Learning Robot Skills with Temporal Variational Inference Supplement

A. Appendix

In this document, we provide additional details and insights into our approach.

A.1. Derivation of Temporal Variational Inference:

While our main paper presents the specific temporal variational inference that we use to learn policies, the notion of temporal variational inference is more generally applicable to sequential data with hidden states (such as in HMMs).

We present a detailed derivation of this general temporal variational inference objective, and contrast it with standard variational inference below. We then provide a detailed derivation of the gradient of this objective in our case, explaining how dependencies on system dynamics may be factored out.

We begin by considering observed sequential data $x = \{x_t\}_{t=1}^{t}$ (in our case, this corresponds to state-action tuples, i.e. $\{x_t = (s_t, a_t)\}_{t=1}^{T}$), interacting with a sequence of unobserved latent variables $y = \{y_t\}_{t=1}^{T}$ (in our case, y_t is simply ζ_t). For these sequences of data, the true likelihood of observed data $\mathcal{L} = \mathbb{E}_{x\sim p(x)}\big[\log p(x)\big]$ is lower bounded by J, where:

$$J = \mathbb{E}_{x\sim p(x)}\big[\log p(x)\big] - D_{KL}\big[q(y|x)||p(y|x)\big]$$

where $q(y|x)$ is a variational approximation to the true conditional $p(y|x)$, and the lower bounded due to the non-negativity of KL divergence. Expanding the KL divergence term, and using the fact that $p(x, y) = p(y|x)p(x)$, we have:

$$J = \mathbb{E}_{x\sim p(x)}\big[\log p(x)\big] - \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log \frac{q(y|x)}{p(y|x)}\Big]$$

$$J = \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log p(x, y) - \log q(y|x)\Big]$$

Standard variational inference then decomposes the joint likelihood $p(x, y)$ into a "decoder" $p(x|y)$ and a prior $p(y)$:

$$J_{\text{StandardVI}} = \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log p(x|y) + \log p(y) - \log q(y|x)\Big]$$

Given the sequential nature of x and y, inferring the conditional $p(x|y)$ does not provide a useful insight into how the *sequence* of x and y will evolve, and is often difficult to learn. In contrast, our temporal variational inference makes use of the following decomposition of joint likelihood $p(x, y)$:

$$p(x, y) = \underbrace{\prod_{t=1}^{T} p(x_t|x_{1:t-1}, y_{1:t-1})}_{= p(x||y)} \underbrace{\prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1})}_{= p(y||x)}$$

$p(x||y)$ and $p(y||x)$ are *causally conditioned distributions*, i.e. they only depend on information available until the current timestep. $p(x||y)$ and $p(y||x)$ are formally defined as $\prod_{t=1}^{T} p(x_t|x_{1:t-1}, y_{1:t-1})$ and $\prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1})$ respectively (Ziebart et al., 2013). We can plug this decomposition into the objective J to give us the temporal variational inference objective:

$$\begin{aligned} J_{\text{TemporalVI}} &= \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log p(x||y) + \log p(y||x) - \log q(y|x)\Big] \\ &= \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log \prod_{t=1}^{T} p(x_t|x_{1:t-1}, y_{1:t-1}) + \log \prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1}) - \log q(y|x)\Big] \\ &= \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\sum_{t=1}^{T} \log p(x_t|x_{1:t-1}, y_{1:t-1}) + \sum_{t=1}^{T} \log p(y_t|x_{1:t}, y_{1:t-1}) - \log q(y|x)\Big] \end{aligned}$$

A.2. Derivation of Gradient Update:

Now that we have a better understanding of the origin of the temporal variational inference objective, we present a derivation of the gradient to this objective in our case. In our option learning setting, the TVI objective is:

$$J_{\text{TemporalVI}} = \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_{t=1}^{T}\big\{\log\eta(\zeta_t|s_{1:t},a_{1:t-1},\zeta_{1:t-1}) + \log\pi(a_t|s_{1:t},a_{1:t-1},\zeta_{1:t}) + \log p(s_{t+1}|s_t,a_t)\big\} + \log p(s_1) - \log q(\zeta|\tau)\Big]$$

Assuming distributions π, η, and q are parameterized by θ, ϕ, and ω respectively, the gradient of this objective is:

$$\nabla_{\theta,\phi,\omega} J_{\text{TemporalVI}} = \nabla_{\theta,\phi,\omega}\,\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_{t=1}^{T}\big\{\log\eta(\zeta_t|s_{1:t},a_{1:t-1},\zeta_{1:t-1}) + \log\pi(a_t|s_{1:t},a_{1:t-1},\zeta_{1:t}) + \log p(s_{t+1}|s_t,a_t)\big\} + \log p(s_1) - \log q(\zeta|\tau)\Big]$$

Note that the system dynamics $p(s_{t+1}|s_t, a_t)$ and the initial state distribution $p(s_1)$ are both independent of the parameterization of networks θ, ϕ, and ω. The gradient of the objective with respect to θ, ϕ, and ω can be separated into additive terms that depend on the dynamics and initial state distributions, and terms that depend on networks π, η, and q. The expectation of the dynamics and initial state distribution terms are constant, and their gradient hence vanishes:

$$\nabla_{\theta,\phi,\omega} J_{\text{TemporalVI}} = \nabla_{\theta,\phi,\omega}\ \underbrace{\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_{t=1}^{T}\log p(s_{t+1}|s_t,a_t) + \log p(s_1)\Big]}_{\text{Constant}} = 0$$

The gradient update of temporal variational inference hence doesn't depend on the dynamics and the initial state distribution, leading to the following gradient update, as presented in the main paper in equation 3:

$$\nabla_{\theta,\phi,\omega} J = \nabla_{\theta,\phi,\omega}\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_t \log\pi(a_t|s_{1:t},a_{1:t-1},\zeta_{1:t}) + \sum_t \log\eta(b_t,z_t|s_{1:t},a_{1:t-1},\zeta_{1:t-1}) - \log q(\zeta|\tau)\Big]$$

A.3. Implementation Details:

We make note of several implementation details below, such as network architectures and hyperparameter settings.

A.3.1. Network Architecture:

We describe the specific architectures of each of the networks q, π, and η in our approach. The base architecture for each of these three networks is an 8 layer LSTM with 128 hidden units in each layer. We found that an 8 layer LSTM was sufficiently expressive for represent the distributions q, π, and η.

The space of predictions for q and η are the binary termination variables b_t at every timestep t, and the continuous parameterization of options z. q and η are thus implemented with two heads on top of the final LSTM layer.

The first head predicts termination probabilities (from which termination variables b_t are sampled), and consists of a linear layer of output size 2 followed by a softmax layer to predict probabilities.

The second head of the network predicts the latent z parameterization at each timestep. It consists of two separate linear layers above the final LSTM layer, that predict the mean and variance of a Gaussian distribution respectively. The mean predictions do not use an activation layer. The variance predictions employ a SoftPlus activation function to predict positive variances. The dimensionality of latent z's is 64 across all three datasets and across all networks.

The prediction space for π is the continuous low-level actions (i.e. joint velocities). Similar to the z prediction, this is implemented by with 2 separate linear layers to predict the mean and variances of a Gaussian distribution, from which actions are drawn. As above, variances use a SoftPlus activation, while mean predictions are done without an activation. The dimensions of the action space are 16 for the MIME dataset, 8 for the Roboturk dataset, and 66 for the Mocap dataset.

A.3.2. HYPERPARAMETERS:

We provide a list of the hyperparameters and their values, and other training details used in our training procedure.

1. Optimizer: We use the Adam optimizer (Kingma & Ba, 2014) to train all networks in our model.
2. Learning Rate: We use a learning rate of 10^{-4} for our optimizer, as is standard.
3. Epsilon: The exploration parameter ϵ is used in our training procedure to both scale perturbation of latent z's sampled from our model, as well as to explore different latent b's in an epsilon-greedy fashion. We use an initial ϵ value of 0.3, and linearly decay this value to 0.05 over 30 epochs of training, and found this works well. We considered a range of $0.1 - 0.3$ for the initial value of epsilon, and a range of $0.05 - 0.1$ for the final epsilon.
4. Ornstein Uhlenbeck Noise Parameters: Our DDPG implementation for the RL training uses the Ornstein Uhlenbeck noise process, with parameters identical to those used in the DDPG paper (Lillicrap et al., 2015).
5. Loss Weights: In practice, the various terms in our objective are reweighted prior to gradient computation, to facilitate learning the desired behaviors and to prevent particular terms from dominating others.
   (a) Option likelihood weight: During initial phases of training, we reweight the option likelihood term $\sum_t \log \eta(\zeta_t | s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})$ in our gradient update by a factor of 0.01, to prevent the variational network from getting inconsistent gradients from the randomly initialized η. Once the variational policy has been trained sufficiently, we set the weight of this option likelihood to 1.
   (b) KL Divergence weight: We reweight the KL divergence term, as done in the β-VAE paper (Higgins et al.), by a factor of 0.01.

A.3.3. RL DETAILS:

For our reinforcement learning experiments, we use variants of the following Robosuite (Mandlekar et al., 2018) environments to evaluate our approach:

- SawyerPickPlace - An environment where a sawyer robot grasps and places objects in specific positions in respective bins. We use 4 variants of this task, SawyerPickPlaceBread, SawyerPickPlaceCereal, SawyerPickPlaceCan, SawyerPickPlaceMilk, where the objective is to place the corresponding object into the correct bin.
- SawyerNutAssembly - An environment where a sawyer robot must pick a nut up and place it around an appropriately shaped peg. We use 2 variants of this task, SawyerNutAssemblySquare and SawyerNutAssemblyRound, where the shape of the nut and peg are varied.

The 3 baseline algorithms specified in the main paper and our approach all share the same policy architectures (i.e., an 8 layer LSTM with 128 hidden units) for both low-level policies (in all baselines and our approach) and high-level policies (our approach and the hierarchical RL baseline).

For these pick-place and nut-assembly tasks, the information available to the policies are the sequence of joint states of the robot, previous joint velocities executed, the `robot-state` provided by Robosuite (consists of sin and cos of the joint angles, gripper positions, etc.), and the `object-state` provided by Robosuite (consisting of absolute positions of the target objects, object positions relative to the robot end-effector, etc.). The output space for the policies is always the joint velocities (including the gripper).

A.3.4. DATASET DETAILS:

Regarding the CMU Mocap dataset, the data used in this project was obtained from mocap.cs.cmu.edu, the database was created with funding from NSF EIA-0196217.

References

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mandlekar, A., Zhu, Y., Garg, A., Booher, J., Spero, M., Tung, A., Gao, J., Emmons, J., Gupta, A., Orbay, E., Savarese, S., and Fei-Fei, L. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. The principle of maximum causal entropy for estimating interacting processes. *IEEE Transactions on Information Theory*, 59(4):1966–1980, April 2013. ISSN 1557-9654. doi: 10.1109/TIT.2012.2234824.