tolstaya, Ethan Stump, Alec Koppel, Alejandro Ribeiro
Dept. of Electrical and Systems Engineering
University of Pennsylvania
eig@seas.upenn.edu

October 3, 2018

Composable Learning

- Learning in **multi-agent** systems with **infrequent** communication
- Models learned by different agents **composed** as one

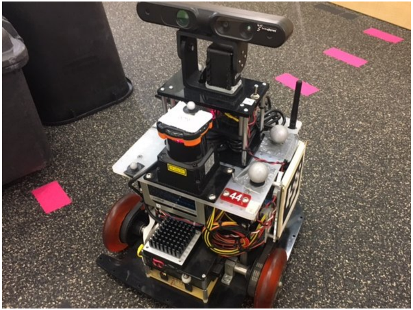
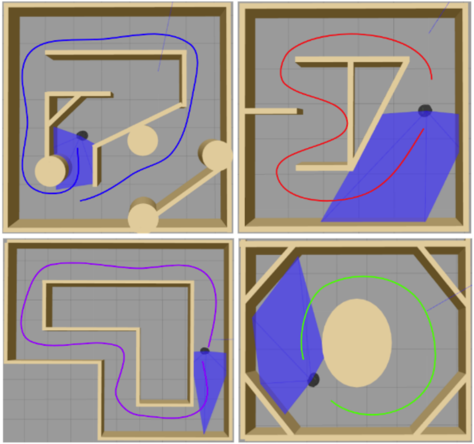


Figure 1: Scarab robot



- YouTube Video

Reinforcement Learning

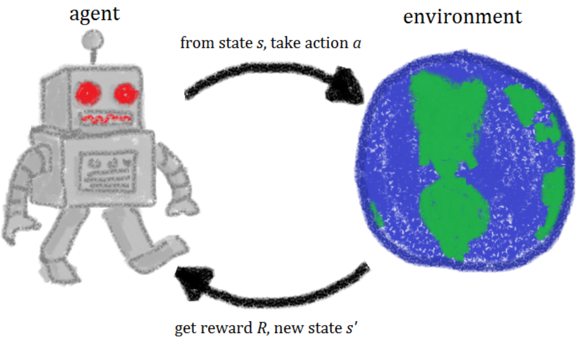


Figure 2: In Markov Decision problems, the goal is to find a controller $\pi(s)$ that maximizes the accumulation of rewards [Bel54].

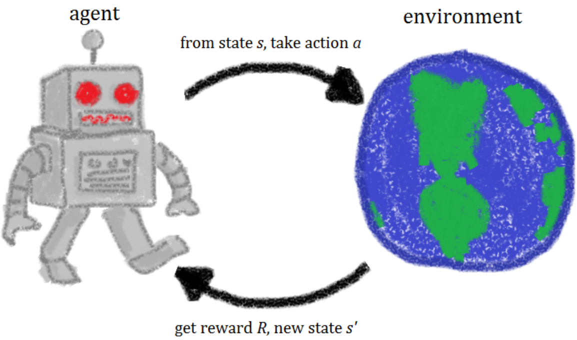


Figure 2: In Markov Decision problems, the goal is to find a controller $\pi(s)$ that maximizes the accumulation of rewards [Bel54].

$$V^{\pi}(\mathbf{s}) := \mathbb{E}_{\mathbf{s}'}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \pi(\mathbf{a}_t), \mathbf{s}'_t) \mid \mathbf{s}_0 = \mathbf{s}\right] \tag{1}$$

Parameterizing the Action-Value Function

- Action-value function, the accumulation of rewards given initial $\mathbf{s}, \mathbf{a}$

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) := \mathbb{E}_{\mathbf{s}'}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \pi(s_t), \mathbf{s}'_t) \mid \mathbf{s}_0 = \mathbf{s},\right] \qquad (2)$$

- Advantage Function, where $\max_{\mathbf{a}} A(\mathbf{s}, \mathbf{a}) = 0$ [Bai94]

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = V^{\pi}(\mathbf{s}) + A^{\pi}(\mathbf{s}, \mathbf{a}) \qquad (3)$$

Parameterizing the Action-Value Function

- Action-value function, the accumulation of rewards given initial $\mathbf{s}, \mathbf{a}$

$$Q^\pi(\mathbf{s}, \mathbf{a}) := \mathbb{E}_{\mathbf{s}'}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \pi(s_t), \mathbf{s}_t') \mid \mathbf{s}_0 = \mathbf{s},\right] \qquad (2)$$

- Advantage Function, where $\max_{\mathbf{a}} A(\mathbf{s}, \mathbf{a}) = 0$ [Bai94]

$$Q^\pi(\mathbf{s}, \mathbf{a}) = V^\pi(\mathbf{s}) + A^\pi(\mathbf{s}, \mathbf{a}) \qquad (3)$$

- Parameterizing the advantage function as a quadratic function yields computational savings [GLSL16]

$$Q(\mathbf{s}, \mathbf{a}) = V(\mathbf{s}) - \frac{1}{2}(\mathbf{a} - \pi(\mathbf{s}))L^T(\mathbf{s})L(\mathbf{s})(\mathbf{a} - \pi(\mathbf{s})) \qquad (4)$$

Model:

- $V(\mathbf{s})$ - value of state $\mathbf{s}$
- $\pi(\mathbf{s})$ - policy at state $\mathbf{s}$
- $L(\mathbf{s})$ - curvature of the advantage at $\mathbf{s}$

Bellman Error

- Bellman optimality equation [BS04]:

$$Q^*(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}'}[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')] \tag{5}$$

- To find the optimal policy, we seek to satisfy (5) for all state-action pairs, yielding the cost functional:

$$J(V, \pi, L) = \mathbb{E}_{\mathbf{s},\mathbf{a}}(y(\mathbf{s}, \mathbf{a}) - Q(\mathbf{s}, \mathbf{a}))^2, \tag{6}$$

  where $y(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}'}[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V(\mathbf{s}')]$.

- Finding the Bellman fixed point reduces to the stochastic program:

$$V^*, L^*, \pi^* = \arg \min_{V, \pi, L \in \mathcal{B}(\mathcal{S})} J(V, \pi, L) \ . \tag{7}$$

Reproducing Kernel Hilbert Spaces (RKHS)

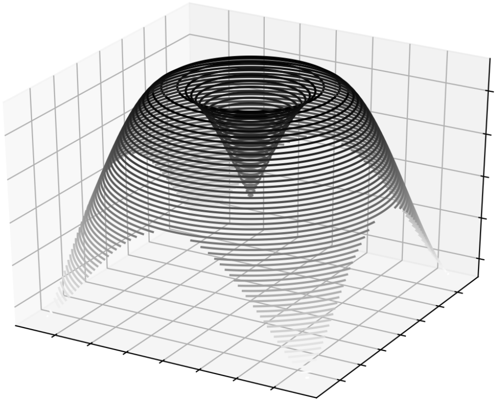


Figure 3: **Goal**: Approximate a smooth function via samples

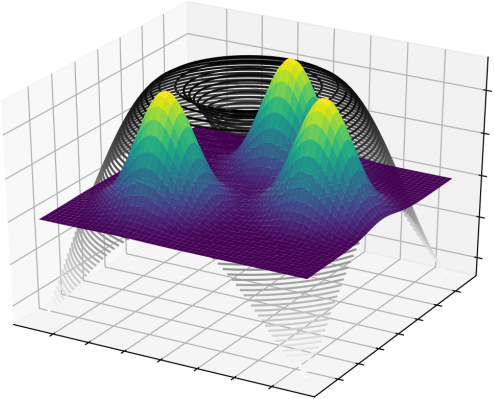


Figure 4: **Method**: Gradient descent in the RKHS.

Reproducing Kernel Hilbert Spaces (RKHS)

- We restrict $\mathcal{B}(\mathcal{S})$ to be a reproducing Kernel Hilbert space (RKHS) $\mathcal{H}$ to which V, π and L belong [KTSR17].
- An RKHS over $\mathcal{S}$ is a Hilbert space is equipped with a reproducing kernel, an inner product-like map $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ [NK09, AMP09]:

$$\text{(i)}\langle \pi, \kappa(\mathbf{s}, \cdot)\rangle_{\mathcal{H}} = \pi(\mathbf{s}), \qquad \text{(ii)}\mathcal{H} = \text{span}\{\kappa(\mathbf{s}, \cdot)\} \tag{8}$$

- A continuous function over a compact set may be approximated uniformly by a function in a RKHS equipped with a universal kernel [MXZ06].
- We use the Gaussian kernel with constant diagonal covariance Σ

$$\kappa(\mathbf{s}, \mathbf{s}') = \exp\{-\frac{1}{2}(\mathbf{s} - \mathbf{s}')\Sigma(\mathbf{s} - \mathbf{s}')^T\} \tag{9}$$

Stochastic Gradient Descent in the RKHS

- **Goal**: Learn V, π and L using samples $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$
- **Solution**: Stochastic semi-gradient descent [SB18] uses the directional derivative of the loss where the target value y_t is fixed:

$$y_t := r_t + \gamma V_t(\mathbf{s}'_t) \tag{10}$$

- Temporal difference: $\delta_t := y_t - Q_t(\mathbf{s}_t, \mathbf{a}_t)$
- We obtain the stochastic functional semi-gradients of the loss $J(V, \pi, L)$ via the reproducing property of the RKHS:

$$\hat{\nabla}_V J(V, \pi, L) = -\delta_t \kappa(\mathbf{s}_t, \cdot) \tag{11}$$

$$\hat{\nabla}_\pi J(V, \pi, L) = -\delta_t L(\mathbf{s}_t) L(\mathbf{s}_t)^T (\mathbf{a}_t - \pi_t(\mathbf{s}_t)) \kappa(\mathbf{s}_t, \cdot)$$

$$\hat{\nabla}_L J(V, \pi, L) = \delta_t L(\mathbf{s}_t)^T (\mathbf{a}_t - \pi_t(\mathbf{s}_t))(\mathbf{a}_t - \pi_t(\mathbf{s}_t))^T \kappa(\mathbf{s}_t, \cdot)$$

- The optimal V, π and L functions in the RKHS are of the form:

$$V(\mathbf{s}) = \sum_{n=1}^{N} purple w_{Vn} \kappa(\mathbf{s}_n, \mathbf{s}), \quad \pi(\mathbf{s}) = \sum_{n=1}^{N} \mathbf{w}_{\pi n} \kappa(\mathbf{s}_n, \mathbf{s}), \quad L(\mathbf{s}) = \sum_{n=1}^{N} \mathbf{w}_{Ln} \kappa(\mathbf{s}$$

Model Learning

Algorithm 1 Q-Learning with Kernel Normalized Advantage Functions

Input: l_0, $\{\alpha_t, \beta_t, \zeta_t, \epsilon_t, \Sigma_t\}_{t=0,1,2...}$

1: $V_0(\cdot) = 0, \pi_0(\cdot) = 0, L_0(\cdot) = l_0 I, \rho_0(\cdot) = 0$

2: **for** $t = 0, 1, 2, \ldots$ **do**

3: Obtain trajectory $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ where $\mathbf{a}_t \sim \mathcal{N}(\pi_t(\mathbf{s}_t), \Sigma_t)$

4: Compute the target value and Bellman error

$$y_t = r_t + \gamma V_t(\mathbf{s}'_t), \quad \delta_t = y_t - Q_t(\mathbf{s}_t, \mathbf{a}_t)$$

5: Compute the stochastic estimates of the gradients of the loss

$$\hat{\nabla}_V J(Q_t) = -\delta_t \kappa(\mathbf{s}_t, \cdot),\ \hat{\nabla}_\pi J(Q_t) = -\delta_t L(\mathbf{s}_t) L(\mathbf{s}_t)^T (\mathbf{a}_t - \pi_t(\mathbf{s}_t)) \kappa(\mathbf{s}_t, \cdot),$$

$$\hat{\nabla}_L J(Q_t) = \delta_t L(\mathbf{s}_t)^T (\mathbf{a}_t - \pi_t(\mathbf{s}_t))(\mathbf{a}_t - \pi_t(\mathbf{s}_t))^T \kappa(\mathbf{s}_t, \cdot)$$

6: Update V, π, L, ρ:

$$V_{t+1} = V_t - \alpha_t \hat{\nabla}_V J(Q_t), \quad \pi_{t+1} = \pi_t - \beta_t \hat{\nabla}_\pi J(Q_t),$$

$$L_{t+1} = L_t - \zeta_t \hat{\nabla}_L J(Q_t), \quad \rho_{t+1} = \rho_t + \kappa(\mathbf{s}_t)$$

7: Obtain greedy compression of V_{t+1}, π_{t+1}, L_{t+1}, ρ_{t+1} via KOMP

8: **end for**

9: **return** V, π, L

Model Composition

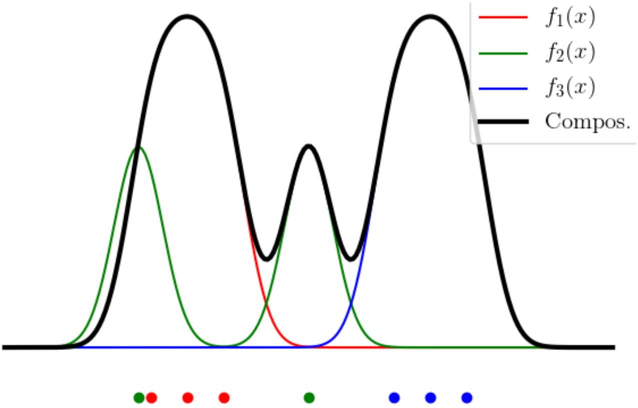


Figure 5: **Goal**: Compose multiple models off-line.

Model Composition

Given: N models π_i each trained on $D_i = \{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)\}_{t=1,\ldots N_i}$

Goal: Fit Π, which performs as well as π trained on $\bigcup_{i=1}^{N} D_i$

- Interpolate among π_i to get Π by setting $\Pi(\mathbf{s}) = \pi_i(\mathbf{s})$, $\forall \mathbf{s}$

Challenge: Policies π_i can disagree for $\mathbf{s} \in \mathcal{S}$

- While training π_i, count the number of training samples around **s** to evaluate the support of the model at **s**:

$$\rho_{i,t+1}(\mathbf{s}) = \rho_{i,t}(\mathbf{s}) + \kappa(\mathbf{s}_t, \mathbf{s}) \tag{12}$$

- For every $\mathbf{s} \in \mathcal{S}$, choose the policy with the highest density of training samples, $\rho_i(\mathbf{s})$
- For our application, we use the kernel density of π_i without explicitly fitting ρ_i

$$\tilde{\rho}(\pi_i, \mathbf{s}) = \sum_{\mathbf{s}_k \in \pi_i} \kappa(\mathbf{s}_k, \mathbf{s}) \tag{13}$$

Model Composition

Algorithm 2 Composition with Conflict Resolution

Input: $\{\pi_i(\mathbf{s}) = \sum_j^{M_i} w_{ij}\kappa(\mathbf{s}, \mathbf{s}_{ij}),\ \rho_i(\mathbf{s}) = \sum_j^{M_i} v_{ij}\kappa(\mathbf{s}, \mathbf{s}_{ij})\}_{i=1,2\dots,N}$, ϵ

Initialize $\Pi(\cdot) = 0$, append centers $D = [\mathbf{s}_{11}, \dots, \mathbf{s}_{ij}, \dots]$
for each $\mathbf{s}_{ij} \in D$ chosen uniformly at random **do**
    if $\rho_i(\mathbf{s}_{ij}) > \max_{k \neq i} \rho_k(\mathbf{s}_{ij})$ **then**
        $\Pi = \Pi(\cdot) + (\pi_i(\mathbf{s}_{ij}) - \Pi(\mathbf{s}_{ij}))\kappa(\mathbf{s}_{ij}, \cdot)$
    end if
end for
Obtain compression of π using KOMP with ϵ
return f

- **State**: 5 range readings from LIDAR at at an angular interval of 34° with a field of view of 170°
- **Action**: angular velocity of the Scarab robot, $a \in [-0.3, 0.3]$ rad/s
- **Reward**:

$$r(s) = \begin{cases} -200, & \text{if collision} \\ +1, & \text{otherwise} \end{cases}$$

- Sensor readings are received and controls are issued at 10 Hz
- Constant forward velocity of 0.15 m/s
- YouTube Video

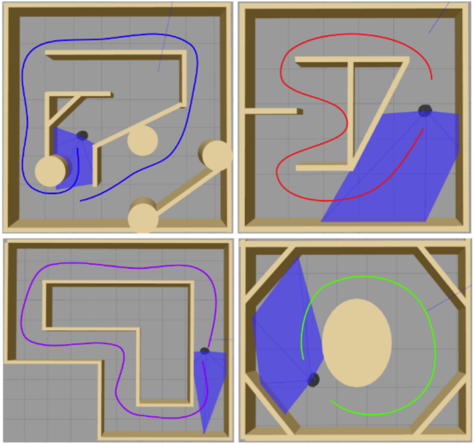


Figure 6: Four environments were simulated using Gazebo for training and testing.

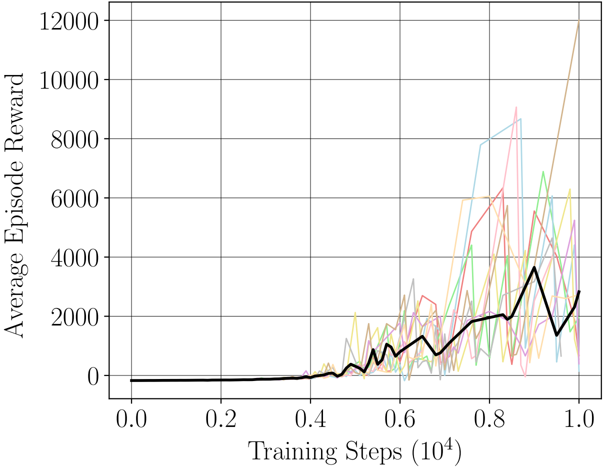


Figure 7: Reward averaged over 10 trials in the Round environment (black)

Simulation Results - Bellman Error

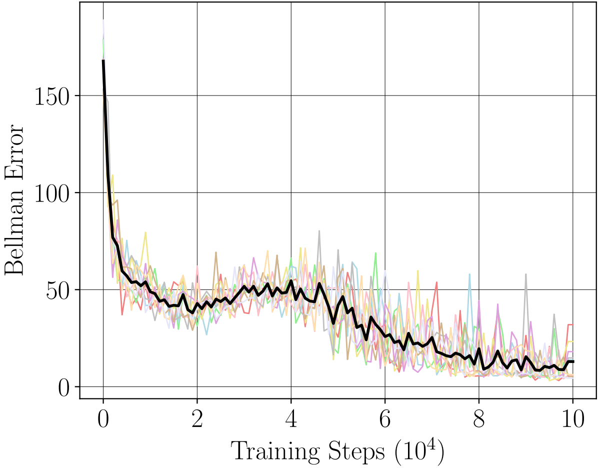


Figure 8: Training loss averaged over 10 trials in the Round environment (black)

Simulation Results - Model Composition

Policies / Reward	Round	Maze	Circuit 2	Circuit 1
1 - Round	**1000**	-11663	-608	-608
2 - Maze	**1000**	**1000**	-5	-407
3 - Circuit 2	**1000**	-11663	**1000**	196
4 - Circuit 1	**1000**	-11462	-407	**1000**
1 / 2	**1000**	**1000**	-5	-206
1 / 3	**1000**	-11663	799	-206
1 / 4	**1000**	-11261	-206	799
2 / 3	**1000**	**1000**	**1000**	-5
2 / 4	**1000**	**1000**	-5	799
3 / 4	**1000**	-11462	397	397
1 / 2 / 3	**1000**	**1000**	799	196
1 / 2 / 4	**1000**	**1000**	-5	**1000**
1 / 3 / 4	**1000**	-11663	397	799
2 / 3 / 4	**1000**	**1000**	799	-206
1 / 2 / 3 / 4	**1000**	**1000**	**1000**	598

Table 1: Composability results

Conclusion and Future Work

- Contributions
  - Stochastic gradient descent algorithm for RL in RKHS
  - Formulation of the problem of **composable learning**
  - Policy **composition** algorithm
- Future Work
  - Use deep dimensionality reduction techniques for image data
  - Extend to partially observable environments

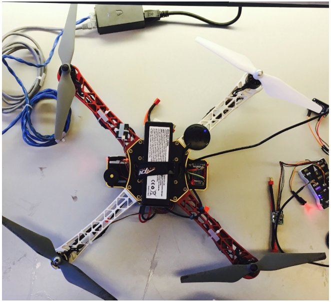


Figure 9: Control of multiple quadrotors based on image data

References I

Andreas Argyriou, Charles A Micchelli, and Massimiliano Pontil, *When is there a representer theorem? vector versus matrix regularizers*, Journal of Machine Learning Research **10** (2009), no. Nov, 2507–2529.

Leemon C Baird, *Reinforcement learning in continuous time: Advantage updating*, Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, vol. 4, IEEE, 1994, pp. 2448–2453.

Richard Bellman, *The theory of dynamic programming*, Tech. report, DTIC Document, 1954.

Dimitir P Bertsekas and Steven Shreve, *Stochastic optimal control: the discrete-time case*, 2004.

Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine, *Continuous deep q-learning with model-based acceleration*, CoRR **abs/1603.00748** (2016).

- Alec Koppel, Ekaterina Tolstaya, Ethan Stump, and Alejandro Ribeiro, *Nonparametric stochastic compositional gradient descent for q-learning in continuous markov decision problems*, IEEE Transactions on Automatic Control (under preparation) (2017).
- Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang, *Universal kernels*, Journal of Machine Learning Research **7** (2006), no. Dec, 2651–2667.
- Vladimir Norkin and Michiel Keyzer, *On stochastic optimization and statistical learning in reproducing kernel hilbert spaces by support vector machines (svm)*, Informatica **20** (2009), no. 2, 273–292.
- Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press Cambridge, 2018.