Python Repo: `https://github.com/XanderJC/medkit-learn`

A Domains and Real Data Information

A.1 Hospital Wards.

The *Ward* domain is based on the care of 6,321 patients at the Ronald Reagan UCLA Medical Center in California who were treated on the general medicine floor between 2013-2016 [4]. These patients were treated for a variety of conditions including pneumonia, sepsis, and fevers were in general stable and deterioration that required ICU care was rare. Measurements were taken roughly every 4 hours, with average stays lasting 9 days, and include common vital signs such as pulse and blood pressure alongside lab tests and results and include 8 static (although categorical features are one-hot encoded, extending the space) and 35 temporal features for which we model the dynamics. The action space is taken as a choice of one to three binary actions (encoded for a maximum size of eight) marking the treatment of various oxygen therapy devices.

A.2 Intensive Care Unit.

The *ICU* domain simulates the treatment of 23,106 of patients in the intensive care unit from Amsterdam UMC [19]. These patients are in a more critical state than those on the general wards while suffering similarly from a variety of conditions and consequently are monitored more frequently, with the database containing around 1 billion clinical observations at varying timesteps down to minute by minute recordings. We aggregate data into one hour timesteps and model the treatment of mechanical ventilation alongside the prescription of antibiotics and oxygentherapy. We include 36 static features including height, initial weight, and commorbidities with 24 series features focusing on vital signs including heart rate, blood pressure, and various chemical blood concentration levels.

B Further Environment Model Details

B.1 The Customised State Space Model

The CSS is a variety of a deep non-Markovian hidden state-space model. The *non-Markovianity* comes from the property in the model that transitions are parameterised given discrete states simply as follows:

$$\begin{aligned} p(z_t|\vec{x}_{t-1}, \vec{y}_{t-1}, \vec{z}_{t-1}) &= p(z_t|\vec{\alpha}_t, \vec{z}_{t-1}) \\ &= \sum_{t'=1}^{t-1} \alpha_{t'}^{t-1} \mathbf{P}_{y_{t'}}(z_{t'}, z_t), \end{aligned} \tag{7}$$

with $\mathbf{P}_{y_t}$ a baseline transition matrix given intervention y_t is made and $\alpha_{t'}^{t-1}$ some attention weight on the previous timesteps such that $\sum_{t'=1}^{t-1} \alpha_{t'}^{t-1} = 1$. This departs from the traditional IOHMM [57] by the inclusion of the attention weights that induce time-dependency on points further in the past than the previous time point and can be learnt during training.

The emission distribution allows for a flexible representation: let $p_\psi(x_t)$ be any distribution with support over $\mathcal{X}$ and parameter(s) ψ (for example some Gaussian mixture) then we let:

$$p(x_t|z_t, x_s) = p_{\psi^*}(x_t), \quad \text{with } \psi^* = f_\gamma(z_t, x_s). \tag{8}$$

We take f_γ to be a γ-parameterised function approximator to output the parameters of the emission distribution given the current state and static features of the patient - a standard choice being an MLP that takes in the concatenation of z_t and x_s. This alleviates a common problem with state-space models where the observations are ultimately drawn from some finite mixture of distributions of order $|\mathcal{Z}|$, as now the dependence on the static features allows for a very flexible output. The CSS dynamics model allows Medkit users to post-hoc customise the number of states and the Markovianity of the environment through the attention mechanism (e.g users can pass a vector that specifies exact weights or an integer representing the number of states back to look.)

Learning and inference. Exact inference over the hidden states in this model is intractable and so we use an inference network to parameterise an approximate posterior over the latent hidden state, in particular assuming a factorisation that mirrors the true structure of the posterior as follows:

$$q_\phi(\vec{z}_T|\vec{x}_T, \vec{y}_T) = q_\phi(z_1|\vec{x}_T, \vec{y}_T) \prod_{t=2}^{T} q_\phi(z_t|z_{t-1}, \vec{x}_{t:T}, \vec{y}_{t:T}). \tag{9}$$

Thus the posterior depends on the previous hidden state and all future observations and actions. This can be practically achieved by using a backward LSTM [59] to summarise the future into its hidden state before passing that and the previous state into a new network to obtain the approximate posterior.

Our inference network leads to a factorised Evidence Lower BOund (ELBO) given by:

$$\log p_\theta(\vec{x}_T|\vec{y}_{T-1}) \geq \mathbb{E}_{q_\phi}\big[\log p_\theta(\vec{x}_T, \vec{z}_T|\vec{y}_{T-1}) - \log q_\phi(\vec{z}_T|\vec{x}_T, \vec{y}_T)\big], \tag{10}$$

leading to an optimisation task over θ and ϕ. This is readily done using stochastic variational inference whereby we take a Monte Carlo estimate of the ELBO by sampling from the posterior and taking gradients [38].

B.2 The Sequential VAE Model

The SVAE is again a variety of a deep non-Markovian hidden state-space model, although now the states are no longer discrete but rather live in some latent space of $\mathbb{R}^d$. This then allows for more flexibility in the transition dynamics, in particular by making use of neural architectures. An encoder network predicts the approximate posterior over the latent variables and we employ essentially the same method as for teacher forcing in order to model dynamics in the latent space. This again means that an inference network is required for doing approximate inference and similarly leads to an ELBO optimisation scheme. With a joint optimisation scheme, we learn a representation that generates the observations well but also captures the features relevant for the transitions. This expressiveness allows for a higher fidelity model than the custom state-space but however comes at the cost of interpretable structure which we have established may be useful should algorithms be designed to uncover such things.

C Further Experimental Details

C.1 Computation

All experiments were performed on a 2016 MacBook Pro, using a 2.9 GHz Dual-Core Intel Core i5 with 8GB of LPDDR3 RAM and no GPU acceleration.

C.2 Predictive Score

The predictive score measures how well the synthetic data can be used as a direct substitute for real data when it comes to a prediction task.

For our experiments we generate 500 sample trajectories of maximum length 30 to be used as the synthetic data. A network consisting of an LSTM layer followed by two fully connected layers is then trained on this data for 50 epochs. This is then applied to a real data set and the AUROC is reported.

C.3 Discriminative Score

The discriminative score measures how well the synthetic data 'hides' amongst real data.

For our experiments a training dataset is generated using 100 real trajectories and 100 synthetic trajectories along with their associated real/synthetic label. A single layer LSTM is then trained on this dataset to predict the label, before being applied to a test set of 100 real trajectories and 100 synthetic trajectories. We report the absolute value of the test accuracy minus 0.5.

C.4 Imitation Learning Benchmarks

All methods used are based on neural network and so in the experiments we maintain the same architecture of 2 hidden layers of 64 units each connected by exponential linear unit (ELU) activation functions.

Publicly available code was used in the implementations of a number of the benchmarks, specifically:

- VDICE [37]:
  `https://github.com/google-research/google-research/tree/master/value_dice`

- EDM [31]:
  `https://github.com/wgrathwohl/JEM`

Note that VDICE was originally designed for continuous actions with a Normal distribution output which we adapt for the experiments by replacing with a Gumbel-softmax.

D Frequently Asked Questions

Q: What would the process be for outside contributors wanting to add new medical domains?

A: This should be relatively easy. The first way would be for them to provide us with a copy of the medical data, we could then add the details and train the models before adding them to the package. If they would rather not share the data, then the second way would be for the outside contributors themselves to fork the repo, define the domain, and train the models using the codebase before making a pull request to us. We would then make sure the pre-trained models work appropriately before merging into the package.

Q: Could Medkit not generate biased or inappropriately skewed data?

A: It is definitely possible, as with any generative model, however, we have inspected all of the models to try to ensure that this is not happening. We would emphasize that the data generated by Medkit should not be used for scientific discovery though - for example, it should not be used to say anything about how real doctors treat patients in the ICU nor how real patients will react to treatment. It is rather intended to be used with a focus on inspecting algorithms for decision modelling.

Q: How important is the use of differential privacy in the training of the models?

A: We would like to emphasize that all of the real data used to train models within Medkit had already undergone a full pre-processing procedure from their respective data guardians to de-identify the data and ensure the privacy of individuals. As a consequence, the added differential privacy layer that we add is not essential but does serve as an extra "insurance policy" just in case.

Q: Can Medkit be used effectively for traditional model based reinforcement learning?

A: It is worth saying that model-based RL is certainly not the focus of this work, as we're much more interested in decision modelling based on the policy of a demonstrator. There are no prescribed rewards in Medkit at the moment - so agents cannot be trained out-of-the-box, however a reward function can be manually specified and integrated if the user so wishes.

References

[4] Alaa, A. M., Yoon, J., Hu, S., and Van der Schaar, M. (2017). Personalized risk scoring for critical care prognosis using mixtures of gaussian processes. *IEEE Transactions on Biomedical Engineering*, 65(1):207–218.

[57] Bengio, Y. and Frasconi, P. (1995). An input output hmm architecture. In *Advances in neural information processing systems*, pages 427–434.

[19] Elbers, P. W. G. (2019). AmsterdamUMCdb v1.0.2 ICU database.

[59] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[31] Jarrett, D., Bica, I., and van der Schaar, M. (2020). Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33.

[37] Kostrikov, I., Nachum, O., and Tompson, J. (2019). Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations*.

[38] Krishnan, R., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.