AA228 - Dynamic Glucose Monitoring with Partially Observable Markov Decision Processes

Isha Thapa
Department of Management Science & Engineering
Stanford University
ishadt@stanford.edu

Isabelle Rao
Department of Management Science & Engineering
Stanford University
isarao@stanford.edu

William Cai
Department of Management Science & Engineering
Stanford University
willcai@stanford.edu

December 6, 2019

Abstract

Continuous glucose monitoring (CGM) has been used in the past to help with timely glucose management for patients suffering with diabetes. Traditional CGM takes measurements at fixed intervals, using the measurements to determine whether the patient has a low or high glucose level, triggering an alert. However, this paradigm of fixed-interval measurements may be wasteful in that measurements which deplete the battery life of the monitoring device are taken even when the patient has little probability of being outside the thresholds for a normal blood glucose level. This paper uses a partially observable Markov decision process to optimize when to measure the glucose level and send an alert to the patient. We use an open glucose monitoring dataset from the Jaeb Center for Health Research. We find that using a POMDP increases the efficiency of the algorithm for our defined reward function, which encodes a small penalty for taking a measurement and a large penalty for failing to alert the patient when their glucose level is abnormally high or low. Throughout this paper we make the strong assumption that alerting the patient has no causal effect on their blood glucose level: such an assumption is necessary because in our data patients are always alerted at fixed thresholds.

***K*eywords** Continuous Glucose Monitoring · Partially observable Markov decision process · POMDP · Particle filter · Fast Informed Bound · One-step look-ahead · Optimal policy · Alpha-vectors

1 Introduction

Continuous glucose monitoring (CGM) has been used in the past to help with timely glucose management for patients suffering with diabetes. For instance, Miller et al. (2019) [1] proposed a standardized approach based on individual CGM data to set patient-specific disease-management targets and monitor progress.

Since the monitors track the glucose level continuously, their batteries are quite large, which makes them inconvenient to wear for patients. Moreover, continuously monitoring patients might not be

necessary. Samples could be taken at dynamically chosen intervals of time to check the glucose level. However, this raises the question of how often to sample. For example, if the levels seem stable or far from levels which necessitate an alert, it might not be necessary to sample more frequently. But if glucose levels start to increase, decrease, or move near an alert threshold, it might be necessary to sample more frequently.

2 Data

We obtained CGM data from the Jaeb Center for Health Research, which hosts open datasets. We use data from 'A Randomized Trial Comparing Continuous Glucose Monitoring With and Without Routine Blood Glucose Monitoring in Adults with Type 1 Diabetes', where half the patients receive continuous glucose monitoring. The data consists of 226 patients with an average of 60715 glucose measurements for each patient. Most measurements are taken in increments in 5 minutes, but occasionally there are measurements taken at irregular intervals or long gaps between measurements. In order to simplify the analysis, we restrict ourselves to considering sequences of at least 100 measurements in a row, each spaced 300 seconds apart, filtering out all measurements which do not fit in such a sequence.

3 Methods

We propose modeling the problem as a POMDP. Our actions are whether to sample a data point, and whether to send an alert to the user. Our state space is continuous, and consists of the three previous glucose measurements. The observation state is the glucose value if the policy chooses to sample, and 0 otherwise. Importantly, we make the strong assumption that our action does not affect which state we transition to:

$$P(s'|s,a) = P(s'|s)$$

This assumption is necessary because from our data we cannot infer what will happen if a patient is not alerted, because glucose levels were continually monitored and patients always received an alert when they moved outside a threshold. This assumption may not be realistic, because a patient might reasonably inject insulin given an alert that their blood sugar is too high or eat a carbohydrate-high meal if their sugar is too low.

3.1 Update Belief

We assume that we always sample the first three glucose measurements.

If we choose to sample, then we simply update the state using the measured value. For example, if at time t the state is: $s_t = [g_2^t, g_1^t, g_0^t]$, then at $t+1$ the state is: $s_{t+1} = [g_2^{t+1} = g_1^t, g_1^{t+1} = g_0^t, g_0^{t+1}]$ where g_0^{t+1} is the newly measured glucose level.

If we do not sample, then we use a particle filter to update our belief since the state space is continuous. For our generative model, we use a normal distribution with mean predicted by a linear regression. Because of the temporal aspect of our data, we decided to use an auto-regressive (AR) model trained on the entire glucose data to predict the mean.

$$\mu_{t+1} = \alpha + \beta_0 g_0^t + \beta_1 g_1^t + \beta_2 g_2^t$$

The standard deviation σ is constant, and is calculated as follow:

$$\sigma = \frac{1}{\sqrt{(n-2)}} \sum_i \epsilon_i^2$$

where $\{\epsilon_i\}_{i=1..n}$ are the residuals, and n is the number of observations in the dataset.

Therefore, the next glucose level if we do not sample is a random variable, with mean predicted from the AR model and a constant standard deviation: $g_0^{t+1} \sim \mathcal{N}(\mu_{t+1}, \sigma)$. We sample 5 points.

3.2 Reward Function

Our reward function will take into account the energy required to sample and send the data points, but also the negative consequences that could occur to the patients if we fail to identify irregular levels and alert the patient. Let a be the decision whether to alert the patient, and s the decision whether to sample a data point. We use the following reward function: $R(a, s, \text{state}) = R_1(a, \text{state}) + R_2(s)$ where $R(s) = -1$ if $s = sample$, and 0 otherwise.

$R_1(a, \text{state})$	Low glucose	Normal level	High glucose
Alert	0	-2	0
Do not alert	-10	0	-3

We consider that a glucose level is low if it is below 70, that it is normal if it is between 70 and 180, and that is high if it is above 180.

If we do not sample, then our belief of the next glucose level is a random variable normally distributed, with mean μ predicted by the AR model, and standard deviation σ calculated as in the previous section.

Therefore, denoting by X the glucose value, we have $X \sim \mathcal{N}(\mu, \sigma)$. Our expected reward if we alert is:

$$r_{\text{alert}} = -10\mathbb{P}(X \leq 70) - 3\mathbb{P}(X \geq 180)$$

And if we do not alert, the reward is:

$$r_{\text{no alert}} = -2\mathbb{P}(70 \leq X \leq 180)$$

We choose to alert if the reward to alert is higher, so finally our expected reward if we do not sample is:

$$r_{\text{no sample}} = \max(r_{\text{alert}}, r_{\text{no alert}})$$

If we choose to sample, then the reward is simply -1 because we can always alert correctly. Thus, we have: $r_{\text{sample}} = -1$

3.3 Fast Informed Bound

We use Fast Informed Bound (FIB) to compute the alpha-vectors iteratively:

$$\alpha_a^{k+1}(s) = R(s, a) + \gamma \sum_o \max_{a'} \sum_{s'} O(o|s', a) T(s'|s, a) \alpha_{a'}^k(s')$$

We used a particle filter to update our beliefs instead of a transition model, so here again we generate samples from the normal distribution with mean predicted from the AR model instead of summing over all possible states s'.

Since our state space is continuous, we use k-dimensional trees to estimate the alpha-vectors. This method organizes a set of k-dimensional points, i.e. the seen states, and allows us to identify the nearest neighbors of unseen states in order to interpolate the alpha-values. More specifically, we find the two closest states, determine their distance to the unseen state, and estimate the alpha-value

as a weighted mean of the values of the nearest neighbors, where the weights are the inverse of the distances.

Using k-d trees is advantageous because storing the tree is not very memory intensive, and searching for the nearest neighbors of a state is quite fast (in the order of $\log(n)$, where n is the number of seen states).

3.4 One-step lookahead

We use a one-step lookahead approximation to improve on the policy that has been computed through the Fast Informed Bound:

$$\pi(b) = \arg\max_a [R(b,a) + \gamma \frac{1}{n} \sum_{i=1}^{n} U(\text{UpdateBelief}(b, a, o_{a,i}))$$

Instead of summing over all observations, we use a sample of n generated observations based on the belief state and action. We generate only one observation per particle in our belief state due to computational limitations. If the action is to not sample, the observation is None. If the action is to sample, then the observation is sampled from the normal distribution with a mean of the predicted value, given the particle, from the AR model.

We use these observations to update our belief and then find the utility estimation for each particle in the new belief state by computing the corresponding alpha value estimation. We then set our utility for each belief to be the minimum utility of the particles in that given belief state.

4 Results

We compare the performance of our policy against two naive policies in table 1: Never-Measure and Always-Measure. In Always-Measure, we incur the cost of 1 for measuring blood glucose at every timestep, but in return we are always correct in whether we alert because we know the true value, so we have a fixed reward of -1 for each timestep. In Never-Measure, we never pay the cost for measuring blood glucose, but our only idea of what state we are in is wherever our particles happened to diffuse to. Finally, in our policy we use one step lookahead with our alpha generated from the fast informed bound to decide what action to take. We find that our policy outperforms Always-Sample, the better of the two naive policies, by around 10%.

Always Sample	Never Sample	Our Policy
1.000	1.141	0.893

Table 1: Comparison of the average cost per step for our policy with one which always samples glucose levels, and another which never samples glucose levels, where policies use 10 particles to represent a belief.

We also investigate how the number of particles affects the performance of the three policies in figure 1. We find that the Always-Sample policy performs equally well, since nothing depends on the belief in that policy, but the other two policies benefit from having a better representation of belief. Our results are stochastic so the plot is not exactly monotone, but there seems to be a decreasing trend, especially for the POMDP. Computational constraints prevent us from re-running the pomdp multiple times to get better estimates or running with more particles.

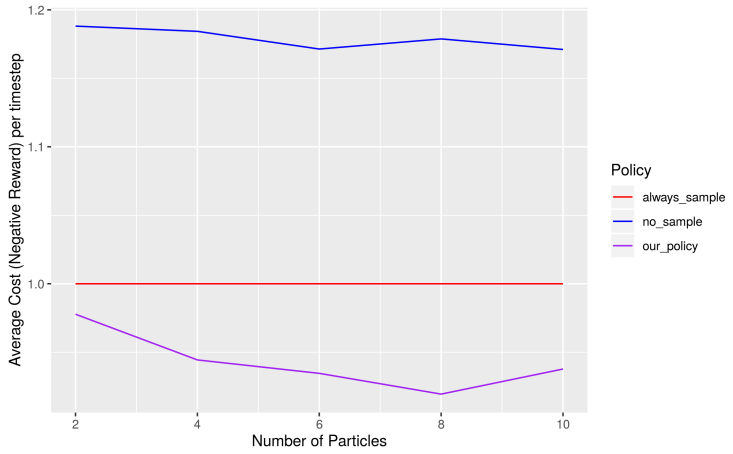

Figure 1: Cost (negative reward) per timestep by the number of particles used to capture belief for our policy along with the two naive policies.

References

[1] Daniel Miller, Andrew Ward, David Maahs, and David Scheinker. 960-p: Personalized diabetes management using data from continuous glucose monitors. *Diabetes*, 68(Supplement 1), 2019.