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.

As described in the main text, the *deep* aspect is manifested in the emission probability distribution whose parameters are predicted from the output of a neural network that takes the static features and current state.

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:

$$q_\phi(\vec{z}_T|\vec{x}_T, \vec{y}_T) = q_\phi(z_1|\vec{x}_T, \vec{y}_T) \prod_{t=2}^{T} q_\phi(z_t|z_{t-1}, \vec{x}_{t:T}, \vec{y}_{t:T}). \tag{8}$$

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 [27] 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:

$$\log p_\theta(\vec{x}_T|\vec{y}_{T-1}) \geq \mathbb{E}_{q_\phi}\big[\log p_\theta(\vec{x}_T, \vec{z}_T|\vec{y}_{T-1}) - \log q_\phi(\vec{z}_T|\vec{x}_T, \vec{y}_T)\big], \tag{9}$$

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 again means that an inference network is required for doing approximate inference and similarly leads to an ELBO optimisation scheme.

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.

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.

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

[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.