RECAP OF APPROACHES

Sara Silvestrelli
Dipartimento di Economia, Metodi Quantitativi e Strategie di Impresa
Università degli studi di Milano-Bicocca

ABSTRACT

The approach chosen to evaluate DRL techniques in DTR contexts is a Deep Q-Network using three type of Neural Network: RNN, LSTM and DNN. The RL technique used as a benchmark is instead QL via OLS. Several strategies have been used for DRL methods, which are discussed in more detail in the papers *Exploration Policies* and *Target Network Updates*.

***K*eywords** RL · DRL · DTR

1 Q-learning

The approach chosen as a benchmark is the RL method of Q-learning where the reward is used as an outcome to predict the Q function in the previous months by updating the values backwards as explained in Algorithm 1 (Chapter 3.1 [1]).

Algorithm 1: Q-learning with Q functions approximation

Initialization: Set the Q-function after the final month to be zero: $\hat{Q}^{\pi}_{T+1} \leftarrow 0$
Set each state S_t to be composed at most by $(S_t,\ S_{t-1},\ S_{t-2},\ S_{t-3})$
for t in $T, ..., 1$ **do**:
Estimate the reward that would occur when following the optimal policy:
$r_{t+1} = r_{t+1} + \max_{A_{t+1}} \hat{Q}^{\pi}_{t+1}(A_{t+1}, S_{t+1})$
Estimate the Q-function, $\hat{Q}^{\pi}_{t}(A_t, S_t)$, for the preceding month using this reward as the outcome (e.g. through OLS).
Estimate the optimal treatment(s) for each patient at month t:
$\hat{A}^{*}_{t} \leftarrow\ argmax_{A_t}\ \hat{Q}_t(A_t, S_t)$
end for

1.1 Additions to Bibliographical Reference

Q-functions are conditional expectations making standard regression techniques applicable. For that reason one of the most popular method for estimating Q-functions in the health sciences is ordinary least squares [2]. Therefore, OLS is used instead of the Random Forest applied by Kyle Humphrey's simulation study.

In addition to this, the state space includes the basic information of a patient's clinical picture in a four memory window.

2 DRL

Recently the DRL is gaining ground in the medical field and there are increasing studies on this subject where the DQN is trained both on simulated patients, such as in the case of Murphy et al. [3] or on actual patient observation data as in Liu et al [4].

Recurrent networks have also been applied in pharmacodynamic contexts, although their applications are limited in the literature so far [5].

For the prevoius reasons, the networks used to highlight the efficiency of DRL methods in DTR contexts are: DNN, RNN, LSTM. The basic algorithm used for all three is the same (Algorithm 2).

Algorithm 2: DQN with experience replay

Initialization: replay memory $\mathcal{D}$ to capacity N , action-value function Q with random weights θ, target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

for episode=1,M **do**:
  Initialize state s_t
  for t=1,T **do**:
    With probability ϵ select a random action a_t
    otherwise select $a_t = max_a \, Q^*(s_t, a; \theta)$
    Execute action a_t in emulator and observe reward r_{t+1} and state s_{t+1}
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
    Set $s_{t+1} = s_t$

    Sample random minibatch of transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ from $\mathcal{D}$
    Set

$$y_j = \begin{cases} r_{j+1} & \text{for terminal state} \\ r_{j+1} + \gamma \, max_{a'} \, \hat{Q}(s_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$$

    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$

    Every C steps reset $\hat{Q} = Q$ i.e., set $\theta^{-} = \theta$
  end for
end for

2.1 Neural Networks Architecture and Implementation

2.1.1 Structure

DNN code

```
init_model = function(){
      model <- keras_model_sequential()
      model %>%
        layer_dense(units = 32, input_shape=self$input_shape, activation = 'relu') %>%

        layer_dense(units = 64, activation = 'relu') %>%

        layer_dense(units = self$output_dim)


      model %>% compile(
        loss = 'mse',
        optimizer = optimizer_adam(1e-03),
        metrics = c("mse")
      )
      return(model)
    }
```

RNN code

```
init_model = function(){
      model <- keras_model_sequential()
      model %>%
        layer_simple_rnn(units = 32,
                         return_sequences = TRUE,
```

```
                    input_shape = self$input_shape
                    ) %>%
    layer_simple_rnn(units = 32
                    ) %>%
    layer_dense(units = self$output_dim)

  model %>% compile(
    loss = 'mse',
    optimizer = optimizer_adam(1e-03),
    metrics = c("mse")
    )
  return(model)
  }
```

LSTM code

```
init_model = function(){
    model <- keras_model_sequential()
    model %>%
      layer_lstm(units = 32,
                 return_sequences=T,
                 input_shape = self$input_shape
                 ) %>%
      layer_lstm(units = 32
                 ) %>%
      layer_dense(units = self$output_dim)

    model %>% compile(
      loss = 'mse',
      optimizer = optimizer_adam(1e-03),
      metrics = c("mse")
      )
    return(model)
    }
```

2.1.2 Settings

- The output dimensions of For the neural networks is equal to the K treatment options. The learning rate η was set to be $1e-3$ using Adam Optimizer.
- A maximum capacity of $20,000$ was chosen for the memory buffer.
- One episode corresponds to one simulated patient.
- Evaluation on 500 new patients.
- $\gamma = 0.999$[1]

The parameters for the exploration (Epsilon-Greedy Policy, Softmax Policy) and target update (Hard Update, Soft Update) strategies are listed in the respective papers.

2.2 Additions to Bibliographical References

Both studies ([3], [4]) were taken as main references for the present work using both the experiential replay technique with a batch of size 32 and target network during training.

In addition to these works the state space includes the basic information (tumor mass, toxicity, stress) plus those features available in the scenario included in a four memory window as for the QL approach.Given the simulative nature of the data and consequently the a priori knowledge, a longer time window in the case of recurring networks would not have been fruitful.

[1]Each reward is weighted by a value $\gamma \in [0, 1]$ called *discount factor* which is higher the more the reward it is associated with is recent. On the opposite, the lower the value of γ the faster future rewards are valued. When $\gamma = 0$ we are only interested in immediate reward which overshadows future rewards ("myopic" evaluation), otherwise when $\gamma = 1$ future or immediate reward are equally prioritized ("far-sighted" evaluation). This function allows us to define the importance of evaluating a reward delayed in time.

In addition to the original algorithm, several new combinations of exploration strategies (Epsilon-Greedy Policy, Softmax Policy) and target update strategies (Hard Update, Soft Update) were tried out. They are discussed in the papers *Exploration Policies* and *Target Network Updates*.

References

[1] Kyle Humphrey. Using reinforcement learning to personalize dosing strategies in a simulated cancer trial with high dimensional data. The University of Arizona, 2017.

[2] Bibhas Chakraborty and Susan A. Murphy. Dynamic treatment regimes. *Annual Review of Statistics and Its Application*, 1(1):447–464, 2014.

[3] Brian Murphy, Mustafa Nasir-Moin, Grace von Oiste, Viola J. Chen, Howard A. Riina, Douglas Kondziolka, and Eric Karl Oermann. Patient level simulation and reinforcement learning to discover novel strategies for treating ovarian cancer. *ArXiv*, abs/2110.11872, 2021.

[4] Ning Liu, Ying Liu, Brent Logan, Zhiyuan Xu, Jian Tang, and Yanzhi Wang. Learning the dynamic treatment regimes from medical registry data through deep q-network. *Scientific Reports*, 9(1), December 2019.

[5] Xiangyu Liu, Chao Liu, Ruihao Huang, Hao Zhu, Qi Liu, Sunanda Mitra, and Yaning Wang. Long short-term memory recurrent neural network for pharmacokinetic-pharmacodynamic modeling. *International Journal of Clinical Pharmacology and Therapeutics*, 59(2):138–146, February 2021.