Case-Based Off-Policy Evaluation Using Prototype Learning (Supplementary material)

Anton Matsson[1] **Fredrik D. Johansson**[1]

[1]Chalmers University of Technology

A THE PROTOTYPE MODEL

In Section 3.1, we only briefly described the objective function

$$J(\Theta) = \mathrm{NLL}(\mathcal{D}; \Theta) + \lambda_d R_d(\Theta) + \lambda_c R_c(\Theta) + \lambda_e R_e(\Theta),$$

where $\mathrm{NLL}(\mathcal{D}; \Theta)$ is the negative log-likelihood, $R_d(\Theta)$, $R_c(\Theta)$ and $R_e(\Theta)$ are regularization terms and λ_d, λ_c and λ_e are regularization parameters. Θ denotes the set of model parameters, i.e., the parameters of the encoding network e, the weights B, c and the prototypes $\tilde{H}$. For a given dataset $\mathcal{D} = ((h^1_{t_1}, a^1_{t_1}), \ldots, (h^m_{t_m}, a^m_{t_m}))$, drawn according to a distribution p_μ, the NLL loss of the estimate $\hat{p}_\mu$, parameterized in Θ, is defined as

$$\mathrm{NLL}(\mathcal{D}; \Theta) = -\frac{1}{m} \sum_{i=1}^{m} \log\left(\hat{p}_\mu(A_t = a^i_{t_i} \mid H_t = h^i_{t_i})\right).$$

Furthermore, the regularization terms are defined as follows (see Ming et al. [2019] for further details):

- The **diversity** regularization

  $$R_d(\Theta) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \max\left(0, d_{\min} - d(\tilde{z}_i, \tilde{z}_j)\right)^2,$$

  where $d(z, z') = \|z - z'\|_2$, penalizes latent prototypes that are too close to each other. The parameter $d_{\min}$ is a tunable hyperparameter in our experiments.

- The **clustering** regularization

  $$R_c(\Theta) = \sum_{h \in \mathcal{D}} \min_i d(\tilde{z}_i, e(h))^2$$

  encourages the encoded histories to approach the most similar latent prototypes, which creates a clustering structure in the latent space.

- The **evidence** regularization

  $$R_e(\Theta) = \sum_{i=1}^{n} \min_{h \in \mathcal{D}} d(\tilde{z}_i, e(h))^2$$

  encourages the latent prototypes to approach the encodings that are most similar to them.

A.1 PROTOTYPE VALUE

In Section 3.3, we showed how the prototypes can be used to compute the value of a policy π for prototype j at time t, $V_{j,t}(\pi)$. Below, we derive a statistical estimator for $V_{j,t}(\pi)$ using observations under μ. First, we have

$$\begin{aligned} V_{j,t}(\pi) &:= \mathbb{E}_\pi\left[\sum_{t' \geq t} R_{t'} \;\middle|\; J_t = j\right] \\ &= \mathbb{E}_\pi\left[\frac{p(J_t = j \mid H_t)}{p_\pi(J_t = j)} \sum_{t' \geq t} R_{t'}\right]. \end{aligned}$$

This equation follows from the fact that J_t is conditionally independent of all other variables given H_t. Now, with W importance weights for π and μ,

$$V_{j,t}(\pi) = \mathbb{E}_\mu\left[\frac{p(J_t = j \mid H_t)}{p_\pi(J_t = j)} W \sum_{t' \geq t} R_{t'}\right].$$

Following standard definitions,

$$p_\pi(J_t = j) = \mathbb{E}_\pi[p(J_t \mid H_t)],$$

which may be identified using importance sampling,

$$p_\pi(J_t) = \mathbb{E}_\pi[p(J_t \mid H_t)] = \mathbb{E}_\mu[p(J_t \mid H_t) W_t],$$

with $W_t = \prod_{t'=0}^{t} \frac{p_\pi(A_{t'} \mid H_{t'})}{p_\mu(A_{t'} \mid H_{t'})}$. Hence, we may estimate

$$\hat{p}_\pi(J_t = j) = \frac{1}{m} \sum_{i=1}^{m} p(J_t = j \mid H_t = h^i_t) w^i_t.$$

For trajectories i which end before t, we let $\hat{p}(J_t = j \mid H_t = h^i_t) = 0$.

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

A.2 IS THERE A GOOD PROTOTYPE MODEL?

In Section 3.1, we asked the question: Assuming that adjusting for the history H_t is sufficient for unbiased policy evaluation, do there exist prototype histories $\tilde{H}$, an encoding e and a similarity function s such that evaluation using the prototype model is accurate? Here follows an example when this is provably the case.

Consider the history at the first time step, $H_0 = X_0 \in \mathbb{R}^d$. Assume that for each action a, the distribution of histories in which action a is taken is isotropic Gaussian, $(H_0 \mid A_0 = a) \sim \mathcal{N}(\mu_a, \gamma^2)$. Then, the joint distribution of (H_0, A_0) is a Gaussian mixture model (GMM) with components identified by the actions, $a = 1, \dots, k$. We have by the definition of the GMM that

$$p(A_0 = a \mid H_0 = h) \propto e^{-\frac{\|h-\mu_a\|_2^2}{\gamma^2}} = s(\mu_a, h),$$

where s is defined as in (3). As a result, with the component means $\tilde{X} = [\mu_1, \dots, \mu_k]$ as prototypes and $S(\tilde{X}, h) = [s(\mu_1, h), \dots, s(\mu_k, h)]^\intercal$, the behavior policy is given by $p(A_0 \mid H_0 = h) = S(\tilde{X}, h)/(\sum_a s(\tilde{x}_a, h))$, which matches (4)with $B = 1/(\sum_a s(\tilde{x}_a, h))$ and $c = 0$, up to the application of the softmax function. Furthermore, the prototype assignment probability in (8)is equal to the behavior policy. We state a generalization below.

Theorem 1. *Assume that there exists a bijective, differentiable encoding function* $e : \mathcal{H} \to \mathcal{Z}$ *such that* $\forall t : (e(H_t), A_t) \sim \mathrm{GMM}$ *with stationary component means* $\{\mu_a\}_{a=1}^k$ *and variance* γ^2*. Then, with prototypes* $\tilde{H} = [e^{-1}(\mu_1), \dots, e^{-1}(\mu_k)]^\intercal$*, and* S *as defined above,*

$$\forall t : p(A_t \mid H_t = h) \propto S(e(\tilde{H}), e(h)).$$

Proof. Due to bijectivity, $p(A_t \mid H_t = h) = p(A_t \mid e(H_t) = e(h))$. The final result follows from the same argument as for the special case of H_0 above. $\square$

Theorem 1 shows that there are indeed problems for which a prototype model that exactly describes the behavior policy *exists*. This also implies that there exists a prototype estimate of the value $V(\pi)$ which is unbiased. However, it does not give guarantees for recovering such a model from data, or that the training set contains samples which act well as prototypes. Learning encoding functions e which satisfy the conditions of Theorem 1 has been studied in the context of normalizing flows [Kong and Chaudhuri, 2020, Rezende and Mohamed, 2015].

B EXPERIMENTAL DETAILS

The prototype approach for off-policy evaluation was evaluated on real-world sepsis data extracted from the MIMIC-III database [Johnson et al., 2016]. In addition, a synthetic environment for sepsis management, provided by Oberst and Sontag [2019], was used to study the bias induced by prototypes as a function of the trajectory length. In this section, we give further details about the experiments. To produce the results presented in this paper, we needed about 750 core-hours of computational time. The neural networks were implemented in PyTorch [Paszke et al., 2019] and trained on GPU (Nvidia Tesla T4) using the skorch framework [Tietz et al., 2017]. Other models were implemented using scikit-learn [Pedregosa et al., 2011].

B.1 USING DATA FROM MIMIC-III

We extracted the dataset of sepsis patients from the MIMIC-III database using the code provided by Komorowski et al. [2018].[1] This dataset contains the features listed in Supplementary Table 2 in Komorowski et al. [2018] as well as the total fluid intake and the total urine output for each patient. We also built the AI Clinician using the code provided by Komorowski et al. [2018]. To evaluate the 500 candidate policies, we used only the MIMIC-III test data and not data from the eICU Research Institute Database.

We used the train-test split associated with the best performing candidate policy in our experiments. We trained and evaluated the estimators of the behavior policy using a subset of the available features: heart rate, systolic blood pressure, diastolic blood pressure, mean blood pressure, shock index, hemoglobin, BUN, creatine, urine output over 4 hours, pH, base excess, bicarbonate, lactate, PaO_2/FiO_2 ratio, age, Elixhauser index and SOFA score. In addition, we included the treatment dose of vasopressors and IV fluids, respectively, over the previous 4 hours. These values were set to 0 at the initial time steps.

For ProNet and ProSeNet, we selected parameters of the diversity regularization $(d_{\text{min}}, \lambda_d)$ by performing 3-fold cross-validation over a grid of points in the parameter space $\{1, 2, 3, 4, 5\} \times \{0.00001, 0.0001, 0.001, 0.01, 0.1\}$. These parameters were optimized for each combination of prototypes n and prediction prototypes q in our experiments. The parameters λ_c and λ_e were set to 0.001, and we performed the projection step, see (6), every fifth epoch.

For LR and RF, we searched for optimal models using 3-fold cross-validation, considering the following parameter values:

- LR: regularization: $\{\mathrm{L1}, \mathrm{L2}\}$; regularization strength: 10 values spaced evenly on a log-scale from 1×10^{-4} to 1×10^4;
- RF: maximum tree depth: $\{5, 10, 15, 20, \mathrm{None}\}$.

To create the model based on post-hoc clustering of en-

[1] The original code is available at `https://github.com/matthieukomorowski/AI_Clinician`.

codings, we performed K-means clustering to cluster the encodings of the converged RNN model into 10 clusters. We used the encodings that had the shortest Euclidean distance to the cluster centroids as "prototypes" and trained a logistic regression model on the similarity vector between encodings and "prototypes".

We estimated the value of the target policies—the AI Clinician and the zero-drug policy—by performing weighted importance sampling of the test set trajectories. In accordance with Komorowski et al. [2018], we used a final reward $r^i = \pm 100$ based on the survival of the patient. For all estimators of the behavior policy, we excluded a small number of sequences whose WIS weight exceeded 100.

B.2 USING THE SEPSIS SIMULATOR

To estimate the bias induced by prototypes as a function of trajectory length, we utilized the sepsis simulator provided by Oberst and Sontag [2019].[2] We used the full state representation consisting of five discrete variables—a binary diabetes indicator and four ordinal-valued vital signs (heart rate, systolic blood pressure, blood glucose level and blood oxygen level)—as well as the previously administered treatments. There is a probability of 0.2 that a randomly initialized patient has diabetes. Furthermore, there are three binary treatment variables—antibiotics, vasopressors and ventilation—resulting in an 8-dimensional action space. A simulated patient is discharged if the patient has normal vitals and is not given any treatments; discharge results in a positive reward. Death is associated with a negative reward and occurs if at least three vitals are abnormal. If neither discharge nor death has occurred at the end of the sequence, the reward is zero. We refer to the code for details about, for example, the levels of the vitals and the transition probabilities of the Markov decision process (MDP).

For the experiment presented in Figure 8, we used the notebook `learn_mdp_parameters` provided by Oberst and Sontag [2019] to estimate the true parameters of the MPD. We then learned an optimal behavior policy using policy iteration. The policy was softened so that all actions had a nonzero probability of being chosen in each state. We generated trajectories of varying length from this policy to estimate the effect of bias. Some trajectories ended prematurely due to discharge or death and we therefore ensured that the number of collected state-action pairs were roughly the same for all sequence lengths. Specifically, for each sequence length, we generated $20\,000$ state-action pairs for training and sigmoid calibration of the estimators $\hat{\mu}$, respectively.

Since we used the full state representation, we made the Markov assumption and estimated the behavior policy using a vanilla FNN and FNN-based prototype models. We trained all models over 30 epochs, using a batch size of 128. Otherwise we used the same architectures and settings as in the main experiment with the MIMIC-III data.

The collected training set was also used to estimate the MDP parameters and learn a target policy π using policy iteration. Again, this policy was softened to avoid zero probabilities. We generated an additional evaluation set of the same size as the training and calibration sets. We repeated the process of data collection and learning 100 times. To produce Figure 8, we removed a small fraction of samples for which the weight ratio exceeded 10^3.

[2]The simulator is publicly available at `https://github.com/clinicalml/gumbel-max-scm/tree/sim-v2`.

References

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11): 1716–1720, 2018.

Zhifeng Kong and Kamalika Chaudhuri. The expressive power of a class of normalizing flow models. In *International Conference on Artificial Intelligence and Statistics*, pages 3599–3609. PMLR, 2020.

Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 903–913, 2019.

Michael Oberst and David Sontag. Counterfactual off-policy evaluation with gumbel-max structural causal models. In *International Conference on Machine Learning*, pages 4881–4890. PMLR, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer,

R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Marian Tietz, Thomas J. Fan, Daniel Nouri, Benjamin Bossan, and skorch Developers. *skorch: A scikit-learn compatible neural network library that wraps PyTorch*, jul 2017. URL `https://skorch.readthedocs.io/en/stable/`.