AUTOMATIC BLOOD PRESSURE CONTROL USING INTERPRETABLE REINFORCEMENT LEARNING MODEL

Tom-Avi Shapira and Danny Lange

Digital Medical Electronics – 048940, Spring 2022
Technion - Israel Institute of Technology
{tom-avi@campus,dlange@}.technion.ac.il

ABSTRACT

Blood pressure control is a common task in the medical world. This task is especially necessary during and after cardiac surgical interventions because some of the natural control systems of the body are disrupted. Intervention like that can cause the blood pressure to rise or fall dramatically, so in order to prevent a life-threatening situation, it needs to be monitored and controlled. As of today, this problem is solved by drugs treatment, given by a medical staff member that must monitor the patient's blood pressure often and give him an exact dose of a drug which is difficult to calculate. In this paper we first discuss a solution for automatic blood pressure control using reinforcement learning. This solution is considered as a "black box" that is hard to interpretate or verify. Thus, we suggest a PID controller solution that is tuned based on the reinforcement learning solution. Besides our goal of implementing automatic blood pressure control system, this paper demonstrates how a real problem in the medical world can be solved through reinforcement learning while receiving a generalized interpretable model that can be verified.

1. INTRODUCTION

Blood pressure (BP) is one of the vital signs as defined in the medical world [1]. Due to this fact, monitoring and controlling it is a common task. As of today, BP can be controlled by drugs treatment which is given by a medical staff member. For satisfactory care, BP monitoring must be often, and the treatment must be accurate. This is not simple due to the complexity of drugs' effect on BP over time. Abnormal values of BP can cause a life-threatening situation. Hence, when talking about cardiac surgical intervention, which disrupts the BP control systems of the body, monitoring and controlling it is not just a common task, but an essential one.

Today, monitoring BP continuously is not a solved problem and considered as a holy grail in the digital medical electronics world. In this paper we will focus on BP control under the assumption of using a classic BP monitor every fixed time. When talking about automatic BP control, a PID controller is a standard solution, but requires a sophisticated mathematical model [2]. Therefore, in the first part of this paper we propose a BP automatic control system, based on machine (deep) learning.

There are three main approaches of machine learning. Supervised learning, unsupervised learning, and Reinforcement Learning (RL). The first is commonly used when we have labeled data, the second is commonly used when our date is not labeled but has a unique pattern or statistics, and the last is commonly used when the goal is learning how to control a task properly with a pre-defined actions set. As mentioned before, our goal is controlling BP, thus we are using RL.

RL is concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. The agent basically explores the environment and learns how to take actions correctly in each situation. As part of the learning process, the agent balances between exploration and exploitation of what he has learned so far. Practically, the agent's learning process is done by tuning a Deep Neural Network (DNN).

Unfortunately, when talking about medical systems, deep learning is not a valid approach. A system based on deep learning is not interpretable and considered as a "black box" that is hard and sometimes even impossible to verify. Therefore, in the second part of this paper, based on PiRL [3], we show how to add constrains to our DNN solution. Eventually these constrains will create a generalized interpretable model that can be verified. In fact, we show how to tune a PID controller while learning from our agent (based on the DNN solution) and not according to a complex mathematical model. Finally, we discuss the advantages of this approach and present simulations and results.

parameter	value
C_0	$\frac{1 \cdot 10^6}{1498 \cdot 10^3} \left[\frac{ng}{ml}\right]$
F	1
K_a	$0.8 \left[\frac{1}{h}\right]$
K_e	$\frac{0.693}{50} = 0.014 \left[\frac{1}{h}\right]$

Table 1: The PK model parameters.

2. APPROCH

2.1. PKPD Model

In order to talk about BP control, we first present a model that explains both the way a drug spreads in the blood, and the drug's effect on the BP. In this section we discuss the model we chose to work with.

2.1.1. Pharmacokinetic (PK) Model

The PK model explains the way a drug spreads in the blood. There are 3 main ways to inject a drug into the body. Through the vein, through infusion and orally using a pill. We found the first way not too interesting to use, since the influence of the drug is immediate. On the other hand, we found the second way too complicated to calculate. Hence, we chose to use for our model an oral treatment using pills. Accordingly, the mathematical model is described in Equation 1, where C_0 represents the initial concentration of the given drug, normalized according to the volume of distribution of a person weighing 70 kg. F represents the bioavailability of the drug, K_a and K_e represent the absorption and elimination rates of the drug into and from the blood correspondingly, and t represents the time elapsed since the administration of the drug in hours. You can see more about the mathematical models and the parameters in the following link[2]. In our work, we used Amlodipine as a drug for lowering BP. We took the PK model parameters of amlodipine from [4] [5], while using Equation 2 from here[1]. We used half time ($t_{1/2}$) of amlodipine equals to 50 hours to calculate K_e. You can see the parameters' values in Table 1.

$$C(t) = \frac{C_0 F K_a}{K_a - K_e}\left(e^{-K_e t} - e^{-K_a t}\right) \tag{1}$$

$$K_e = \frac{0.693}{t_{1/2}} \tag{2}$$

[1]https://pharmacy.ufl.edu/files/2013/01/5127-28-equations.pdf

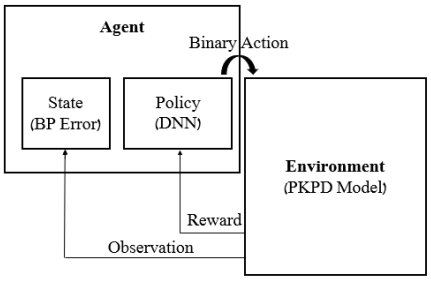

Figure 1: The reinforcement learning model.

2.1.2. Pharmacodynamic (PD) Model

The PD model explains how a drug affects the BP. We took the mathematical model, as described in Equations 3 and 4, from [5]. These equations describe the systolic BP (SBP) and the diastolic BP (DBP), depending on the concentration of the drug in the blood at any time. Notice that C(t) units should be [ng/ml] and the BP units are [mmHg] as usual. Finally, we calculate the mean BP using Equation 5.

$$SBP(t) = SBP_{initial}\left(1 - 0.164\left(\frac{C(t)}{C(t) + 8.27}\right)\right) \tag{3}$$

$$DBP(t) = DBP_{initial}\left(1 - 0.164\left(\frac{C(t)}{C(t) + 2.97}\right)\right) \tag{4}$$

$$BP(t) = DBP(t) + \frac{SBP(t) - DBP(t)}{3} \tag{5}$$

2.2. RL Model

In this section we present our RL model. A schematic can be seen in Figure 1. If you don't familiar with concepts in RL, you can read the following reference [6].

2.2.1. Environment

We built our environment based on the PKPD model we presented in section 2.1. At each time, the environment takes as input a binary action that represents whether a drug was administered or not. Then, according to the equations in section 2.1.1, using superposition, the amount of the drug in the blood is calculated. Following that, using the equations in section 2.1.2, the blood pressure is calculated. Finally, a reward and an observation, which is the BP error relative to a predefined referenced BP, as represented in Equation 6, are calculated and returned from the environment. While taking an action that

$$
\begin{aligned}
&if\ abs(BP_{Error}) \leq 3: \\
&\qquad reward = 100 \\
&elif\ abs(BP_{Error}) < abs(BP_{Pre\_Error}): \\
&\qquad reward = 50 \\
&elif\ BP_{Error} \cdot BP_{Pre\_Error} < 0: \\
&\qquad reward = 20 \\
&else: \\
&\qquad reward = -20
\end{aligned}
$$

Figure 2: The reward function of the environment.

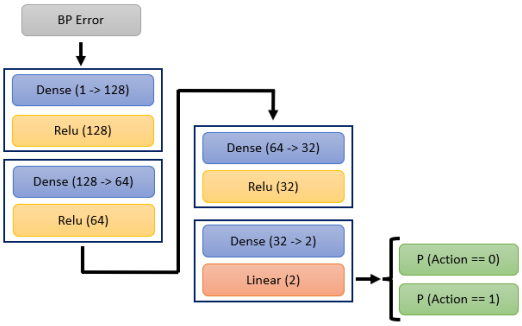

Figure 3: The network architecture that determines the agent policy.

decreases the BP error, the reward is positive. Otherwise, the reward is negative.

$$BP_{Error}(t) = BP_{reference} - BP(t) \tag{6}$$

Our reward function is presented in Figure 2. Notice that we also save the previous BP error (BP_{Pre_Error}) to detect an improvement in the error. We also considered the case of a change in the BP error sign as a good case. Such a large change between two BP measurements can only be accepted when a new drug is given. Therefore, this branch encouraging lowering BP.

2.2.2. *Agent*

We implemented our agent using DQN Agent[2]. We used a SequentialMemory with a limit of 10^6 observations (BP errors), a learning rate equals to 0.0001 and EpsGreedyQPolicy (for the learning process) with a constant epsilon equals to 0.2. we trained our agent for 50,000 steps. therefore, it was important to pick a small epsilon which could allow an exploitation and effective learning in a short time.
Our network architecture that determines the agent policy is based on the following solution[3] and represented in Figure 3. The Network takes as input an observation (BP

[2]https://keras-rl.readthedocs.io/en/latest/agents/dqn/
[3]https://towardsdatascience.com/teaching-a-computer-to-land-on-the-moon-c168d551fc68

$$
\begin{aligned}
&if\ c_1 \cdot P + c_2 \cdot I + c_3 \cdot D > c_4 \\
&\qquad 0 \\
&else: \\
&\qquad 1
\end{aligned}
$$

Figure 4: The PID controller sketch.

error) which defines the environment state in a specific time, then calculates the probability of the necessity of giving the patient a drug. P(action == 0) represents the probability of not giving a drug and P(action == 1) represents the probability of giving a drug. The agent will choose its action according to the greater probability.

2.2.3. *Training*

We trained our agent to take an action every 1 hour, for 50,000 steps. We defined a training episode as 8 days which are 192 hours. i.e., we trained the agent for 260 episodes. In every episode we used initial SBP and DBP which were randomly selected from [140/90 - 5, 140/90 + 5] [mmHg] respectively. The predefined referenced BP was set to a constant value of 100 [mmHg] in each episode.

2.3. PID Controller Model

In this section we present a PID controller model based on the RL agent we trained as described in section 2.2.

2.3.1. *Background*

A Proportional–Integral–Derivative (PID) controller is a control loop mechanism employing feedback. The controller continuously calculates an error value from a reference value. Based on the error value the controller decides how to act at each moment. The error value is calculated by weighting the three components P, I and D. The proportional part is responsible for considering the current error value. The Integral part is responsible for considering the cumulative error over time. the derivative part is responsible for considering the change in the error. We define the derivative to be as shown in Equation 7. Tuning the weights of the controller is not a simple task and requires a complex mathematical model. In addition, if the problem cannot be represented well using a mathematical model, it could limit the tuning. Therefore, and according to the "black box" problem that was described in the introduction, in the next section we will suggest a tuning based on RL non-limited model.

$$\mathrm{D} = BP_{Pre\_Error} - BP_{Error} \tag{7}$$

2.3.2. *Tuning Using PiRL*

We create an interpretable RL model using PiRL [3]

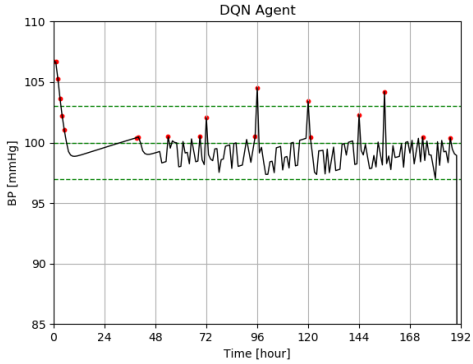

Figure 5: DQN agent simulation.

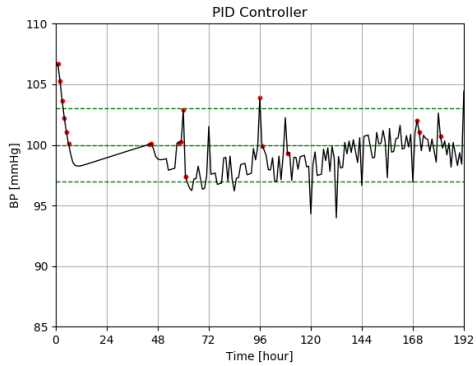

Figure 6: PID controller simulation.

algorithm. PiRL goal is automatically discover expressive policies in a high-level programmatic language. This goal is achieved by synthesize a programmatic policy with syntactic constrains, a sketch, that imitates neural policy. The bottom line of this process is to take a large complex DNN which is non-limited, put constraints on it and produce a model with a small number of parameters that imitates the DNN. In that way we can get an interpretable model which can be verified. Note that it is important to choose for the model a sketch that fits the problem, otherwise the model couldn't imitate the DNN properly. Our problem is automatic BP control and based on [2], a PID controller is a standard solution for it. In PiRL another problem whose standard solution is a PID controller was investigated, and the optimal sketch found was a PID controller. Based on it, we define our sketch to be a PID controller, without implementing the "neighborhood_pool" function as in PiRL. The sketch is represented in Figure 4. In practice, eventually we tune a PID controller weights using a light version of PiRL, based on the following algorithm[4]. After tuning, our sketch weights, c1 to c4, were set to 1.39, 0.01, 0.78, 0.08 respectively. For a sanity check, notice that all of the PID controller parameters are positive. Therefore, a positive BP error (the current BP is below the reference as defined in Equation 6) meaning we shouldn't give the patient a drug, as indeed happens according to the sketch.

3. EXPERIMENT AND RESULTS

3.1. DQN Agent Performance

We present the DQN agent performance in Figure 5. We ran a simulation of the agent and measured the BP every one hour. The initial BP was set to 107 [mmHg] and the reference BP was set to 100 [mmHg]. A dashed green line can be seen at the reference value and plus minus 3 [mmHg] from it. Every red dot represents a drug giving to the patient and starting from 48 hours we apply a random noise to the BP value. Generally, it can be seen that the agent succeeds controlling the BP. Moreover, before adding the noise it can be seen that the agent learns the drug effect on the BP, so it waits for the BP to decrease without giving more doses. However, when adding the noise, the agent fails to filter it. It chooses to give a drug every time the BP is high and causes an excessive concentration of a drug in the blood. One can see that all of the peaks are red and that the average BP drops slightly below the reference. As shown in section 3.3, this phenomenon is even worse when the noise increases.

3.2. PID Controller Performance

We present the PID controller performance in Figure 6. We ran the same simulation as described in section 3.1. Generally, it can be seen that the controller succeeds controlling the BP. While comparing to the DQN agent, it can be seen that before adding the noise, the average BP is less accurate. This is caused due to the lack of BP error samples, which is not sufficient for the integral part of the PID to sum on. Indeed, as can be seen in section 3.3 this error decreases over time. Besides that, note that without adding noise, the PID controller can't be better than the DQN agent, since the first one is tuned based on the second. In addition, it can be seen that while adding the noise, the controller succeeds in filtering it and is not tempted to give a drug every time the BP is high (in particular at the peaks). For example, the controller doesn't give a drug while the average BP is too low (when the time is between 110 to 168 hours). This ability to filter out the noise is actually the controller's ability to generalize the DQN agent's solution which has done overfit. In fact, we prevent the overfit while using the PID controller sketch that has limited the number of parameters of the DNN.

[4] https://github.com/VAIBHAV-2303/PiRL

	Without Noise		Random Noise ϵ [-1, 1]		Random Noise ϵ [-5, 5]	
	DQN Agent	**PID Controller**	**DQN Agent**	**PID Controller**	**DQN Agent**	**PID Controller**
Average |Error| (7 days) [mmHg]	0.5	0.7	1.04 (83%)	0.98 (40%)	3.21 (456%)	2.39 (241%)
Average |Error| after warmup (7 days) [mmHg]	0.4	0.42	0.91 (128%)	0.45 (7%)	3.59 (798%)	0.66 (57%)
Max |Error| (7 days) [mmHg]	1.13	1.74	1.67 (48%)	2.27 (30%)	3.72 (229%)	3.52 (102%)

Table 2: Quantitative blood pressure errors caused by the DQN agent and the PID controller in different noise conditions. The percentages indicate the increase of the error in the presence of noise compared to the condition without noise.

3.3. Quantitative Blood Pressure Error Experiment

We measured the BP errors caused by the DQN agent and the PID controller in different conditions. The results are shown in Table 2. The percentages in the table indicate the increase of the error in the presence of noise, compared to the condition without noise. We ran two types of simulations. The first simulation examines the average BP error during the first 7 days from the moment of administration of the drug. In addition, we noted the maximum BP error we measured during these 7 days. The second simulation examines the average BP error during 7 days after a warmup of 7 days. This simulation has been performed to evaluate the PID controller performance in a steady state after acquiring sufficient history samples. We ran each simulation in 3 noise conditions, as described in the table. While talking about the simulations without noise, since the controller tuned based on the agent, the DQN agent performance are always better. Notice that after a warmup the gap in the errors decreases. When adding the noise, it is clear that the PID controller performances are better and even improved after the warmup. Likewise, it can be seen that the more the noise increases, the more the agent's performances are significantly affected and decreases. Moreover, based on the percentages, it can be seen that generally the DQN agents suffers much from noise comparing to the PID controller.

3.4. PID Controller Interpretability And Verifiability

In this section we will explain how our PID controller sketch succeeds controlling the BP and how the controller's weights we received can be interpreted and verified. Once you understand how to interpret the model, you can test it on different BP values and perform verification in a simple and clear way. As explained in section 2.3.1, the PID controller has 3 parts, each of which is responsible for considering a different factor of the error.

3.4.1. The Proportional Part (P)

This part is responsible for considering the current error value. Mainly it is affecting the rise time of the controller and its immediate response. Therefore, this part should have the significant weight of the controller, as indeed happened. When talking about the sign of the proportional part, according to the way we defined the error in Equation 6, the weight must be positive, as indeed happened. Otherwise, due to the sketch, a positive error value (that indicates a low BP) will lead to an administration of additional drug. This situation is of course not desirable.

3.4.2. The Integral Part (I)

This part is responsible for decreasing the cumulative error over time and as a result responsible for eliminating the steady state error. In fact, this part must not be zero, otherwise the steady state error will never disappear. When talking about the sign of the integral part, since a consistent positive error indicates a low blood pressure over time and due to the sketch, the situation is similar to the proportional part. Therefore, the weight must be positive, as indeed happened.

3.4.3. The Derivative Part (D)

This part is responsible for considering the change in the error and as a result decreases the setting time of the controller. Meaning, the greater the weight of this part, the faster the controller reaches a stable value. Hence, this part is the reason why the controller manages to handle noise well. When looking at how this part is defined in Equation 7, and the positive value that was set to it after tuning the controller, this property can be interpreted directly and clearly. Without limitation of generality, let's consider a situation when the previous error is positive, and the current error is negative, meaning the

derivative of the error is a high positive value. According to Equation 6, this situation means that in the previous step the BP was too low and in the current step the BP is too high. Such a big change in the BP's value can only occur due to noise in the measurements. As mentioned, in that case the derivative part is positive (as the weight that was set after tuning). Hence, according to the sketch, the derivative part will encourage the controller not to be tempted and give a drug despite the high BP was measured.

4. CONCLUSION

In this paper we suggested a solution for automatic blood pressure control using interpretable reinforcement learning model. Moreover, we demonstrated how a real problem in the medical world can be solved through reinforcement learning while receiving a generalized interpretable model that can be verified. First, we implemented an agent that learns to control blood pressure in an environment which simulates a PKPD model, based on [4] [5]. When talking about medical systems, deep learning is not a valid approach. A system based on deep learning is not interpretable and consider as a "black box" that is hard and sometimes even impossible to verify. Therefore, using PiRL [3], we have implemented a PID controller that was tuned based on what the agent had learned. In that way we didn't limit the tuning process to a complex mathematical model, which may not be able to represent the problem well. We have shown that the agent was indeed able to control the blood pressure, but due to an overfit effect, when adding noise (and changing the environment) it failed. Unlike the agent, the PID controller successfully generalized the solution and managed to control the blood pressure, even in presence of noise. Finally, we explained how the PID controller solution can be interpreted and verified.

5. REFERENCES

[1] Mok WQ, Wang W, Liaw SY. International Journal of Nursing Practice 2015; 21 (Suppl. 2): 91–98. Vital signs monitoring to detect patient deterioration: An integrative literature review

[2] Sandu, C. and Popescu, D., 2016. Reinforcement learning for the control of blood pressure in post cardiac surgery patients. *UPB Sci. Bull., Series C*, *78*(1), pp.139-150.

[3] Verma, A., Murali, V., Singh, R., Kohli, P. and Chaudhuri, S., 2018, July. Programmatically interpretable reinforcement learning. In International Conference on Machine Learning (pp. 5045-5054). PMLR.

[4] Mukherjee, D., Zha, J., Menon, R.M. and Shebley, M., 2018. Guiding dose adjustment of amlodipine after co-administration with ritonavir containing regimens using a physiologically-based pharmacokinetic/pharmacodynamic model. *Journal of pharmacokinetics and pharmacodynamics*, *45*(3), pp.443-456.

[5] Heo, Y.A., Holford, N., Kim, Y., Son, M. and Park, K., 2016. Quantitative model for the blood pressure-lowering interaction of valsartan and amlodipine. *British journal of clinical pharmacology*, *82*(6), pp.1557-1567.

[6] Li, Y., 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.