Reinforcement learning in clinical medicine: a method to optimize dynamic treatment regime over time

Zhongheng Zhang; written on behalf of AME Big-Data Clinical Trial Collaborative Group

Department of Emergency Medicine, Sir Run Run Shaw Hospital, Zhejiang University School of Medicine, Hangzhou 310016, China
Correspondence to: Zhongheng Zhang. Department of Emergency Medicine, Sir Run Run Shaw Hospital, Zhejiang University School of Medicine, No. 3, East Qingchun Road, Hangzhou 310016, China. Email: zh_zhang1984@zju.edu.cn.

Abstract: Precision medicine requires individualized treatment regime for subjects with different clinical characteristics. Machine learning methods have witnessed rapid progress in recent years, which can be employed to make individualized treatment regime in clinical practice. The idea of reinforcement learning method is to take action in response to the changing environment. In clinical medicine, this idea can be used to assign optimal regime to patients with distinct characteristics. In the field of statistics, reinforcement learning has been widely investigated, aiming to identify an optimal dynamic treatment regime (DTR). Q-learning is among the earliest methods to identify optimal DTR, which fits linear outcome models in a recursive manner. The advantage is its easy interpretation and can be performed in most statistical software. However, it suffers from the risk of misspecification of the linear model. More recently, some other methods not so heavily depend on model specification have been developed such as inverse probability weighted estimator and augmented inverse probability weighted estimator. This review introduces the basic ideas of these methods and shows how to perform the learning algorithm within R environment.

Keywords: Reinforcement learning; Q learning; dynamic treatment regime (DTR)

Submitted May 18, 2019. Accepted for publication Jun 16, 2019.
doi: 10.21037/atm.2019.06.75
View this article at: http://dx.doi.org/10.21037/atm.2019.06.75

Introduction

Precision medicine is an important concept in the modern era, which dictates that the treatment regime should be individualized depending on genetic background, demographics and clinical characteristics. Many clinical conditions can be extremely heterogeneous that the treatment regime is highly variable. For example, sepsis is a heterogeneous syndrome which is characterized by different clinical outcomes and responses to fluid resuscitation (1). Treatment regimes should be individualized for different subphenotypes. Even for a certain subphenotype, the treatment regime can evolve over time. Thus, individualized dynamic treatment regime should be adopted to achieve the best potential outcome. Reinforcement learning (RL) is a powerful method in computer science that has been successfully used to teach an agent to learn to interact with the environment, aiming to achieve the best reward/return. This is quite similar to the situation in clinical practice, where a doctor needs to adopt an action (treatment regime) depending on a patient's conditions (e.g., demographics, genetic background, clinical characteristics and previous response to a specific treatment regime). Thus, reinforcement learning can be used to solve a clinical decision problem, whereby the concept of precision medicine can be realized. In this review article, we will introduce (I) the concept of reinforcement learning, (II) how this concept can be adopted to clinical research, and (III) how to perform RL using R language.

Basic ideas behind reinforcement learning

The key elements of a RL system include a policy, a reward signal, a value function and a model of the environment (*Figure 1*). A policy is usually a function mapping from a state to an action. A policy defines how an agent takes

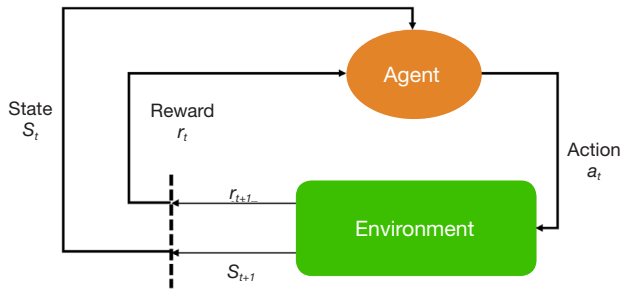

Figure 1 Schematic illustration of reinforcement learning system.

action in response to the environment. A reward defines the goal of a RL problem. Every time (t) when an agent takes an action (a_t) on the environment, the environment sends a reward signal (r_{t+1}) to the agent. The goal of the agent is to maximize the reward by taking an appropriate action. While the reward signal indicates the immediate benefit after taking an action, the value function defines the long-term benefit. The value of a state is the total amount of reward an agent can expect to accumulate over time, starting from that state. For example, when a doctor resuscitates a patient with septic shock, the action space comprises vasoactive agents and fluid infusion, and the immediate reward can be a stabilized blood pressure, or a normalized serum lactate (2). Since the policy is a mapping from the state to an action, the doctor will make treatment decision depending on the environment state of the patient. The mortality outcome at hospital discharge can be considered as a final reward. The state space of septic shock can be any combinations of hemodynamics, electrolytes, serum lactate and demographics. However, this can be an extremely large space due to the curse of dimensionality, and dimension reduction is usually required (3).

However, several key assumptions that are commonly used in RL in computer science may not be applicable in clinical trials. First, many RL algorithms such as dynamic programming requires that the complete knowledge of the environment is known (based on physical laws and domain knowledges), which however is usually unsatisfied in clinical data. In the septic shock example, the underlying mechanisms behind hospital mortality cannot be represented by a deterministic equation and must be estimated from samples. Second, the data collection can be cheap and easy to perform in some RL examples such as the blackjack, in which tens of thousands of episodes of the game can be simulated (i.e., the data generating mechanism is known). This is not true in clinical setting because the sample size is usually limited and collecting data is expensive (e.g., also for the confidential issues). Of note, the state space and action space can be larger than the sample size in most situations, and thus parametric models are usually required to model the data. Third, the Markov property is assumed to be true in computer science, which is defined that the probability of each possible value for S_t and R_t depends only on the immediately preceding state (S_{t-1}) and action (A_{t-1}), but not on earlier states and actions. However, there is no such assumption from the pathophysiology that earlier conditions have no causal association with final clinical outcome. Thus, the notion of history must be introduced in solving clinical problems. The history at a given stage includes all the present and past states, plus the present action. As a result, the history space grows exponentially with the number of stages (4).

Working example

A simulated dataset is created for the illustration purpose.

```
> set.seed(123)
> n=500
> for (i in 1:3) {
    assign(paste('x1',i,sep = ''),rnorm(n))
}
> A1linpred <- exp(-0.1+0.5*x11+0.5*x12)
> A1pro <- A1linpred/(1+A1linpred)
> A1 <- rbinom(n,1,prob = A1pro)
> A1opt <- (x11 > -0.54)*(x12 < 0.54)
> x21 <- 0.8*x11 + 0.6*A1 + rnorm(n)
> x22 <- 0.8*x12 - 0.6*A1 + rnorm(n)
> x23 <- 0.8*x13 + 0.7*A1 + rnorm(n)
> A2linpred <- exp(-0.1+0.5*x21+0.5*x22)
> A2pro <- A2linpred/(1+A2linpred)
> A2 <- rbinom(n,1,prob = A2pro)
> A2opt <- (x21 > 0.3)*(x23 < 0.46)
> y <- 2 + 0.25*x11 + 0.25*x12 -
          0.25*x13 - 0.5*(A1-A1opt)^2 -
          (A2-A2opt)^2 + rnorm(n)
> dt <- data.frame(x11=x11,x12=x12,x13=x13,
                   x21=x21,x22=x22,x23=x23,
                   A1=A1, A2=A2,
                   y=y)
> head(round(dt,2))
    x11   x12   x13   x21   x22   x23   A1   A2   y
```

1	−0.56	−0.60	−1.00	−1.15	−1.73	−1.91	0	0	3.44
2	−0.23	−0.99	−1.04	1.30	−1.51	−1.02	1	1	0.55
3	1.56	1.03	−0.02	1.71	−1.19	1.56	1	1	−0.73
4	0.07	0.75	−0.13	−1.06	−1.38	1.76	0	0	1.96
5	0.13	−1.51	−2.55	0.56	−0.42	−2.16	0	1	2.22
6	1.72	−0.10	1.04	3.50	0.22	1.64	1	1	1.02

As noted above, a data frame is generated with two stages. On stage 1, there are three covariates denoted as x11, x12 and x13; and treatment was A1. On stage 2, the three covariates are x21, x22 and x23, and the treatment is A2. Finally, we observe an outcome which is a continuous variable. Higher values indicate a better outcome. For stage 1, the optimal decision rule dictates that A1 should take value 1 for subjects with x11 >-0.54 and x12 <0.54. However, clinicians choose A1 value by a logit function (the propensity function) in clinical practice. Thus, the A1 used in real world practice may not equal the value adopted by the optimal rule. The purpose of RL is to learn the optimal rule to maximize the outcome Y. In the same vein, the optimal decision rule for A2 is to assign A2=1 to subjects with x21 >0.3 and x23 <0.46. However, the exact form of the outcome function is not easy to specify but can be approximated.

$$\begin{aligned} &Y=2+0.25\cdot x11+0.25\cdot x12-0.25\cdot x13 \\ &-0.5\cdot[A1-I(x11>-0.54)\times I(x12<0.54)]^2 \\ &-[A2-I(x21>0.3)\times I(x23<0.46)]^2 \end{aligned} \quad [1]$$

There is an indicator function I(·) which returns 1 when the express is true, and 0 otherwise. The quadratic function suggests that the outcome Y can only be maximized when A1 and A2 is equal to the value determined by the optimal decision rule in both stage 1 and 2.

Q-learning algorithm

Q-learning is a temporal difference control algorithm that can be used to estimate optimal dynamic treatment regime from longitudinal clinical data. Suppose there are two stages, the Q function can be defined as (5):

$$Q_2(h_2,a_2)\triangleq E(Y\mid H_2=h_2,A_2=a_2) \quad [2]$$

$$Q_1(h_1,a_1)\triangleq E(\max_{a_2\in\{-1,1\}} Q_2\left(h_2,a_2\right)\mid H_1=h_1,A_1=a_1) \quad [3]$$

The Q function at t=2 measures is the value function of assigning a_2 to a patient with history h_2. In a similar sense, the Q function at t=1 measures the quality of assigning a_1 to a patient with history h_1, assuming an optimal regime at stage 2 is adopted. The first step of Q learning is the regress Y on H_1 and A_2 to obtain estimators of $\hat{\beta}_2$:

$$\hat{Q}_2(H_2,A_2,\beta_2)\triangleq H_{20}^T\beta_{20}+A_2H_{21}^T\beta_{21} \quad [4]$$

Where H_2 includes history variables up to t=2; H_{20} is the main effect model (treatment-free) consisting a subset of variables in H_2. Similarly, H_{21} is also a subset of variables of H_2 for the contrast function. A contrast function defines how the expected outcome varies depending on treatment regime. The only means by which the treatment can influence outcome are via the contrast function (6).

The second step is to maximize the Q_2 function by varying A2. In the above example, if A2 takes value {-1, 1}, we can define $\tilde{Y}\triangleq\max_{a_2\in\{-1,1\}}\hat{Q}_2\left(H_2,a_2,\beta_2\right)$, which gives $\tilde{Y}=H_{20}^T\hat{\beta}_{20}+\left|H_{21}^T\beta_{21}\right|$. $\tilde{Y}$ is the potential outcome when optimal regime is adopted at stage 2. Finally, the estimated $\tilde{Y}$ is regressed on H_1 and A_2 to obtain $\hat{\beta}_1$ in the model

$$\hat{Q}_1(H_1,A_1,\beta_1)\triangleq H_{10}^T\beta_{10}+A_1H_{11}^T\beta_{11} \quad [5]$$

The optimal regime at stage one can be obtained by maximizing the above equation.

Q-learning with R

In the example, the *DynTxRegime* package (v4.1) will be used to obtain optimal regime at each stage. The package *iqLearn* is also available in R that performs Q-learning algorithm (5).

```
> library(DynTxRegime)
> moMain <- buildModelObj(model = ~x11+x12+x13+
      x21+x22+x23,
      solver.method = 'lm')
> moCont <- buildModelObj(model = ~x21+x22+x23,
      solver.method = 'lm')
```

The above code creates two components (e.g. the main effect and the contrast components) of the outcome model. Note that all covariates are specified to be predictive on the outcome, and only covariates collected at stage two determine the optimal decision rule. However, the functional form is mis-specified in the above equation.

```
> fitSS <- qLearn(moMain = moMain,
          moCont = moCont,
          data = dt, response = y,
```



```
        txName = 'A2')
```

The second stage can be fit by using the above code. We can create a confusion matrix to see whether the learning algorithm can assign optimal regime to more subjects in the training set, than that assigned by the physician (e.g., suppose the longitudinal data were collected from electronic healthcare records and the treatment was determined by physicians).

```
> A2Qlearn <- optTx(fitSS)$optimalTx
> table(A2, A2opt)
                A2opt
A2              0       1
0               217     45
1               167     71

> table(A2Qlearn, A2opt)
                A2opt
A2Qlearn        0       1
0               357     45
1               27      71
```

The result shows that the Q-learning method is not able to assign optimal regime to more subjects than that assigned by physicians. In the observational cohort (the first matrix), 217 patients received A2=0, and 71 received A2=1, which are optimal regimes. The Q-learning algorithm correctly assigned A2=0 to 357 patients, and A2=1 to 71 patients. Other patients fail to receive the optimal regime if treatment regime is determined by the Q-learning algorithm. This is due to the misspecification of the outcome linear function, which has been widely discussed in the literature (7,8).

Q-learning is implemented through a backward recursive fitting procedure based on a dynamic programming algorithm. The previous step fitted a linear regression model for stage two, and next we will regress the estimated outcome $\tilde{y}$ on covariates (X1) and treatment (A1) at stage one.

```
> moMain <- buildModelObj(model = ~x11+x12+x13,
                solver.method = 'lm')
> moCont <- buildModelObj(model = ~x11+x12+x13,
                solver.method = 'lm')
> fitFS <- qLearn(moMain = moMain, moCont = moCont,
        data = dt, response = fitSS,
        txName = 'A1')
```

The specification of the regression model is similar to that at for stage two except that the response argument takes an object returned by previous fit. The coefficient for the first stage model can be examined by the following code:

```
> coef(fitFS)
$outcome
$outcome$Combined
(Intercept)     x11             x12             x13
1.592159131     0.156777481     0.188064560     -0.277882456
A1              x11:A1          x12:A1          x13:A1
-0.026082177    -0.014829966    0.000654683     0.007832428
```

Again, the confusion matrix can help to examine the correctness of Q-learning algorithm to assign optimal regime.

```
> A1Qlearn <- optTx(fitFS)$optimalTx
> table(A1,A1opt)
                A1opt
A1              0       1
0               141     140
1               95      124

> table(A1Qlearn, A1opt)
                A1opt
A1Qlearn        0       1
0               219     264
1               17      0
```

It appears that the Q-learning algorithm fails to assign optimal regime to more subjects than the physicians' judgement. However, the linear model cannot correctly capture the function form as that for data generation.

Inverse probability weighted estimator (IPWE)

To circumvent the risk of model misspecification as demonstrated above, the inverse probability weighted estimator for the population mean outcome can be employed (9). The estimator for $E\{Y^*(g_\eta)\}$ is:

$$\mathrm{IPWE}(\eta) = n^{-1}\sum_{i=1}^{n}\frac{C_{\eta,i}\bullet Y_i}{\pi_c(X_i;\eta,\hat{\gamma})} \quad [6]$$

Where $C_\eta = A\cdot g(X,\eta)+(1-A)\cdot[1-g(X,\eta)]$, and $g(X,\eta)\in G_\eta$

is a specific treatment regime conditional on covariate X and coefficient η. The equation indicates that the observed outcome Y (under treatment A) is the counterfactual outcome $Y^*(g_\eta)$ (under treatment $g(X,\eta)$) when C_η=1. Otherwise, $Y^*(g_\eta)$ is not observed because a patient actually received treatment A is not equal to $g(X,\eta)$. Suppose the treatment regime can only takes value 0 or 1. $\pi_c(X_i;\eta,\hat{\gamma})$ is the probability of C_η=1 giving covariates X:

$$\Pr(C_\eta = 1|X) = \pi_c(X,\eta,\hat{\gamma}) = \pi(X,\hat{\gamma})\cdot g(X,\eta) + [1-\pi(X,\hat{\gamma})]\cdot[1-g(X,\eta)] \quad [7]$$

The estimator IPWE(η) is actually a weighted average of observed outcome under decision rule $g(X,\eta)$. Observed outcome Y not following the decision rule was excluded (by the $C_{\eta,i}$ factor in the numerator) from the calculation. For patients who actually receive A=0, by substituting A=0 into the IPWE(η) equation, the contrast function can be written as:

$$\hat{C}_{IPWE}(X_i) = -\frac{[1-g(X,\eta)]\bullet Y_i}{[1-\pi(X,\hat{\gamma})]\bullet[1-g(X,\eta)]} = -\frac{Y_i}{[1-\pi(X,\hat{\gamma})]} \quad [8]$$

Similarly, the contrast function for A=1 can be written as:

$$\hat{C}_{IPWE}(X_i) = \frac{g(X,\eta)\bullet Y_i}{\pi(X,\hat{\gamma})\bullet g(X,\eta)} = \frac{Y_i}{\pi(X,\hat{\gamma})} \quad [9]$$

Collectively, the contrast function is written as:

$$\hat{C}_{IPWE}(X_i) = \frac{A_i\bullet Y_i}{\pi(X,\hat{\gamma})} - \frac{(1-A_i)\bullet Y_i}{[1-\pi(X,\hat{\gamma})]} \quad [10]$$

From the above discussion, we note that only the propensity model is required in the estimation, the mean outcome model is not required, which reduces the risk of model misspecification. We then define a function Z=I($C(X)$>0), so that Z=1 indicate subjects who will benefit from treatment A=1 than A=0. The optimization problem can be regarded as minimization of a weighted misclassification error:

$$g^{opt} = \arg\min_{g\in G} E\left\{|C(X)|\bullet\left[I\{C(X)>0\} - g(X)\right]^2\right\} = \arg\min_{g\in G}\sum_{i-1}^{n}\hat{W}\left[\hat{Z}_i - g(X_i)\right]^2 \quad [11]$$

Where $\hat{W}_i = |\hat{C}(X_i)|$ is an estimate of contrast function for each subject. The problem can be considered as a classification problem with $\hat{Z}_i$ as the binary response, X_i as the predictor and $\hat{W}_i$ as the weight. Classification and regression trees (CART) can be used to solve this problem (10).

The following code builds a propensity score model in which the variables x21 and x22 determine the choice of treatment regime. Since treatment A is a binary variable, a generalized linear model with the response variable following binomial distribution.

```
> moPropen <- buildModelObj(model = ~ x21+x22,
      solver.method = 'glm',
      solver.args = list('family'='binomial'),
      predict.method = 'predict.glm',
      predict.args = list(type='response'))
```

Then, the *rpart* package (version 4.1-13) is employed to solve the classification problem. The CART model uses x21, x22 and x23 as the predictors.

```
> library(rpart)
> moClass <- buildModelObj(model = ~x21+x22+x23,
      solver.method = 'rpart',
      solver.args = list(method="class"),
      predict.args = list(type='class'))
```

The second-stage analysis using IPW can be done with the following code.

```
> fitSS_IPW <- optimalClass(moPropen = moPropen,
      moClass = moClass,
      data = dt, response = y,
      txName = 'A2')
> library(rpart.plot)
> rpart.plot(classif(object = fitSS_IPW))
```

Recall that the optimal decision rule is to assign A=1 to subjects with x21 >0.3 and x23 <0.46. The CART assigns x21 ≥0.15 and x23 <0.49 to receive A=1; but there are still moderate misclassifications (*Figure 2*).

The confusion matrix can be obtained with the following code.

```
> A2IPW <- optTx(fitSS_IPW)$optimalTx
> table(A2,A2opt)
                A2opt
```


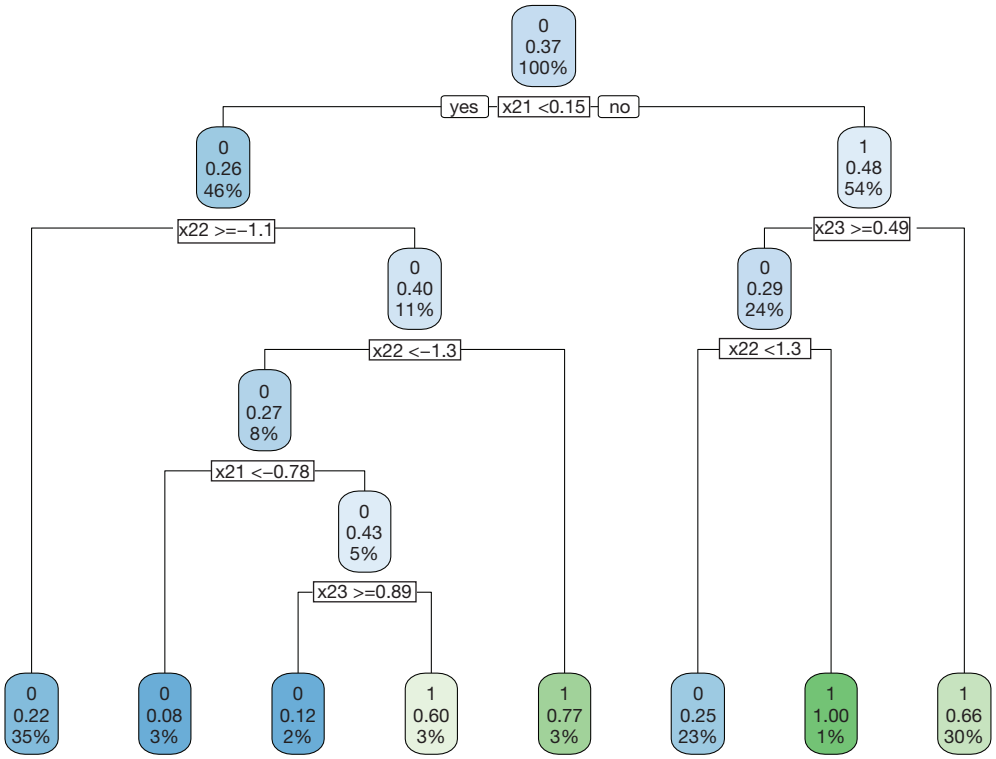

Figure 2 Decision tree to assign treatment regime A=1 or A=0 to subjects with covariates X2 at stage two by the IPWE method.

A2	0	1
0	217	45
1	167	71

> table(A2IPW,A2opt)

	A2opt	
A2IPW	0	1
0	334	0
1	50	116

The results showed that the IPW method can significantly improve the accuracy of assigning optimal regime to patients. There are only 50 patients who would benefit from A=0 but receive A=1 as determined by the IPW method.

Augmented Inverse Probability Weighted Estimators (AIPWE)

The IPWE method does not depend on parametric or semi-parametric at all, making it robust to model misspecification. However, the IPWE estimator of contrast values can be too noisy to successfully inform the classification regime. One method to circumvent the problem is to maximize on a doubly robust augmented inverse probability weighted estimator (AIPWE) for the population mean outcome over all possible regimes (9,11). The AIPWE for the mean outcome is given by:

$$\text{AIPWE}(\eta)=n^{-1}\sum_{i=1}^{n}\left\{\frac{C_{\eta,i}\bullet Y_i}{\pi_c(X_i;\eta,\hat{\gamma})}-\frac{C_{\eta,i}-\pi_c(X_i;\eta,\hat{\gamma})}{\pi_c(X_i;\eta,\hat{\gamma})}\bullet m\left(X_i;\eta,\hat{\beta}\right)\right\} \quad [12]$$

The $m(X_i;\eta,\beta)=\mu(1,X,\beta)\cdot g(X,\eta)+\mu(0,X,\beta)\cdot[1-g(X,\eta)]$ is a model for the potential outcome under treatment regime g_η that $E(Y^*(g_\eta)\,|\,X)=\mu(1,X)\cdot g(X,\eta)+\mu(0,X)\cdot[1-g(X,\eta)]$. AIPWE possesses the doubly robust property that the estimate is consistent for $E(Y^*(g_\eta))$ if either $\pi(X,\gamma)$ or $\mu(A,X,\beta)$ is correctly specified. The contrast function for AIPWE estimator can be derived from the equation:

$$\hat{C}_{AIPWE}(X_i)=\frac{A_i\bullet Y_i}{\pi(X,\hat{\gamma})}-\frac{(1-A_i)\bullet Y_i}{[1-\pi(X,\hat{\gamma})]}-\frac{A_i-\pi(X,\hat{\gamma})}{\pi(X,\hat{\gamma})}\bullet\mu\left(1,X,\hat{\beta}\right)-\frac{A_i-\pi(X,\hat{\gamma})}{[1-\pi(X,\hat{\gamma})]}\bullet\mu\left(0,X,\hat{\beta}\right) \quad [13]$$

The only difference between $\hat{C}_{AIPWE}(X_i)$ and $\hat{C}_{IPWE}(X_i)$ is that the former borrows information from specified

parametric regression model of outcome $\mu(A,X,\beta)$, trying to strike a balance between two extremes that are non-parametric or completely depending on the parametric model (11). R code to perform AIPWE is very similar to that for IPWE, but we need to define the parametric outcome model. The outcome model is defined as

```
> moMain <- buildModelObj(model =
~x21+x22+x23+x11+x12+x13,
            solver.method = 'lm')

> moCont <- buildModelObj(model = ~x21+x22+x23,
            solver.method = 'lm')
```

then the second stage for AIPWE analysis is:

```
> fitSS_AIPW <- optimalClass(moPropen = moPropen,
                 moMain = moMain, moCont = moCont,
                 moClass = moClass,
                 data = dt, response = y,
                 txName = 'A2')
```

Note that the only difference between AIPWE and IPWE is the use of parametric outcome model as specified by the *moMain* and *moCont* arguments. We know from the Q-learning section that the linear outcome model is mis-specified. The confusion matrix is examined in the following code:

```
> A2AIPW <- optTx(fitSS_AIPW)$optimalTx
> table(A2,A2opt)
                    A2opt
A2                  0                   1
0                   217                 45
1                   167                 71

> table(A2AIPW,A2opt)
                    A2opt
A2AIPW              0                   1
0                   366                 15
1                   18                  101
```

The results show that the AIPWE significantly improve the accuracy of correctly assigning optimal regime to the patients. In this analysis, only 18 subjects are incorrectly assigned A=1 by the AIPW method, and 15 subjects are falsely assigned A=0. This number is better than the IPWE method.

```
> rpart.plot(classif(object = fitSS_AIPW))
```

The decision tree shows that x21 and x23 are used to split most nodes (*Figure 3*). Subjects with x21 <0.3 were correctly assigned A=0. The subjects with x23 ≥0.46 are also correctly assigned to receive A=0.

The first stage decision rule can be learned in a recursive manner as that for stage two. The only difference is that the predictors for both outcome model and propensity model are chosen from x11, x12 and x13. Furthermore, the response argument in the optimalClass() function receives the object returned in the stage two analysis.

Firstly, we define the propensity for treatment model and methods. Here the variable x11 and x12 are used to predict treatment assignment.

```
> moPropen <- buildModelObj(model = ~ x11+x12,
                    solver.method = 'glm',
                    solver.args = list('family'='binomial'),
                    predict.method = 'predict.glm',
                    predict.args = list(type='response'))
```

Secondly, we define the classification model in which the CART method is used. Other machine learning methods such as support vector machine are also allowed and can be passed to the buildModelObj() function by the solver.method argument.

```
> moClass <- buildModelObj(model = ~x11 +x12+x13,
                solver.method = 'rpart',
                solver.args = list(method="class",
                    control = rpart.control(minsplit = 50)),
                predict.args = list(type='class'))
```

Thirdly, we define the outcome model which is composed of the main effect model and contrast model.

```
> moMain <- buildModelObj(model = ~x11+x12+x13,
                    solver.method = 'lm')
> moCont <- buildModelObj(model = ~x11+x12+x13,
                    solver.method = 'lm')
```

Finally, the AIPWE model can be fit with the optimalClass() as before. Note that an optimalClass object

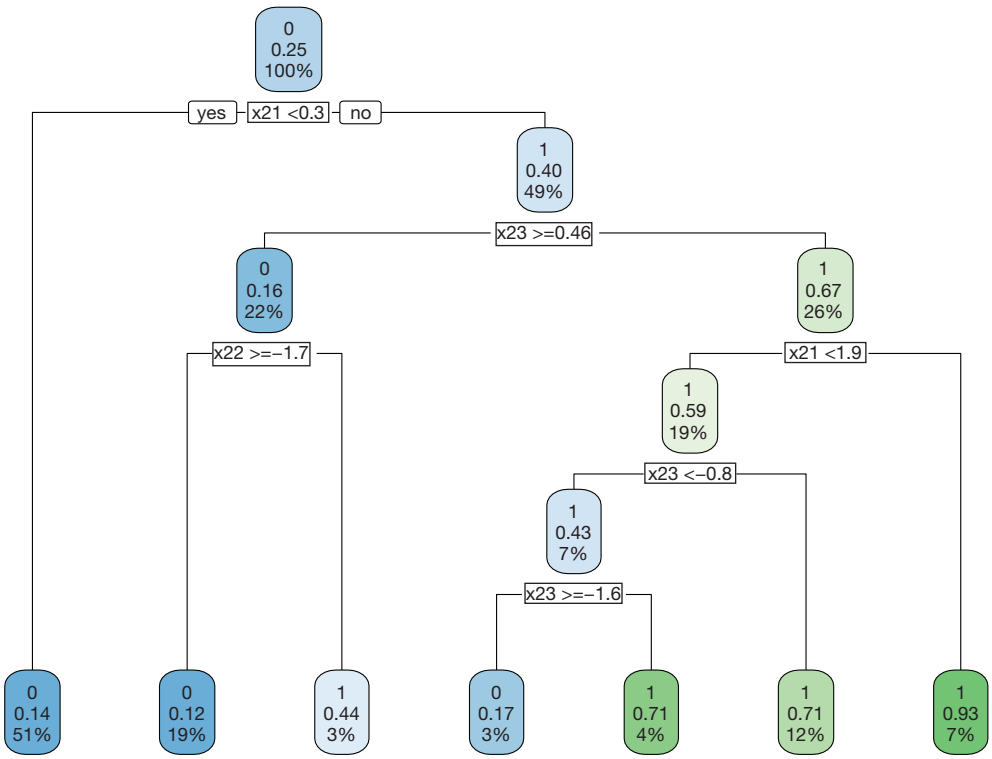

Figure 3 Decision tree to assign treatment regime A=1 or A=0 to subjects with covariates X2 at stage two by the AIPWE method. AIPWE, Augmented Inverse Probability Weighted Estimators.

is assigned to the response argument, which passes the counterfactual outcome under optimal treatment at stage two to the modeling strategy for the first stage. This is consistent with the idea behind reinforcement learning algorithm in computer science that the action-value function is updated by assuming all subsequent steps takes the optimal action to maximize the action-value function.

```
> fitFS_AIPW <- optimalClass(moPropen = moPropen,
                             moMain = moMain, moCont = moCont,
                             moClass = moClass,
                             data = dt, response = fitSS_AIPW,
                             txName = 'A1')
> rpart.plot(classif(object = fitFS_AIPW))
```

The data generating mechanism dictates that optimal regime A=1 should be given to subjects with x11 > −0.54 and x12 <0.54. The majority of subjects receiving A=1 following the AIPWE rule is those with x11 >−0.52, x12 <0.57 and x12 >−1 (*Figure 4*). Classification accuracy can be further examined with confusion matrix:

```
> A1AIPW <- optTx(fitFS_AIPW)$optimalTx
> table(A1, A1opt)
A1opt
A1          0          1
0         141        140
1          95        124

> table(A1AIPW, A1opt)
            A1opt
A1AIPW      0          1
0         217         30
1          19        234
```

The above output shows that the treatment options made by physician misclassify 95 subjects to the A=1 and 140 to the A=0 group. The AIPWE rule is able to significantly improve the classification accuracy. Only 19 subjects are misclassified to A=1 group and 30 subjects are misclassified

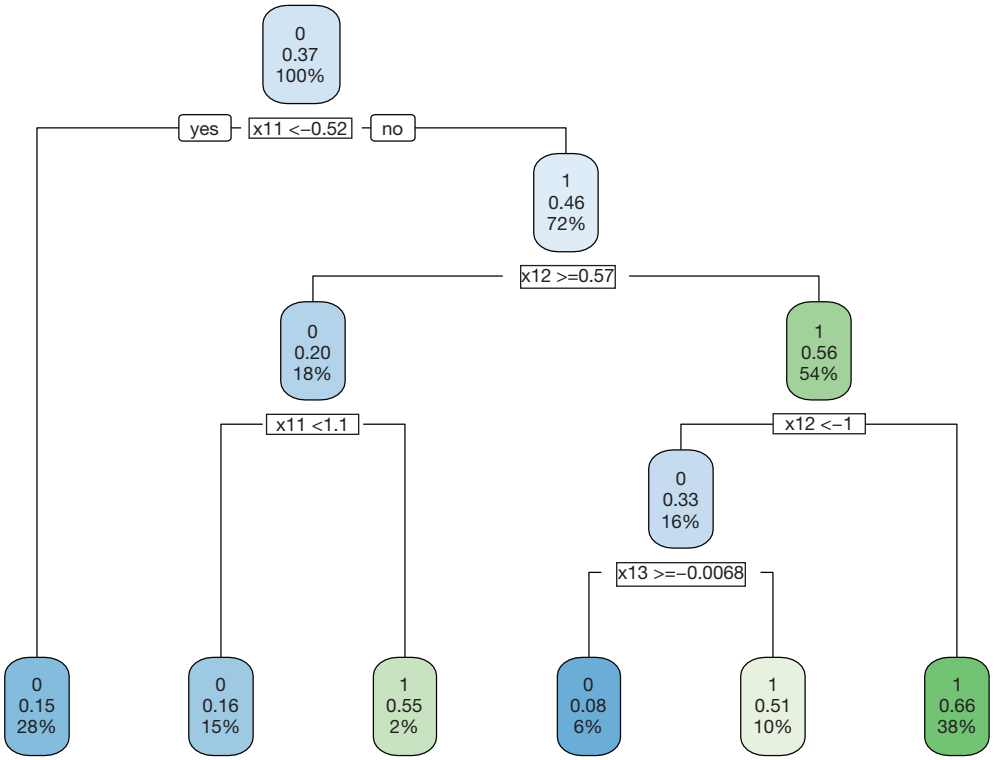

Figure 4 Decision tree to assign treatment regime A=1 or A=0 to subjects with covariates X1 at stage one by the AIPWE method. AIPWE, Augmented Inverse Probability Weighted Estimators.

to the A=0 group.

Final remarks

Exploring dynamic treatment regime borrows ideas from the reinforcement learning algorithm from the computer science. The Q-learning algorithm is easy to interpret for domain experts, but it is limited by the risk of misspecification of the linear outcome model. IPWE overcomes this problem by non-parametric modeling that the mean outcome is estimated by weighting the observed outcome. AIPWE borrows information from both the propensity model and mean outcome model and enjoys the property of double robustness. However, these statistical method does not allow the action space to have multiple levels. In real clinical practice, the interest may focus on the combinations of treatment regime, giving rise to a multi-dimensional action space.

Acknowledgments

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

Ethical Statement: The author is accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

References

1. Zhang Z, Zhang G, Goyal H, et al. Identification of subclasses of sepsis that showed different clinical outcomes and responses to amount of fluid resuscitation: a latent profile analysis. Crit Care 2018;22:347.
2. Raghu A, Komorowski M, Ahmed I, et al. Deep Reinforcement Learning for Sepsis Treatment. Vol. cs.AI, arXiv.org. 2017.
3. Komorowski M, Celi LA, Badawi O, et al. The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. Nat Med 2018;24:1716-20.
4. Statistical Methods for Dynamic Treatment Regimes

- Reinforcement Learning, Causal Inference, and Personalized Medicine | Bibhas Chakraborty | Springer. New York: Springer-Verlag, 2013 Jan 1. Available online: https://www.springer.com/us/book/9781461474272
5. Linn KA, Laber EB, Stefanski LA. iqLearn: Interactive Q-Learning in R. J Stat Softw 2015;64.
6. Wallace MP, Moodie EE, Stephens DA. Dynamic Treatment Regimen Estimation via Regression-Based Techniques: Introducing R Package DTRreg. Journal of Statistical Software 2017;80(2).
7. Wallace MP, Moodie EE. Doubly-robust dynamic treatment regimen estimation via weighted least squares. Biometrics 2015;71:636-44.
8. Barrett JK, Henderson R, Rosthøj S. Doubly Robust Estimation of Optimal Dynamic Treatment Regimes. Stat Biosci 2014;6:244-60.
9. Zhang B, Tsiatis AA, Laber EB, et al. A robust method for estimating optimal treatment regimes. Biometrics 2012;68:1010-8.
10. Loh WY. Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2011;1:14-23.
11. Zhang B, Tsiatis AA, Davidian M, et al. Estimating Optimal Treatment Regimes from a Classification Perspective. Stat 2012;1:103-14.

Cite this article as: Zhang Z; written on behalf of AME Big-Data Clinical Trial Collaborative Group. Reinforcement learning in clinical medicine: a method to optimize dynamic treatment regime over time. Ann Transl Med 2019;7(14):345. doi: 10.21037/atm.2019.06.75

