University of Padova

Department of Mathematics

Master Thesis in Data Science - Machine Learning for Intelligent Systems

Reinforcement Learning approaches for Artificial Pancreas control

Supervisor
Gian Antonio Susto
University of Padova

Co-supervisors
Simone Del Favero
Mirco Rampazzo
Francesco Zanini
Eleonora Manzoni
University of Padova

Master Candidate
Alvise Dei Rossi

Academic Year
2021-2022

To my family and friends,
For your love, support and patience.

Abstract

People with type 1 diabetes are affected by a chronic deficiency of insulin secretion in their body; as a consequence, insulin has to be continually self-administered to keep in check their blood glucose levels. In recent years, rapid technological advancements in continuous glucose monitoring and insulin administration systems have allowed researchers to work on automated control methods for diabetes management, commonly referred to as Artificial Pancreas. The development of control algorithms in this context is a very active research area. While traditional control approaches have been the main focus so far, Reinforcement Learning (RL) seems to offer a compelling alternative framework, which has not been thoroughly explored yet.

This thesis investigates the employment of several RL approaches, based on the algorithm Sarsa(λ), on in silico patients, using the FDA accepted UVa-Padova Type 1 Diabetes simulator. The way the overall representation of the problem affects the performance of the system is discussed, underlying how each component fits into the general framework proposed and evaluating the pros and cons of each method. Particular emphasis is also placed on the interpretability of both the training process and the final policies obtained. Experimental results demonstrate that classic RL methods have the potential to be a viable future approach to achieve proper control and a good degree of personalization in glycemic regulation for diabetes management.

Contents

Listing of figures

Listing of tables

Listing of acronyms

BG . . . . . Blood glucose

T1D . . . . . Type 1 Diabetes

T2D . . . . . Type 2 Diabetes

AP . . . . . Artificial Pancreas

RL . . . . . Reinforcement Learning

ML . . . . . Machine Learning

NN . . . . . Neural Network

CGM . . . . . Continuous glucose monitoring

SMBG . . . . . Self monitoring blood glucose

CSII . . . . . Continuous subcutaneous insulin infusion

CHO . . . . . Carbohydrates

BR . . . . . Basal rate

IOB . . . . . Insulin on board

COB . . . . . Carbohydrates on board

TIR . . . . . Time in range

HBGI . . . . . High Blood Glucose Index

LBGI . . . . . Low Blood Glucose Index

PID . . . . . Proportional integral derivative

MPC . . . . . Model Predictive Control

MDP . . . . . Markov Decision Process

1 Introduction

1.1 Diabetes, a global pandemic

Diabetes Mellitus refers to a range of metabolic disorders which are characterized by chronic elevated blood glucose (BG) levels. This happens mainly as a consequence of the deficiency of natural secretion of insulin by the pancreas (Type 1 Diabetes, T1D), its ineffective use in the body (Type 2 Diabetes, T2D) or as a complication during pregnancy (Gestational Diabetes), when the extra needs of insulin are not met. Diabetes symptoms include increased thirst, frequent urination, fatigue, irritability, blurred vision, and frequent infections, depending on blood sugar level. In the long term, especially if left unchecked or undiagnosed, diabetes can lead to severe complications such as retinopathy, cardiovascular diseases, neuropathy, nephropathy and lower limbs amputation.

Nowadays, over 530 million people in the world are affected by diabetes and the number is rapidly rising [1, 2]. Overall, T1D accounts for approximately 5% of all cases and affects about 20 million individuals worldwide, while most of the others are related to T2D. According to the World Health Organization (WHO), about 1.5 to 5 million deaths are directly linked to diabetes every year, making it the ninth leading cause of death in 2019 [3] and one of the largest components of global health expenditure (about 966 billion USD dollars in 2021 [1]). Figures such as these clearly exemplify why research regarding the treatment of all types of diabetes is to be considered a priority by the scientific community.

1.2 Manual control treatment for T1D

T1D is associated with decreased life expectancy and increased morbidity, but strong evidence indicates that good metabolic control decreases diabetes complications [4, 5]. The standard insulin replacement therapy for T1D consists in delivering boluses of fast-acting insulin right after the meals to compensate for the rapid blood glucose increase, and in delivering 1-2 daily injections of slow-acting insulin to keep glucose levels within a specified target range (usually 70-180 $\mathrm{mg\,dL^{-1}}$), while the patient is in fasting condition. Correction boluses also have to be administered in case hyperglycemia (i.e. the deviation to high levels of BG) is detected. In order to do so, the patient is supposed to autonomously check periodically the blood glucose concentration through capillary measurements (self-monitoring BG, SMBG) and calculate the amount of insulin to deliver based on the amount of carbohydrates (CHO) (s)he is assuming, the current level of blood glucose, the previous amounts of insulin injected, and some patient-specific parameters (such as carbohydrate-to-insulin ratio, CR, or correction factor, CF), which are usually estimated by a physician based on observations over time and gradually adjusted by trial and error. If hypoglycemia (i.e. the condition in which BG level is lower than the standard range) is detected, instead, the patient has to react immediately by eating carbohydrates in order to bring it back to a safe level and subsequently assess the return to normality by SMBG check. This approach tends to be error-prone. In fact, even with a due amount of vigilance, many patients may still suffer significant diabetes-associated complications. Another aspect to consider is that this approach, being so reliant on the constant attention of the patient, also worsens the quality of life, considering the number of actions that have to be constantly performed for diabetes management.

1.3 Artificial Pancreas, automated control

Numerous technological advancements have been made in recent years to improve therapies for T1D. The general trend in improving the system described in the previous section has been to gradually automate all steps of the process. Continuous glucose monitoring (CGM) sensors have now enabled to collect in real time measurements of BG levels, usually at intervals of 1-5 minutes [6]. To automate insulin delivery, insulin pumps (Continuous subcutaneous insulin infusion, CSII) are now available, allowing the diabetic patient to regulate the dosage required throughout the day [7]. Both of these two medical wearable devices have seen recent improvements in terms both of accuracy and reliability. This has allowed researchers to consider sys-

tems denoted as Artificial Pancreas (AP), which provide BG regulation for T1D patients using a smart control algorithm that, based on CGM data, modulates the insulin infusion rate delivered by the pump [8].

The framework described is highly generic: not only can different systems be envisioned based on the type of algorithm employed, but we can also differentiate them based on the amount of data and features exploited, and, critically, on the required degree of interaction with the patient. Currently, most controllers still require regular patient or caregiver intervention, operating in open-loop control with the user. Ideally, instead, a fully closed-loop insulin delivery system [9] would be able to determine autonomously, based only on CGM data history and previous injections, the proper insulin dosage to be applied, without any knowledge of external data, with the exception, possibly, of patient-specific parameters which are easily available, such as the age of the user, the gender, the weight and the basal rate. The algorithm should be able to do so in a personalized way, taking into consideration past observations and possibly some imposed priors for safety reasons.

Fully closed-loop control systems implicitly include the benefit of alleviating the patients' need to plan each day based on their illness, thereby improving quality of life. Since they rely solely on sensor data, the system is typically reactive and no preemptive exceptional dosing is ever planned. As a consequence of this, fully closed-loop models should usually be expected to be characterized by slightly slower reactions to variations in blood glucose level when compared to, for example, hybrid systems where the user is providing an estimate of the amount of CHO assumed right after a meal, or simply an announcement of the occurrence of the meal[10]. Essentially, hybrid systems trade-off some degree of autonomy, hence being more burdensome for the patients, in exchange for improved performance. It should also be noted that blood glucose concentration in diabetes is influenced by several disturbances not directly related to food intake: stress, physical activity, and infections are all factors that affect it[11, 12]. In the long term, to account for this, the AP hybrid closed-loop framework could easily be extended to make use of data obtained through wearable devices and apps for health services, which would allow, through automatic data collection, user-annotated data, and communications about the patient's physiological status, to attain a system which is reactive to external relevant factors too. Information could also flow in the other direction, from the control algorithm to the user, with the control algorithm delivering warnings, alerts, or possibly asking for confirmations, especially early on, when the infusion policy in place is still being refined. Figure 1.1 shows a diagram of the possible systems described so far.

Another core decision when designing an AP system is obviously the choice of the control

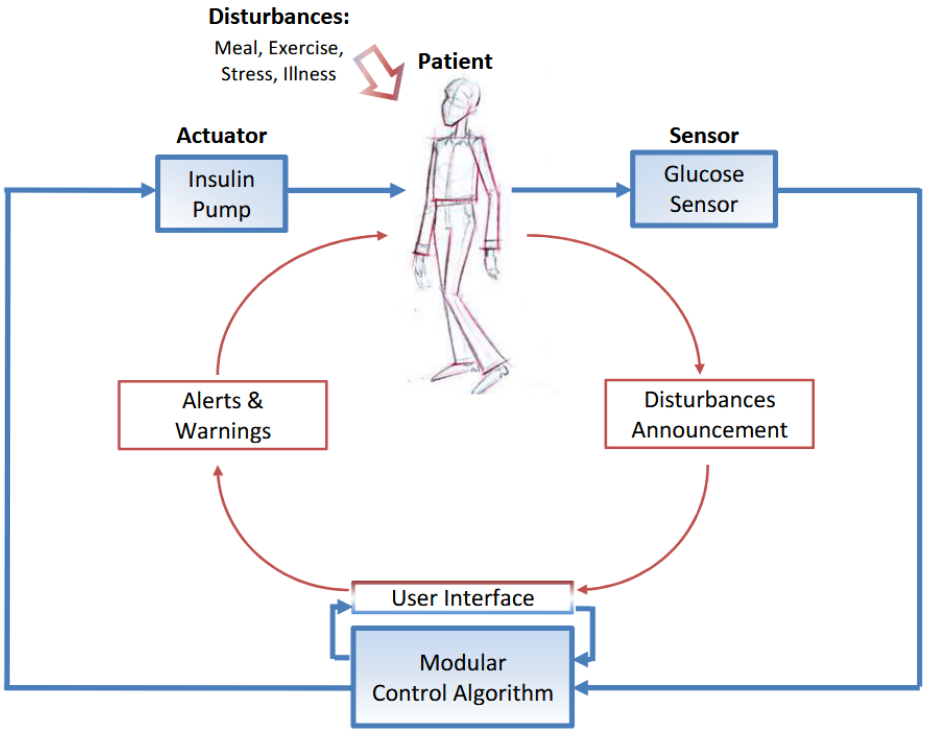

Figure 1.1: A diagram of possible AP systems. Fully closed-loop methods don't use connections denoted in red, related to direct communications between controller and patient, that influence the actions of the control algorithm. Hybrid methods allow for all types of flow of information, at the cost of autonomy of the system. Figure from S. Del Favero 'Control of Biological Systems course' @ UniPD (2022)[13].

algorithm employed. The complexity of the task and the demanding goal of maintaining blood glucose in the appropriate range over time, makes it necessary to use an adaptive, flexible and complex algorithm. The system should be able to take into consideration at the same time the long-term effects of the actions taken, due to long-lasting effect of insulin, and the possibility of sudden variations in blood glucose levels, due to meals or other disturbances. As a result, many different approaches are being tested by researchers, ranging from classic control engineering approaches, such as proportional integral control or model predictive control[14], to bio-inspired approaches, imitating the behavior of β-cells[15]. In the field of diabetes, Machine Learning (ML) methods have also attracted significant attention[16]. In particular, some variants of neural networks (NN) have performed particularly well in glucose forecasting[17]. But within the artificial intelligence field, another approach showing great promise, which allows for the implementation of all possible systems described, is Reinforcement Learning (RL). Indeed, over the last few decades, RL has frequently been the powerful framework behind great success stories that have made headlines[18, 19], as well as the foundation for several

applications and solutions in diverse healthcare domains such as long term management of chronic diseases and critical care, clinical diagnosis, and even clinical resource allocation and scheduling[20].

1.4 The reinforcement learning framework

Reinforcement learning is a branch of machine learning which deals with sequential decision-making problems, where a learning agent interacts with the environment. The environment typically reacts to the actions taken by the agent by transitioning to a new state; a numerical reinforcement signal, the reward, is also returned to the agent from the environment as a consequence of its action. A simple diagram of such a framework is shown in Figure 1.2.

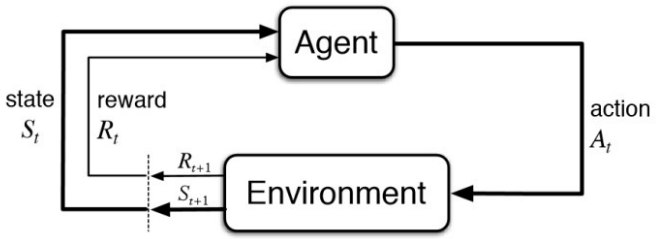

Figure 1.2: Basic Reinforcement Learning Framework. Figure from Sutton and Barto[21]

The decisions made by the agent depend on the observed current state of the environment and its past experiences of the rewards observed. This mapping between the observed state and the actions taken is known as the policy, and it completely defines the agent's behavior. The goal of RL is to learn the optimal policy for a given task, in such a way to maximize the reward obtained over time, also called the return. Several approaches are possible to reach the RL goal. The policy can be parameterized and learned directly (policy gradients methods), or it can be learned by gradually building up estimations of the expected return obtained for every state of the environment encountered (value based methods) and choosing the actions that lead to the best possible outcome, based on the estimations. The two approaches can also be combined, as it's done in actor-critic systems. Finally, some methods add another component to the agent, a model of the environment, which details the probabilities of transitioning between different states and receiving a certain reward after taking an action[21]. In Figure 1.3 an illustration of the main methods specified is shown.

RL is particularly well suited, due to the nature of its goal, for tasks characterized by inherent

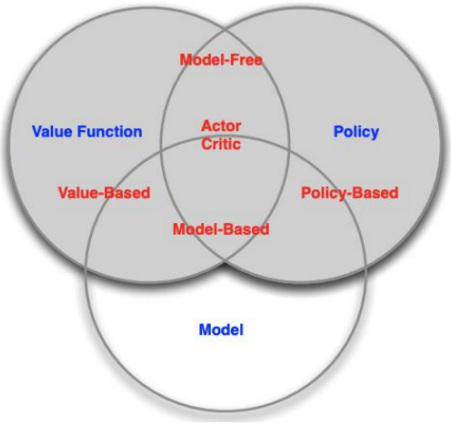

Figure 1.3: Reinforcement Learning methods. Figure from D. Silver 'RL course' @ UCL (2015)[22].

time delays, where the effects of the actions are possibly manifested at later points in time. In fact, sequential decision making tasks often have to deal with the trade-off between immediate rewards and future consequences. RL is able to naturally accommodate for this, as long as it's possible to accurately represent the states of the environment at every step. In the diabetes treatment task, such time lags are present due to both continuous glucose monitors (which actually measure subcutaneous glucose; the time required for plasma-interstitium glucose transport is about 10 minutes[23]) and to the prolonged insulin effect. Another advantage of RL as a framework for the control algorithm in an AP system, is that model-free RL allows to bypass the explicit modelling of the glucose-insulin dynamics. Furthermore, learning the optimal policy can occur over time by observing the responses of the environment, without the need for labeled data specifying which action to take in each instance, though prior knowledge on the task can still be advantageous. Finally, RL allows for personalized policies, adapted to patients' specific needs. All of these notions are strong arguments for the use of RL in an AP system[24].

1.5 Thesis contribution and outline

This thesis aims to discuss the employment of different RL approaches based on the algorithm Sarsa(λ), on in silico patients. The algorithm was never investigated properly in the artificial pancreas literature, but it's usually known to be appropriate for scenarios with long-term consequences or long delays in reward[21]. A quick review of the most relevant approaches already tested in the AP literature is presented in Chapter 2.

Chapter 3, instead, goes into detail about the description of the setup, the assumptions made

and the metrics used; it should be noted that this work has placed particular emphasis on tackling realistic meal schedules, considering scenarios that emulate as close as possible the distributions observed in patients with type 1 diabetes.

The description of the approaches proposed, going in detail on every component, is presented in Chapter 4. Sarsa(λ), the tabular framework, and the linear function approximation framework are described, along with a brief introduction of tile coding.

All experiments performed and the results obtained are discussed in Chapter 5. Final performances of the policies reached are analyzed and compared, between methods and subjects. Two aspects that have seldomly been considered in RL studies for AP systems are discussed in detail: the evolution of the training processes and the interpretability of the decisions taken by the agents.

Finally, concluding remarks and ideas for possible future directions are given in Chapter 6.

2

Relevant literature overview

The development of algorithms for both hybrid and closed-loop Artificial Pancreas systems is a very active research area. The task is being approached from several different angles by a large number of researchers in different fields. In this chapter, an overview of some of the main approaches being tested is presented: Section 2.1 will briefly cover traditional control approaches and mention some other important contributions in the field; Section 2.2 will cover more in depth some ideas of the applications for diabetes management making use of Reinforcement Learning.

2.1 Traditional control algorithms in AP systems

To this day, most approaches which are being clinically tested for AP systems, are using classic control engineering methods[25]. Among these, the two major contenders are for sure proportional integral derivative (PID)[26, 27, 28] and model predictive control (MPC)[29, 30, 31].

2.1.1 Proportional Integral Derivative (PID)

PID systems are characterized by three components[32]: the proportional action, which is proportional to the observed difference between current BG concentration and a reference level, the so called tracking error; the derivative action, which is proportional to the error derivative and intuitively should account for the error trend over time; the integral action, which is pro-

portional to the error integral, i.e. the accumulation of errors over time. An illustration of this is shown in Figure2.1. It should be noted, however, that most PID systems in AP turn out to be quite close to simpler PD systems, as the integral action is not so relevant in practice.

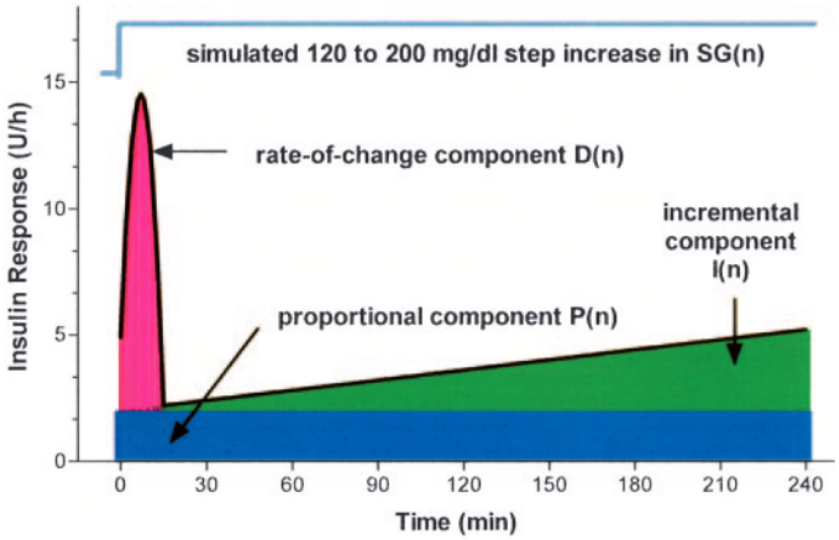

Figure 2.1: Simplified simulated response of a PID algorithm to an hyperglycemic clamp. Figure from G.M.Steil et al.,2006[26].

Basically, in the context of AP systems, PID algorithms estimate the amount of insulin to deliver, based on the weighted sum of the PID terms, in order to bring BG to the desired concentration, minimizing the errors. One of the main strengths of such systems is the intuitive interpretation of how each component (and hence the whole method) works. A particularly appealing (especially for medical practitioners) interpretation can be described, as stated by G.M.Steil et al.[26]: the integral component and the derivative component can be seen as producing respectively the slow second-phase rise and rapid first-phase rise, seen during hyperglycemic clamps. Manual tuning/adjustment of the algorithm parameters is hence simplified by a natural insight on their effect on the system. However, it should be kept in mind that it's crucial to tune the parameters to fit the characteristics of each diabetic patient: huge inter-individual variability, especially when talking about insulin sensitivity, i.e. how sensitive the body is to the effects of insulin, is always one of the main challenges in these systems. Other challenges that affect these systems are typically the possibility of over-delivery of insulin after large post-prandial peaks, leading to a larger percentage of time spent in hypoglycemia and possibly extra CHO assumption to balance the large control actions taken[26].

2.1.2 Model Predictive Control (MPC)

Model predictive control takes a completely different approach to the problem, aiming to calculate the optimal actions to take, based on the predicted effect of a sequence of N control-actions, through a model of the system. Once the first control action of the optimal sequence of actions is taken, the system waits for the new measurement to come in and then re-iterate the evaluation of the optimal sequence from the new state of the system. MPC techniques also appear to be well suited for glucose control in view of the fact that the optimization problem can account for constraints on the feasibility of a control action and on the acceptable states of the system. A simple illustration of the rationale of the method is shown in Figure2.2

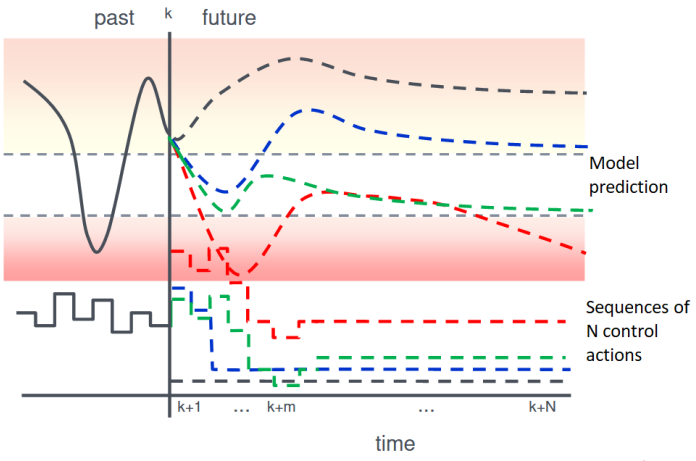

Figure 2.2: MPC can predict the effect of a sequence of control actions. It explores all possible sequences of N control-actions, and chooses the best one (in green), applying the first optimal control action. When the new measurement comes in, the process is repeated. Figure from S. Del Favero 'Control of Biological Systems course' @ UniPD (2022)[13].

While PID approaches are typically reactive, MPC is proactive, predicting the response of the system. For diabetes management, MPC methods require a dynamical systems model that is able to accurately predict future BG concentrations, given known features such as the current BG measurement, insulin delivery, and CHO intake. The appropriate insulin infusion rate is then calculated by minimizing the difference between the model-predicted glucose concentration and a target glucose level over a set time-window[33]. The mathematical model can be either based on prior knowledge (physiological, clinical, experimental) or can be learned from the observed behavior and can be either linear[34] or nonlinear[33]. Clearly, the accuracy of the model plays a key role in the final performance of the system: some studies tried to compensate for inaccuracies by trading off the rapidity of return to euglycemia with a more cautious

utilization of insulin, reducing the risk of overshoot into hypoglycemia[34], or by targeting a range of states as objective of the control actions, rather than a single one[35].

Over the past decade, MPC-based AP prototypes have undergone significant scrutiny: several clinical trials were conducted on patients affected by T1D, at progressively increased levels of autonomy. Initially, the experiments were conducted on hospitalized subjects [36], which proved to be safe, leading to a new wave of experiments in semi-controlled out-patient settings[37, 38] and, finally, even to the validation phase for in real-life autonomous use of such systems[39, 40]. Results obtained by these studies are encouraging and clearly support the notion that the use of AP systems at home can be a safe and beneficial option for patients with T1D, decisively outperforming standard patient-managed therapies.

2.1.3 Fuzzy logic

Some other remarkable approaches to date are fuzzy logic[41] methods. Fuzzy logic-based AP systems use a series of rules, developed by experts, which explicitly state when to deliver insulin and which is the appropriate dose, completely defining the policy to be followed. As PID algorithms, fuzzy logic approaches are strictly reactive and lack the presence of a model. A notable attempt in this direction is the so-called Medical Doctor Logic (MD-Logic) system[42], which is proposed as a wireless fully automated closed-loop system based on a fuzzy logic algorithm, along with personalized system settings and an alert module, including real-time alarms for both hypoglycemia and hyperglycemia. The performances obtained by this system are encouraging, as shown in studies achieving safe control both for patients under close observation[43] and at home[44]. At the very least, the results obtained underline that the use of logic, hard-coded rules, or, more in general, prior knowledge, have the potential to be an important component of AP systems, both for supervision in the early stages of the training of control algorithms and in terms of safety.

2.1.4 Multi-modular control

Given the complexity of the task and the need to guarantee safety, some studies considered multi-modular control approaches[45, 46], equipping their AP prototypes with an additional safety module which is able to override the suggestions made by the main control algorithm, reducing or stopping the proposed insulin infusion if the patient's safety is predicted to be at risk. The nature of the safety algorithm can vary as much as the main control algorithm, with some

approaches suggesting even the use of voting systems between many different methods[47], as a safety net.

2.2 Reinforcement Learning for diabetes management

Nowadays, applications using RL are booming throughout the most different fields of science, testimony to the flexibility of its framework. The use of RL for AP systems is no exception: several studies have been presented in the last decade, testing various RL methods for diabetes management. A recent review of the most relevant applications in this context was presented in Tejedor et al.[48]. Even so, the amount of research is nowhere close to that relevant to the methods discussed in the previous section. We're still very much in the early stages of the exploration of the possibilities that RL offers for diabetes management. So far, no clinical study on real patients making direct use of RL in an AP system is available: all research is based on in silico patients, making use of simulators[49, 50] and glucoregulatory models[51, 52], or, rarely, on past clinical data of real subjects[53].

The strategies employed for the use of RL in AP systems can vary widely, since a lot of components can influence the final performance. The usage of a model is rarely taken into consideration, as it seems so far that one of the main advantages of RL over MPC is the possibility to skip the modelling of the complex interactions between BG concentration, food intake, and insulin delivery[48]. On the other hand, the way the problem is represented in terms of features, action space, and type of reward signal is crucial. As exemplified in the studies reviewed by Tajedor et al.[48], various parameters can be considered when defining the state space, with the most popular features being BG concentration, past insulin doses, food intake and patient specific parameters as age, weight, and sometimes even physical activity[54]. The action space typically refers to the insulin dose to be delivered, either from step to step between CGM observations or as an indication to rise or diminish the infusion rate with respect to a previous period. Sometimes RL is also being used to calculate adequate food intake or physical activity[54], or to better estimate patient specific parameters: Jafar et al.[55] proposes a RL algorithm to optimize CHO ratio and programmed basal rate to improve the performance of a hybrid AP system. Finally, the reward function is arguably one of the most crucial components when designing a successful RL system. Through the reward function experts should be able to accurately express the preference for some desirable states and strongly penalize situations that may be dangerous and detrimental to the patient's health conditions. For diabetes management, the general idea is to consider a risk function that assigns the least risk to normoglycemia and gradually increases the

risk going deeper into hyperglycemia and hypoglycemia[56]. The correct specification of the reward function by experts is so important that some studies reported widely different results when changing only this component in the entire system[57]. The manual specification of possibly very complex reward functions, often resulting in sub-optimal performances, is an open challenge in the field of RL; recent proposals include methods which allow to learn the reward function directly from data[58].

All the components mentioned can be continuous or discretized in ranges, depending on the class of algorithm employed. So far, most studies focused on function approximation methods, especially making use of policy gradient and actor critic methods[59], directly parameterizing the policy and taking advantage of the fact that for these methods it's easy to consider continuous actions[21], as it can be the case for insulin dosing. Only a handful of researchers tested the more classic tabular approaches, focusing mainly on Sarsa and Q-learning[60, 61]. The reason for the preference of function approximation methods resides in the fact that it's often useful to use generalization techniques to transfer knowledge between similar states and actions, since it's often the case that in large and smooth state spaces we generally expect similar states to have similar expected returns and similar optimal actions[21]. Most recently, following the success in glucose forecasting using NNs[17], deep RL algorithms are taking the spotlight, exploiting complex neural network architectures to automatically extract the appropriate features of the problem[57, 62, 63].Although the results of these last approaches are quite promising, it should also be kept in mind that deep RL is often unstable and lacks mathematical convergence guarantees that some other function approximation methods have [21], resulting in poor worst-case performance. This is unacceptable for risky tasks like diabetes management. Fox et al.[62] tried to address such problems through an extensive process of model selection and random restarts.

When safety-critical tasks are considered, as it is the case in healthcare, often concerns not only include the obvious necessity of safety of the algorithms, but also their interpretability[20]. Artificial intelligence systems are unfortunately notorious for being considered obscure and hard to grasp in their decisions, especially for non-practitioners. Deep learning systems even more so. So far, in the context of RL for AP systems, very little effort has gone into providing explainability into both how the training of the algorithms proceeds and the way we can expect them to operate. Such issues will have to be more thoroughly investigated before possibly being able to move to clinical trials on real patients. This thesis will try, among other things, to introduce some methods that attempt to make a step in that direction, quite often trading off complexity of the system for increased ease of interpretation.

Another issue which has to be addressed is the classical dilemma of exploration vs. exploitation, i.e. the necessity to explore the state-action space in order to reach the optimal policy. Obviously, this is not easy for safety-critical tasks, like diabetes management. Ordinary RL exploration and exploitation strategies explore state and action spaces at random, gathering information, but only after a sufficient amount of knowledge is learned does the agent's behavior begin to improve dramatically. These randomized exploration strategies are clearly inadequate, since the execution of some actions can possibly lead to catastrophic states[64]. In the context of an AP system, this may be extremely detrimental to the patient's health and may result in possibly unrecoverable situations without an external intervention (e.g. extreme hypoglycemia). So far in the literature most approaches tested, being in silico, didn't cover enough of this issue, rather focusing on demonstrating the possibility to reach good policies after an initial stage of exploration, using mostly classic ϵ-greedy or greedy methods[48]. This way of proceeding essentially assumes that policies learned in silico could be easily transferred to real patients, with possibly minimal adjustments to be made. If this was not the case, to learn optimal personalized policies, which possibly adapt throughout the long term utilization of an AP system to changes in habits or physiology, future methods should probably be inspired by the growing field of Safe Reinforcement Learning[65] to account for continual online learning of the system. The exploration may be guided through the incorporation of external knowledge into the system[66, 67] or by directly taking into account the risk inherent in every action, which usually is related to its outcome variability[65, 68].

As mentioned, RL can also be envisioned to be used only in the simulator to train a policy which is then efficiently transferred to a real patient, without further or minimal training. Such an approach has seen only minimal recent exploration, mostly thanks to transfer learning techniques in deep RL[62]. Further research is needed in this direction as data efficiency is clearly an important step to be made to facilitate the transition to clinical studies. Other approaches taken in this direction, but not yet tested for diabetes management, include the possibility to combine supervised learning with RL [69], with the idea of learning from past clinical data a personalized initialization to the control algorithm, ensuring faster learning and convergence. Similarly, other approaches have been used in the literature for this purpose, using personalized estimated insulin-to-glucose transfer entropy[70].

3

Setup of the experiments

In this chapter a description of the setup used for the experiments is presented. Section 3.1 briefly introduces the UVa/Padova simulator; Section 3.2 lays out the details of the scenarios considered, underlying the assumptions and some technical implementations. Section 3.3, finally, covers the evaluation metrics which are used in Chapter 5 to compare the performances of different approaches or between different patients.

3.1 The UVa/Padova Simulator

The research for proper control algorithms for T1D management and the implementation of complete AP prototypes has been greatly accelerated in the last decade thanks to the development of effective computer simulations. In fact, in silico testing has been essential as a precursor to clinical studies, granting the possibility to test efficiently, in a cost-effective manner and, most importantly, safely, control approaches which could turn out to be ineffective[71]. The most important step in this direction has been made in 2008, when for the first time a computer model was accepted by the US Food and Drug Administration (FDA) as a substitute for preclinical trials, including closed-loop insulin delivery algorithms[49]. The UVa-PADOVA Type 1 Diabetes Simulator included a population of 300 in silico subjects (100 adults, 100 adolescents, 100 children), with the intention of providing a large enough cohort that can span well enough the key metabolic parameters inter-variability which has been observed in the population of people with T1D. Each subject in the cohort is represented by a parameter vector,

which is randomly sampled from an appropriate joint parameter distribution. The mathematical model which provides the basis for the in silico subjects simulation is described in detail in Dalla Man et al.[72, 73]. Early closed-loop trials observed a substantially larger frequency of hypoglycemia events in real patients compared to in silico ones. This notion, combined with the advent of new models to better describe hypoglycemia and counterregulation[74, 75], led the way to the update in 2013 of the UVa-Padova simulator[50]. Further studies confirmed that the updated version is indeed a valid tool which can be used to test the robustness of closed-loop control algorithms for AP systems, reproducing well the observations made with previous clinical studies[76].

The simulator is implemented in Simulink®, which is part of the larger scientific software MATLAB®, hence most of the methods which have been tested for this thesis are implemented in the same platforms, with the exception of some function calls to external Python libraries, when necessary. The first part of the experiments, which will be presented in Chapter 5, focuses on testing different approaches on the simulations obtained by the data generated from a single patient, in order to understand how every component influences the results that are possible to obtain and tune the methods. The second part, instead, considers the application of the methods tuned in the first part to a subset of 10 different subjects of the UVa/Padova simulator's adult cohort, analyzing the differences in results between methods and patients.

3.2 Simulation scenarios

All the agents are trained making use of the UVa-Padova simulator as environment to simulate episodes which are defined as progressively more realistic, and more challenging. In every episode CGM readings are provided every 5 minutes and the agent can decide to modify the insulin infusion rate after every observation. Initially, the feasibility of the algorithm employed is tested on a simple deterministic single-meal scenario, which also allows to make some considerations about the definition and representation of the problem. Following that, we test scenarios with multiple non-deterministic meals, attempting to closely mimic real-world meal schedules.

3.2.1 Meal schedules and variability

The simple single-meal scenario simulates the intake of a fixed amount of 80 g of CHO, 4 hours after the initialization of the agent, which is initially in normoglycemia at 150 $\mathrm{mg\,dL^{-1}}$. The

total duration of each episode is 20 hours, in order to observe if the agent is able to properly respond to the meal disturbance and, afterward, to stabilize the blood glucose level within the target range (70-180 $\mathrm{mg\,dL^{-1}}$), without falling into hypoglycemia due to an excessive reaction.

The multi-meals scenario, instead, consists of a random generation of a meal plan for each episode following Algorithm 3.1. Every episode is assumed to start in euglycemia at 150 $\mathrm{mg\,dL^{-1}}$ at 4 a.m. and lasts exactly 24 hours, within which the agent can consume up to 6 meals (breakfast, lunch, dinner, and 3 snacks) of different amounts of CHO. Quantities and times of the occurrences of the meals are sampled from truncated gaussian distributions and are assumed to be consumed instantly, for simplicity. The possibility to skip meals is also considered: in particular, main meals are expected to be consumed most of the times, while snacks are only occasionally eaten.

Algorithm 3.1 Meal schedule generator

Require: $var \geq 0$ ▷ variability of the meals in terms of percentage of the mean
$Meal.means \leftarrow [56, 20, 84, 20, 84, 20]$
$Meal.variability \leftarrow Meal.means * var$
$Meal.probs \leftarrow [0.95, 0.3, 0.95, 0.3, 0.95, 0.3]$
$Time.means \leftarrow [7.5, 10.5, 12.5, 16, 19.5, 22]$
$Time.vars \leftarrow [1, 0.5, 1, 0.5, 1, 0.5]$
$Time.LB \leftarrow [4, 8, 10, 14, 17, 20]$ ▷ Lower bounds
$Time.UB \leftarrow [11, 13, 15, 18, 22, 24]$ ▷ Upper bounds
for $j \in [1, ..., 6]$
$Occurrences[j] \sim Binomial(Meal.probs[j])$
$Amounts[j] \sim TruncNormal(Meal.means[j], Meal.variability[j], 0, \infty)$
$Times[j] \sim TruncNormal(Time.means[j], Time.vars[j], Time.LB[j], Time.UB[j])$
end for
return $Occurrences, Amounts, Times$

The mean amount and the variability of CHO consumed during the episode are of crucial importance to the performances that are possible to obtain. This aspect is considered by testing multi-meals scenarios of different variability, once again starting from easier scenarios and progressing to scenarios which try to emulate real distributions observed in the general population of T1D patients [77]. It should be noted, many past studies using RL for AP systems tend to stick to more conservative and rigid scenarios, even when including stochasticity in similar meal schedule generation algorithms[57, 62].

3.2.2 Safety assumptions and constraints

In a classic RL framework, the agent is usually free to choose the action that is evaluated to be the best in the current state or to explore with a small probability and eventually reach the optimal policy. Unfortunately, in the early phases of the training of an agent, the evaluations are usually wrong and the behavior of the agent is rather erratic, testing out multiple actions that are sub-optimal or even harmful. As mentioned before, in a safety-critical task, this is not acceptable and it's better to limit such freedom. Note that these limitations can not only improve the safety of the policy but even speed up the convergence of the training process. One reason for this is that merit attribution is tricky in the context of AP due to the long consequences of the actions and the delay between actions and consequences. An agent which is free to inject as much insulin as it wants at any BG level will initially learn quite slowly, as most episodes are spent in extreme hypoglycemia. In this situation, it should learn to not inject any insulin at all for several steps, while still possibly being severely punished for the effects of a strong previous action. Eventually, the agent would learn, but this long process can be completely skipped by simply overriding the main control algorithm in certain conditions, as it is done, for example, in multi modular systems that include safety modules[45]. In the scenarios tested, this simply translates into the imposed hard-constraint of not delivering any insulin when BG is below 90 $\mathrm{mg\,dL^{-1}}$. More complex constraints, which consider other features, could also be crafted, but since one of the objectives of the thesis is to figure out the pros and cons of RL in an AP system, no further limitations were imposed.

It should also be noted that, following the idea that the control algorithm should be freeing as much as possible the patient from any worry of manually adjusting blood glucose concentration, the scenarios do not take into consideration any interference from the patient. Users' most common strategy to avoid dangerous hypoglycemia and manage failures is to simply consume CHO snacks when below a certain BG threshold. Such a measure is implemented only once in the simulations, to briefly discuss its effects on the training process and the performance, but normally no such safety net is provided to the agents.

3.3 Evaluation metrics

In order to compare the performances of different methods, it's opportune to consider multiple evaluation metrics. The RL goal is to maximize the cumulative (possibly discounted) reward. We can get an initial, rough idea of how well a policy is working by measuring exactly that, for

every episode. Clearly, the way the reward function has been defined plays a huge rule in this and there might be occurrences where the agent is indeed optimizing for this signal, but behaving differently from how we'd ideally like. This may be due to an unexpected misalignment between our intentions and the reward function defined. A further restriction of considering cumulative reward by itself is that it's not easy to compare quantitatively, especially in rather different contexts. Given these limitations, it's clear that we have to consider other, more general, metrics in conjunction with the cumulative reward.

For diabetes management, we would ideally like the system to remain in normoglycemia most of the time and contain the excursions outside the target range (70-180 $\mathrm{mg\,dL^{-1}}$) as much as possible. It's only natural, hence, to include as evaluation metrics the percentage of time per episode spent in hyperglycemia (% hyper) and in hypoglycemia (% hypo), which are going to often be traded off between each other, depending on how cautiously the agent reacts in response to disturbances. Methods that minimize both these percentages, at the same time, are to be preferred. Note, however, that considering these two metrics is not enough; the percentage of time spent in an undesirable range doesn't take into account the intensity of risky events. For example, there would be no difference in perceived risk between an excursion of 2 hours in slight hypoglycemia reaching a minimum BG concentration of 55 $\mathrm{mg\,dL^{-1}}$ and a really dangerous event which takes the same time but reaches 20 $\mathrm{mg\,dL^{-1}}$. To account for this, several metrics based on proper risk functions have been introduced in the literature[56]. The low blood glucose index (LBGI) increases with the frequency and extent of hypoglycemic excursions and, by design, ignores hyperglycemia[78]; the high blood glucose index (HBGI) increases with the frequency and extent of hyperglycemic excursions and ignores hypoglycemia. These metrics, given CGM readings in $\mathrm{mg\,dL^{-1}}$ (x_i), can be computed as follows:

$$
\begin{aligned}
f(x_i) &= \ln(x_i)^{1.084} - 5.381 \\
r_l(x_i) &= 22.77 * f(x_i)^2 * \mathbb{1}(f(x_i) \leq 0) \\
LBGI &= \frac{\sum r_l(x_i)}{n} \\
r_h(x_i) &= 22.77 * f(x_i)^2 * \mathbb{1}(f(x_i) > 0) \\
HBGI &= \frac{\sum r_h(x_i)}{n}
\end{aligned}
$$

Note that the risk function $f(x_i)$ tends to greatly increase when reaching dangerous BG levels, and both LBGI and HBGI, being means over n observations, are dominated by the readings corresponding to such moments. This is particularly true for traumatic events: LGBI explodes in value when BG concentration is really low (0-5 $\mathrm{mg\,dL^{-1}}$), which can be seen as an indication of episode failure. The subdivision into classes of risk according to LBGI and HBGI is discussed, with some differences regarding the high risk area for HBGI, in Kovatchev et al.[79] and Scaramuzza et al. [80].

Analyzing training processes and final results using all of the metrics mentioned in conjunction allows us to properly compare different methods and gain an understanding of their benefits and drawbacks.

4 Methodology

The details of the methods proposed are outlined in this chapter. Section 4.1 goes into details about how every component of the RL framework has been represented. Section 4.2 describes the control algorithm Sarsa(λ) and the necessary notions of tabular and linear function approximation RL, including tile coding.

4.1 Reinforcement Learning components

4.1.1 State representation

Reinforcement Learning frames sequential decision making problems as Markov Decision Processes (MDPs), which assume the Markov property for every environment's state encountered by the agent. The description of the state, hence, should be a signal that summarizes past observations compactly, yet in such a way that all relevant information is retained. This normally requires building the state representation by using more than the current observation, but never more than the complete history[21].

In this respect, the task of diabetes management, although aligned with the RL framework ability to deal with the inherent time delay between actions and consequences, is tricky in its representation since we're arguably dealing, instead, with a Partially Observable Markov Decision Process (POMDP). Differently from MDPs, for POMDPs, the agent cannot observe the complete system state, but it makes observations that depend on the state, forming a belief

about the state the system is currently in. Indeed, not only is the current observation provided by the CGM reading clearly not sufficient to represent the state that we need, but also real sensors provide information with a certain degree of error[81]. Furthermore, similar errors can be introduced in hybrid systems when CHO counting, in any form, is considered [77] or when we estimate the amount of insulin acting in the patient, since insulin action varies between patients and even within each individual depending on the time of day, stress, and other factors.

Nonetheless, we can build up the state representation (or rather the current observation) as best as possible to provide the agent with the most important information needed. For our purposes, we choose to restrict these features to the few essentials, as to retain simplicity of the methods and easily inspect how policies change based on them. The most obvious feature to include is the current BG concentration. Secondly, the rate of change for the BG concentration allows the agent to understand in which direction the feature to control is going and at what speed. A third essential information for the agent is the so-called insulin on board (IOB), which is an estimation of the amount of insulin that is still active in the patient's body from previous doses; basically the whole insulin injection history is condensed into a unique value. In order to calculate it, the method expressed in Ellingsen et al.[82] was followed: the convolution of insulin injections history with a proper 6 hour insulin action curve returns an estimate of the current acting insulin in the system at each time step; an example is shown in Figure 4.1.

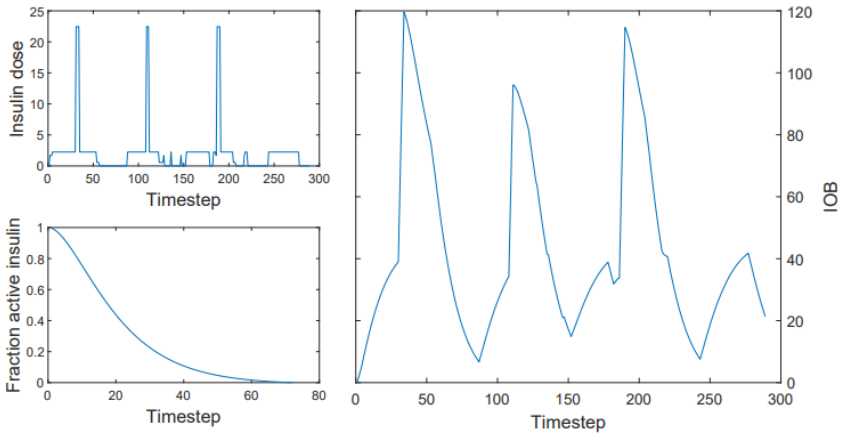

Figure 4.1: IOB (right plot) is calculated by convolution of the insulin history (upper left) with the 6 hour insulin action curve (lower left). At every step of the episode corresponds a value which is representative of the insulin in action.

Similar to IOB, when hybrid systems are briefly considered, the fraction of ingested carbohydrates not yet absorbed by the system at each time t, called Carbohydrates On Board (COB), can be computed as reported in Schiavon et al.[83]. Figure 4.2 shows how, from a daily meal

schedule, it's possible to obtain at every instant the current COB by simple convolution with the CHO absorption curve.

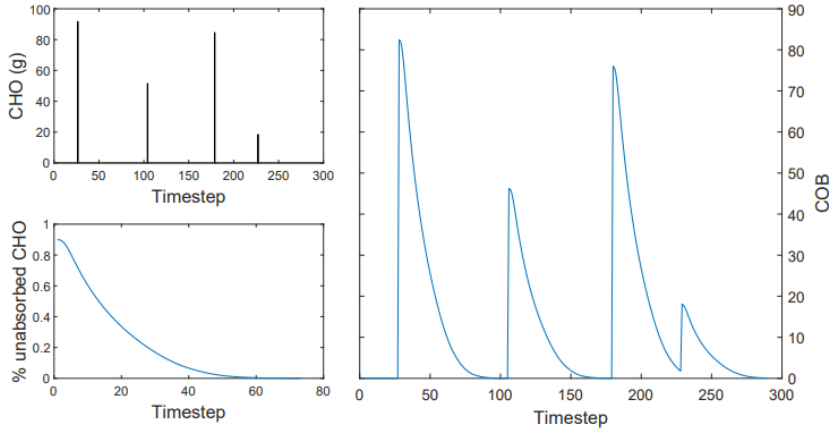

Figure 4.2: Carbohydrates on board (COB) are calculated by convolution of the daily meal schedule (upper left) with the 6 hours carbohydrates absorption curve (lower left). The resulting COB profile is shown on the right.

4.1.2 Action sets

For the approaches proposed, it's necessary to consider discrete action sets. Three action sets are tested in the single-meal scenario, settling on the one considered best for the multi-meal scenario. The idea behind the selection of these action sets is intuitive: the agent should have the possibility to easily maintain BG concentration within the target range when no disturbances occur, but should also be able to respond more vigorously when a meal is detected. Furthermore, as mentioned, since the inter-person variability associated with insulin response is large, it's probably beneficial to consider action sets that are somewhat related to it. Action sets are, hence, chosen to be multipliers of the basal rate of the patient.

The first action set selected is rather conservative, allowing only insulin doses close to the patient's basal rate (0 to 2 times); the second allows for stronger but not yet radical actions (0 to 6 times the patient's basal rate); the third considers also radical actions (up to 20 times the patient's basal rate), which should be used by the agent to quickly fight BG concentration spikes, but inevitably introducing bigger risks of hypoglycemia if abused.

4.1.3 Reward functions

As mentioned in Section 2.2, the choice of the reward function is quite important when using RL in an AP system. Rather different performances can be obtained using the same approach by changing marginally only this component.

The general idea for diabetes management is to consider adequate risk functions[56] and adapt them to the RL framework. This adaptation can be carried out by simply inverting the risk functions (since we want to maximize a reward signal) and assigning positive rewards relative to the permanence of the agent in the proper target range. The Kovatchev reward function is derived as described from Kovatchev's risk function[56] (on the left in Figure 4.3). Note that in this case, though, the rewards barely vary within normoglycemia, so the agent can't easily develop a preference for a target value (e.g. $125\ \mathrm{mg\,dL^{-1}}$) within the range, which is something we might want to communicate within a reward function. It's possible, instead, to consider reward functions that linearly approximate the behaviour of the Kovatchev reward function in hypoglycemia and hyperglycemia, but attribute a quadratic improvement in normoglycemia, with the vertex being the target value we'd like the agent to stay closest to. The height of the vertex, which corresponds to the maximum reward in a time step, can be appropriately tuned.

In the simplest case, the vertex can be fixed to be exactly in the middle ($125\ \mathrm{mg\,dL^{-1}}$, for a target range of 70-180 $\mathrm{mg\,dL^{-1}}$), but an alternative is to split euglycemia with two different quadratic functions sharing the vertex and move it closer to either hypoglycemia or hyperglycemia, with the intent to encourage the agent to stay closer to that region. We call the first case the LQL (linear-quadratic-linear) reward function (middle of Figure 4.3) and the second one the LQQL (linear-quadratic-quadratic-linear) reward function (on the right in Figure 4.3).

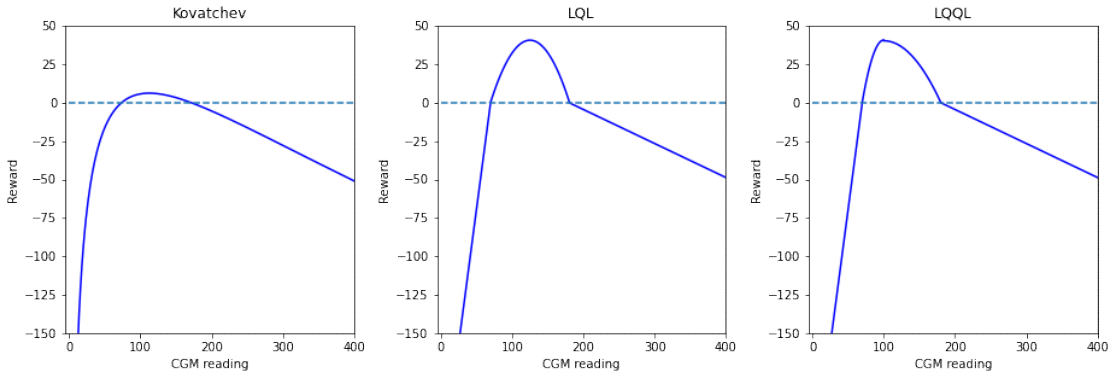

Figure 4.3: Reward functions dependent on BG level only: Kovatchev risk-inspired (left), LQL (middle) and LQQL (right).

Further information can be introduced into the reward function: the agent should be rewarded not only for the current blood glucose level but also according to the fact of whether it's proceeding in the right direction or not, therefore the reward should take into account the BG rate. This sensible approach was proposed in Lee et al.[63] and is emulated here. All reward functions defined before are compatible with this approach, as the portion of reward dependent on the rate can be defined as independent from the part associated with the concentration. Plots exemplifying the effect of the rate on the base reward functions are shown in Figure 4.4. It's easy to see, from the frontal plots, that the base reward functions define the overall shape, but lateral views show that the agent receives larger rewards when the rate suggests a forthcoming return to normoglycemia: in hypoglycemia (left view) the reward is larger for positive rates, while the opposite is true in hyperglycemia (right view).

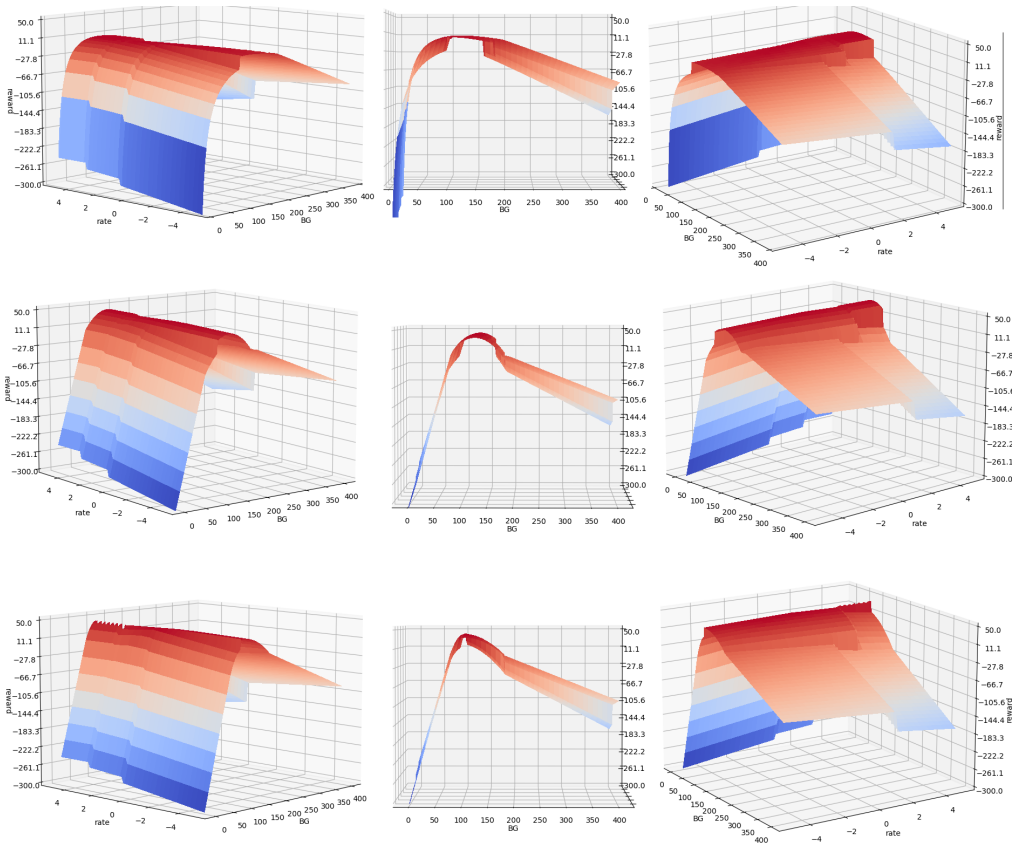

Figure 4.4: Reward functions dependent on both BG concentration and BG rate. From the top: Kovatchev risk-based, LQL-based and LQQL-based.

4.2 Reinforcement learning methods employed

All approaches tested in this thesis are based on the Sarsa(λ) algorithm. This choice is due to essentially two main reasons. Firstly, it has not been explored at all in the RL for AP systems literature: neither in studies based on tabular methods, where standard Sarsa and Q-learning have usually been preferred, nor in function approximation articles, where policy gradient and actor-critic algorithms are dominant instead. Secondly, there are clear indications in the literature that methods based on eligibility traces are particularly well suited to approach partially observable markov decision processes[84]. In fact techniques which are specifically engineered for POMDPs are often compared to the performances obtained by using Sarsa(λ)[85]. This is mentioned by Sutton and Barto too: "Eligibility traces are the first line of defence against both long-delayed rewards and non-Markov tasks"[21]. Since, as argued in Section 4.1.1, in AP systems we're dealing with scenarios where the state is not fully observable and actions have long-term consequences, it's only logical to test out the method which is considered to be the most robust in such circumstances. Furthermore, Sarsa(λ) is arguably one of the most flexible classic RL algorithms: eligibility traces unify and generalize Temporal Difference to Monte Carlo methods, incorporating all possible n-steps returns at once. Assuming it's properly tuned, such flexibility can intuitively lead to increased performance, with respect to more rigid approaches, and faster learning, particularly when rewards are delayed by many steps [21].

In the following sections an introduction to Sarsa(λ) is presented, for reference, along with some necessary definitions and principles of tabular and linear function approximation RL.

4.2.1 Sarsa(λ) in tabular RL

Sarsa(λ) is a model-free action-value based control algorithm. It builds up estimations of the expected return for pairs (state, action), called Q-values, eventually leading to good policies (optimal if certain conditions are met). Return and Q-values in tabular RL are defined as follows:

$$\begin{aligned} &G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = R_{t+1} + \gamma G_{t+1} \\ &Q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \end{aligned} \tag{4.1}$$

where $0 \leq \gamma \leq 1$ is a discount factor and π is the policy being followed. N-steps returns can then be expressed by bootstrapping to the Q-values after observing n rewards:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n}) \tag{4.2}$$

RL algorithms employ different versions of n-steps returns, ranging from usual Sarsa / TD-learning methods, where 1-step returns are used, to n-step methods, using a specific n-step return, and finally to Monte Carlo methods that do not bootstrap but rather compute the return as the cumulative sum of all rewards, possibly discounted, in the episode, which can be thought of as computing a n-step return with n = ∞. Based on the choice of n, the performance of the methods varies, sometimes widely. The λ-return G_t^λ can be defined as a convex combination of all possible n-step returns:

$$G_t^\lambda = (1-\lambda) \cdot \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \quad 0 \leq \lambda \leq 1 \tag{4.3}$$

The weights fade by λ with each additional step, giving more importance to close returns. In the case of $\lambda = 0$, the G_t^λ return is equal to the 1-step return, as in standard TD-methods; while, on the other extreme, if $\lambda = 1$, we get a method equivalent to Monte Carlo algorithms. Essentially, the λ-return provides a way to move smoothly between MC and one-step TD methods. Sarsa(λ) employs the λ-return as a target by updating at every step the estimates of the Q-values:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(G_t^\lambda - Q(S_t, A_t)) \tag{4.4}$$

The reasoning reported so far is called the forward-view of Sarsa(λ) and it's useful to provide an intuitive explanation of how the algorithm works, but its implementation is tricky: G_t^λ can only be computed at the end of each episode, since all possible n-step returns are needed, hence it can't be done online and all rewards of the episode must be kept in memory. Only at the end of the episode are all Q-values visited updated. The real implementation of Sarsa(λ) is through its backward-view, using eligibility traces, which allows for online learning. The idea is to attribute the merit (or fault) of the observed rewards to the (s,a) pairs encountered in the episode, proportionally to how recently they were visited. Eligibility traces keep track of the encounters, decaying exponentially in their values at every step of the episode. There are different types of eligibility traces; the most commonly used are accumulating traces, defined as follows:

$$\begin{aligned} &E_0(s,a) = 0 \quad \forall(s,a) \\ &E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + \mathbb{1}(S_t = s, A_t = a) \quad \forall(s,a) \end{aligned} \tag{4.5}$$

The trace E(s,a) indicates the eligibility of each (s,a) pair for undergoing learning changes when a reinforcing event occurs[21]. The reinforcing events we are concerned with are the one step TD-errors:

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \tag{4.6}$$

TD-errors and eligibility traces are then used to update at every step all Q-values estimations, progressively refining them and improving the policy being followed:

$$Q(s,a) = Q(s,a) + \alpha \delta_t E_t(s,a) \quad \forall (s,a) \tag{4.7}$$

Although they seem rather different, forward-view and backward-view of Sarsa(λ) are two ways to implement the same idea and converge to similar results; for some implementations it's possible to prove that they're actually equivalent[21]. Figure 4.5 shows two illustrations to provide an intuition of the ideological difference between the two views: in the forward-view the update of Q-values is done by looking forward at future rewards and future (s,a); in the backward-view at each step we look at the current TD error and propagate it backward to each prior (s,a) encountered, according to how much it contributed, as measured by its eligibility trace. The algorithm is extendable to function approximation and differences will be briefly described in the next section.

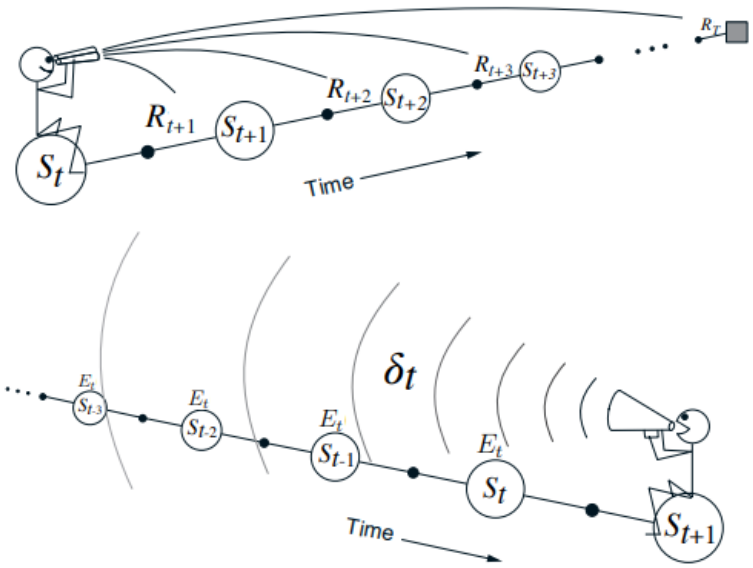

Figure 4.5: Illustration of the mechanisms of Sarsa(λ): forward view (above) and backward view (below). Note how we need all the episode's rewards to implement the first, while the second can be implemented online by propagating reinforcing events over time to the responsible states, as measured by eligibility traces. Figure from Sutton and Barto [21].

4.2.2 Sarsa(λ) in linear function approximation

Function approximation is an instance of supervised learning: using examples from a desired function (for example, the action value function), it generalizes from them, building up an approximation of the entire function. This allows to extend the familiar RL framework in order to deal with large problems or to impose generalization constraints by default; furthermore, function approximation methods are also particularly well-suited for partially observable problems[21]. While in tabular methods action values are independent entries of a table, function approximation methods approximate them through a parameterized functional form using a weight vector $\mathbf{w} \in \mathbb{R}^d$. When an approximate action-value is updated, the change affects many other close state-action pairs. The function computed can be of any level of complexity, but here only linear methods will be considered. In this context, corresponding to every (s,a) pair, exist a real-valued feature vector $\mathbf{x}(s,a) \in \mathbb{R}^d$; the approximate Q-values are computed by the inner product between the weight vector $\mathbf{w}$ and the feature vector $\mathbf{x}(s,a)$:

$$q(s,a) \approx \hat{q}(s,a,\mathbf{w}) = \mathbf{w}^T\mathbf{x}(s,a) = \sum_{i=1}^{d} w_i x_i(s,a) \tag{4.8}$$

The weight vector $\mathbf{w}$ is updated at every step by stochastic gradient descent, in order to minimize the MSE ($J(\mathbf{w})$) between the approximate action values and the true action values:

$$J(\mathbf{w}) = \mathbb{E}_\pi[(q_\pi(S,A) - \hat{q}(S,A,\mathbf{w}))^2] \tag{4.9}$$

$$\Delta\mathbf{w} = \alpha(q_\pi(S,A) - \hat{q}(S,A,\mathbf{w}))\mathbf{x}(S,A) \tag{4.10}$$

Since the true action values are never known (otherwise the problem would be solved), they have to be estimated. The nature of the target values depends on the method employed: in Sarsa(λ) the λ-return is used, a biased sample of the true action value. Once again, though, the implementation of Sarsa(λ) in function approximation is done through its backward-view: an eligibility trace vector $\mathbf{z_t} \in \mathbb{R}^d$ is initialized to zero at the beginning of each episode, is incremented on each time step by the action value gradient (which is equal to the feature vector of the current (s,a) pair, in linear function approximation), keeping track of which components of the weight vector have recently contributed, and then fades away by $\gamma\lambda$[21]:

$$\mathbf{z_{-1}} = \mathbf{0}; \quad \mathbf{z_t} = \gamma\lambda\mathbf{z_{t-1}} + \mathbf{x}(S_t, A_t) \tag{4.11}$$

The reinforcing events we are concerned with are still one-step TD-errors:

$$\delta_t = R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}) \tag{4.12}$$

Finally, the weight vector gets updated on every step based on the eligibility trace vector and the current TD-error, affecting the action value estimation of all (s,a) pairs that are computed using the components of the weight vector that changed:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \delta_t \mathbf{z_t} \tag{4.13}$$

Linear learning methods are of particular interest because, in practice, they can be very efficient, both in terms of data efficiency and computation. Whether or not this is the case, critically depends on how the states are represented: choosing carefully the features appropriate to the task is an important way to add prior domain knowledge into a RL system. Intuitively, the features should correspond to the natural features of the task, along which generalization is most appropriate. There are several ways to define the features associated with each (s,a) pair: in this thesis, tile coding, briefly explained in the next section, has been employed.

4.2.3 Building linear features : tile coding

As we'll see in Section 5.1.5, the ideal system generalizes between close (s,a) pairs but also allows for fine discrimination, when necessary. In an AP system, most close observations demand for similar evaluation and action, but a sudden change in policy (from basal rate to vigorous actions) is needed, as soon as a meal is detected. Tile coding allows for this kind of behavior.

Tile coding is a flexible and efficient form of coarse coding for multi-dimensional continuous spaces: the state-action space is divided into several partitions, called tilings, whose elements are called tiles. The partitions built are overlapping but offset from one another, by a fixed amount in each dimension. Any (s,a) pair can be associated with a binary feature vector **x**(s,a) of dimensionality equal to the total number of tiles, such that each component is associated with a specific tile and is 1 if the (s,a) pair falls into that tile (in this case the feature is said to be present), or 0 otherwise (the feature is absent). Given a (s,a) pair, it falls into exactly one tile for each tiling; thus, if k tilings are considered, each (s,a) pair is associated with k tiles and its binary feature vector has exactly k present features. Essentially, binary features roughly encode the location of a (s,a) pair in the state-action space: since corresponding to each feature is a single component of the weight vector **w**, which is affected by learning, generalization occurs between (s,a) pairs

that share the same present features, proportionally to the number of features shared. The nature of generalization is determined by the shape of the tiles, but the finest discrimination possible is ultimately controlled by the total number of features (the number of tilings and tiles per tiling)[21]. A further advantage of tile coding is computational efficiency: approximate action values can be computed by simply summing up the few components of the weight vector associated with the present features of the (s,a) pair[21].

Tilings are easy to visualize when thinking two-dimensionally, as a simple grid (see Figure 4.6 for an intuitive example), but in the context we'll consider, using BG concentration, BG rate, IOB, and insulin dose, 4D hyper-grids must be employed.

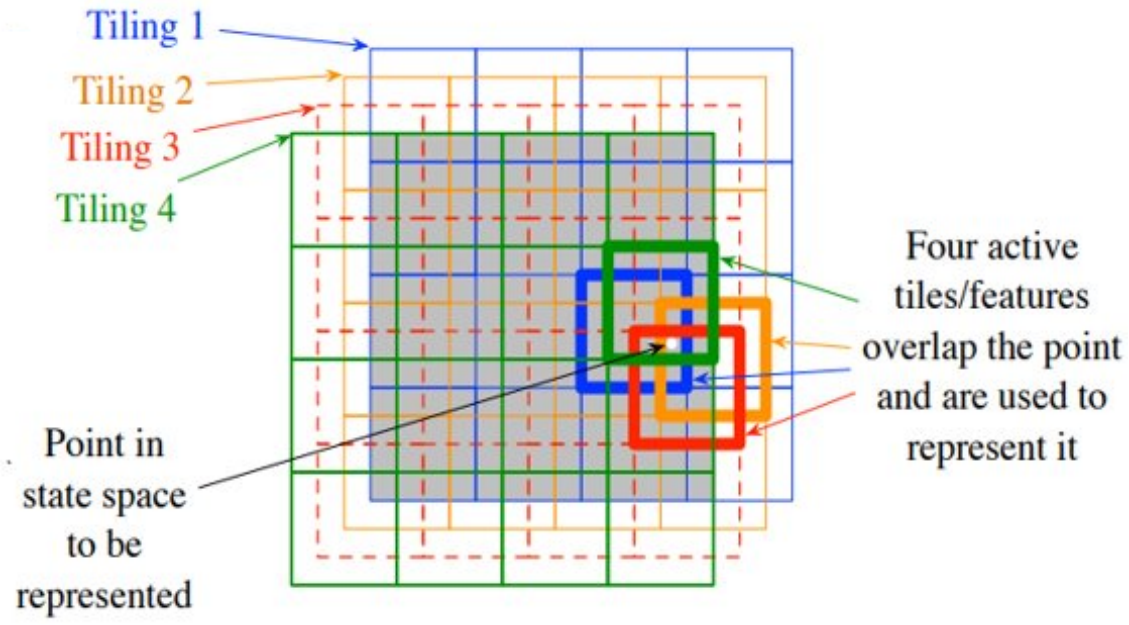

Figure 4.6: Example of tile coding with a group of 4 overlapping grid-tilings on a two-dimensional space. Tilings are offset from one another by a fraction of a tile width. A specific state falls into 4 tiles, one for each tiling, roughly encoding its position. Figure from Sutton and Barto[21].

An interesting idea which extends the framework proposed above is that multiple different groups of tilings can be considered, independently of one another: a great deal can be done to control the tendency to generalize and allow discrimination, depending on the implementation. For example, one could envision a system that generalizes partly along a subset of dimensions separately, but also along the entire set of dimensions jointly, such that, if needed, it could learn a specific response, demanded by that particular state configuration. These design choices will be discussed thoroughly in Section 5.2.1.

5 Experiments and results

The experiments performed are described in this chapter, along with interpretations of the results and comparisons. Section 5.1 presents the results obtained by tabular RL, discussing how each RL component influences the results and providing a direct interpretation by table inspection. Afterward, Section 5.2 discusses the results obtained by linear function approximation, underlying pros and cons of generalization, with respect to tabular methods and once again providing visual explainability of the results. Finally, Section 5.3 is dedicated to the topics of generalization of the methods to multiple patients, policy transfer and personalization.

5.1 Tabular RL approaches

In this section the tabular methods and experiments carried out are described in detail. In order to use such techniques, the state and action spaces must be small enough[21]: to make it viable for an AP system, the features employed to describe the state have to be few and discretized into ranges; discrete actions have to be considered too. A Q-values table is initialized and to each entry corresponds the estimation of the expected return obtained from a (state, action) pair. In practice, the table is implemented as a tensor, where each dimension is associated with a feature of the space and the last one with the action. The number of entries in the table depends on the number of features considered and on the finesse of the discretization process; the speed of learning achievable is strictly correlated to the dimension of the table.

The single deterministic meal scenario is described first: it was explored mainly to understand which of the RL components described in Section 4.1 had the potential to work best, before moving to the more realistic stochastic multiple meal scenario, which is then treated in detail.

5.1.1 Introduction to the task: single meal scenario

In this scenario, a single meal of 80g of CHO is consumed precisely 4 hours from the beginning of the episode, which starts in euglycemia at $150\,\mathrm{mg\,dL^{-1}}$ and lasts 20 hours. The agent is aided in the control task by automatic injection of the patient's basal rate at each time step, so that it should only take care of disturbances. The state representation is defined by discretizing BG level, BG rate, and IOB into a few ranges, as follows:

$$\text{BG ranges} = [[0, 90), [90, 140), [140, 180), [180, 250), [250, 300), [300, 400), [400, \infty)]$$
$$\text{Rate ranges} = [(-\infty, -3), [-3, -2), [-2, -1), [-1, -.3), [-.3, .3), [.3, 1), [1, 2), [2, 3), [3, \infty)]$$
$$\text{IOB ranges} = [[0, 2), [2, 5), [5, 10), [10, 20), [20, \infty)]$$

The ranges of BG concentration are crafted in order to properly define regions where we intuitively expect the agent should behave differently: in the first range, as mentioned in Section 3.2.2, as a safety constraint, the agent must avoid injecting additional insulin. The following two ranges are splitting normoglycemia into the range we'd like the agent to stay in most of the time (90 to $140\,\mathrm{mg\,dL^{-1}}$) and one which is still acceptable but it's most likely encountered right after a meal occurs. Finally, hyperglycemia is divided into the last 4 ranges, corresponding to increasingly dangerous BG levels. The finesse in rate ranges allows the agent to understand properly how fast blood glucose concentration is varying; in particular, at the two extremes of the subdivision we have ranges which are relative to rapidly falling BG and rapidly rising BG. Note that the ranges are to be intended with unit of measure $\mathrm{mg\,dL^{-1}\,min^{-1}}$, hence a BG rate of -3 $\mathrm{mg\,dL^{-1}\,min^{-1}}$ corresponds to two subsequent observations, where BG has dropped by $15\ \mathrm{mg\,dL^{-1}}$. Finally, IOB, in this case, is to be intended as additional insulin w.r.t. basal insulin and it's split into 5 ranges. Ideally, the agent should be able to infer if the level of insulin injected is adequate with respect to the current BG concentration and the rate of change.

Multiple agents were trained, testing the Kovatchev risk-inspired reward function and the LQL reward function, along with the 3 action sets proposed in Section 4.1.2. Kovatchev reward function turned out to be strictly worse than LQL, not only suffering in performance, due to the agent spending more time in hyperglycemia, but also having problems with convergence,

especially when action sets with radical actions were employed. Upon further investigation, as it is the case for the LBGI metric, using a reward function that is defined using a logarithmic function, seems to not be appropriate for episodes that reach complete failure (BG levels below 5 $\mathrm{mg\,dL^{-1}}$), as they end up dominating in signal the others, hurting convergence.

The action set had a huge influence too, both on the results attainable and on the speed of the training process. In all runs, the cautious action set, which only allows the delivery of additional insulin between 0 and 2 times the patient's BR, had a clear disadvantage in terms of performance with respect to the two other action sets, which allow stronger actions in order to respond more decisively to meals; at the same time, though, the number of episodes necessary to converge to good policies was observed to increase substantially as stronger actions were available to the agent. This is due, in part, to the increase in size of the Q-values table (from 1800 entries for the cautious action set to 3600 entries for the larger action set with strong actions), but it's also due to the fact that the agent has more difficulty understanding when stronger actions are appropriate; there's way more risk involved in the training of policies when such actions are available. Table 5.1 provides a summary of the results obtained using the LQL reward function and the three action sets. Note the trade-off between the number of episodes needed for convergence and the performance of the method. The word convergence here, and in all following tables, is to be intended in a qualitative sense, as the number of episodes needed to reach proper policies and not observe further progress; later on stochasticity of the episodes generated will imply a degree of variation in performance even if the policy remains equal. In any case, for the single meal scenario, after reaching proper policies, the agent is always able to be hypoglycemia-free.

Examples of the policies reached with the three action sets are shown in Figure 5.1. Such episodic profiles will often be presented; in the first subplot the blood glucose concentration throughout the episode is shown; a horizontal dashed line indicates the reference level of 125 $\mathrm{mg\,dL^{-1}}$, which we'd like the agent to stay close to. In this case, the sharp increase in BG concentration at 240 min is clearly due to the fixed meal. Below that, the actions taken by the agent are shown, i.e. the insulin profile. Finally, the last subplot illustrates the reward received

Action Set	# Table entries	Cumulative reward	Ep. to converge	% Time hyperglycemia	% Time hypoglycemia	HBGI
0x to 2x BR	1800	4500	100	23.3	0.0	9.5
0x to 6x BR	2520	6000	600	17.5	0.0	6.8
0x to 20x BR	3600	5800	1250	15.4	0.0	5.1

Table 5.1: Summary of results obtained for deterministic single meal scenarios using LQL reward and different action sets.

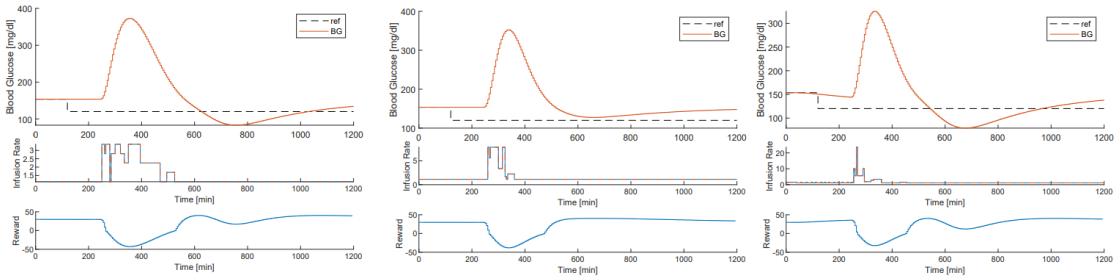

Figure 5.1: Comparison of reaction to a single fixed meal by three agents based on different action set. Agent on the left can only deliver up to 2 times the patient's BR; agent in the middle, up to 6 times the BR; agent on the right, up to 20 times the BR.

by the agent at all steps of the episode. The insulin subplots relative to the three action sets clearly demonstrate the different behaviors of the three agents: while the agent who can use only weak actions (on the left) is forced to deliver insulin for long periods after the detection of the disturbance, agents with access to stronger actions are able to react quickly to the meal and minimize the height and width of the post-prandial peak. This is especially true for the last agent (on the right), who responds quite vigorously for only a couple of time steps after the meal (note the different scale in the insulin profile), and then switches to delivering weaker control actions; if it decided, instead, to deliver even more insulin, it would inevitably end up in hypoglycemia.

Several tests were also run to understand the effect of ϵ-greedy exploration on the performance of the agent. The results showed that exploration, especially when strong actions are available, can cause sharp drops in performance, possibly even for prolonged amounts of time. Upon further inspection, the problem is quite clear: the estimations of the Q-values built up over time get swayed substantially by the large TD-errors obtained in episodes where exploration causes deep dives into hypoglycemia. The problem resides in the very long-term consequences of strong actions: the penalty is not only attributed (rightfully) to the action that caused the failure of the episode, but also to subsequent (s,a) pairs that would normally be the right choice, destroying their evaluations and consequently choosing actions that are suboptimal in the next episodes, possibly even for a long time. This problem is observed, in minor measure, even for small values of ϵ (e.g. 0.01). It's hard to see, in this context, a clear advantage in an ϵ-greedy approach to exploration, as the performances reached by even the best episodes, when using it, are not at all far off from those obtained by approaches without exploration.

Finally, the question that obviously arises, before moving on to the multi-meal scenario, is: are policies learned capable of generalizing to meals of different sizes and/or at different tim-

ings? As it turns out, while timing isn't much of an issue, since agents don't really have access to a feature to differentiate states by time, the amount of CHO consumed in a meal is quite problematic. Having been trained on deterministic fixed meals, the policies learned optimized responses to that amount, since it's all they've ever seen. By simply halving the amount of CHO in the meal, episodes fail catastrophically. For example, using the best performing policy, where strong actions are available, the agent reacts too aggressively to the meal detected and, as a consequence, BG drops sharply into deep hypoglycemia; even worse, as BG dives, the agent delivers even more insulin as it has no idea of what to do in some (s,a) pairs which were not met in the scenario it was trained on. An example of this is shown in Figure 5.2.

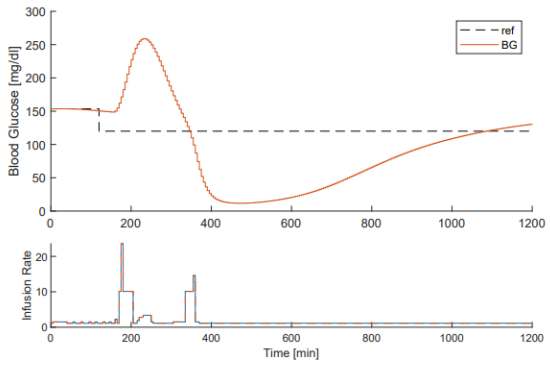

Figure 5.2: Example of episode failure when meal CHO amount is halved (from 80g to 40g): the agent, trained solely on deterministic single meals, fails to generalize to other scenarios and overreacts to the disturbance.

As predictable, in order to obtain agents capable of reacting appropriately to different contexts, it's necessary to consider more variable and realistic meal schedules, in terms of quantity of carbohydrates, timing, and number of daily meals. Much of what has been learned in the simple single fixed meal scenario can be quite useful for the new problem to be considered, such as what can be expected from different action sets, different reward functions, and problems due to exploration. Furthermore, the fact that Sarsa(λ) works appropriately for the simple scenario also bodes well for its application to the realistic case.

5.1.2 Multiple meals scenario introduction and improvements

The algorithm used to generate stochastic meal schedules for the multiple meals scenario is described in detail in Section 3.2.1. State representation is identical to the previous scenario, except that the additional ranges ($[20, 40), [40, 60), [60, \infty)$) are added for the IOB feature,

since a lot more CHO are usually consumed throughout the episode, implying that more insulin is expected to be administered overall.

The variability for the amounts of CHO is initially set to 15% of the mean amount for every meal, as done in several previous studies with stochastic meal schedules. The initial experiments proceeded exactly as done for the single-meal scenario, testing out Kovatchev-based and LQL-based reward functions using different action sets. Once again, the Kovatchev risk-based reward function turned out to be problematic when using stronger actions, hence its use was definitively discarded. Automatic administration of basal rate, which is supposed to help agents by allowing them to focus on dealing exclusively with disturbances, is also switched on and off in different runs, to see if it's actually helpful. The results obtained, using the LQL reward function, are summarized in Table 5.2.

Action Set	Automatic BR	Cumulative reward	Ep. to converge	% Time hyper	% Time hypo	% TIR	HBGI	LBGI
0x to 2x BR	Yes	-5700	1100	51.1	4.4	44.5	14.6	2.0
0x to 2x BR	No	-5200	1000	50.4	3.6	45.0	13.6	1.7
0x to 6x BR	Yes	-2200	2400	36.2	5.3	58.5	10.6	2.4
0x to 6x BR	No	-500	1300	35.4	3.0	61.6	11.2	0.7
0x to 20x BR	Yes	-6400	4400	41.9	11.1	47.0	11.4	5.6
0x to 20x BR	No	300	2300	33.3	1.3	65.4	10.2	0.9

Table 5.2: Summary of the results obtained with a 15% variability, using LQL-based reward, different action sets and switching on and off automatic basal rate. % TIR is to be intended as percentage of time in range, i.e. time in normoglycemia.

From the results, a couple of facts are clear: firstly, the availability of stronger actions seems to be absolutely necessary in multiple meal scenarios: since prolonged low doses of insulin are not enough to fight off disturbances, agents with access only to weak actions end up spending most of their time in hyperglycemia; secondly, automatic basal rate is harmful, rather than helpful, especially for agents that react vigorously to meals, since it hinders the possibility to avoid overshooting into hypoglycemia by completely cutting off further insulin delivery, if necessary. Avoiding automatic basal rate delivery improves both performance and convergence speed. Note, though, that we're not able to completely avoid hypoglycemia, as it was the case in the deterministic single meal scenario.

Some examples of episodes with rather different meal schedules, making use of the policy developed by the best performing agent of Table 5.2, are shown in Figure 5.3: CHO amounts and times of consumption are plotted in the lower subplots; vertical dashed lines are also drawn in the BG concentration profiles and insulin profiles in correspondence to the meals. The policy is clearly able to deal solidly with the scenarios encountered, with the exception of the last

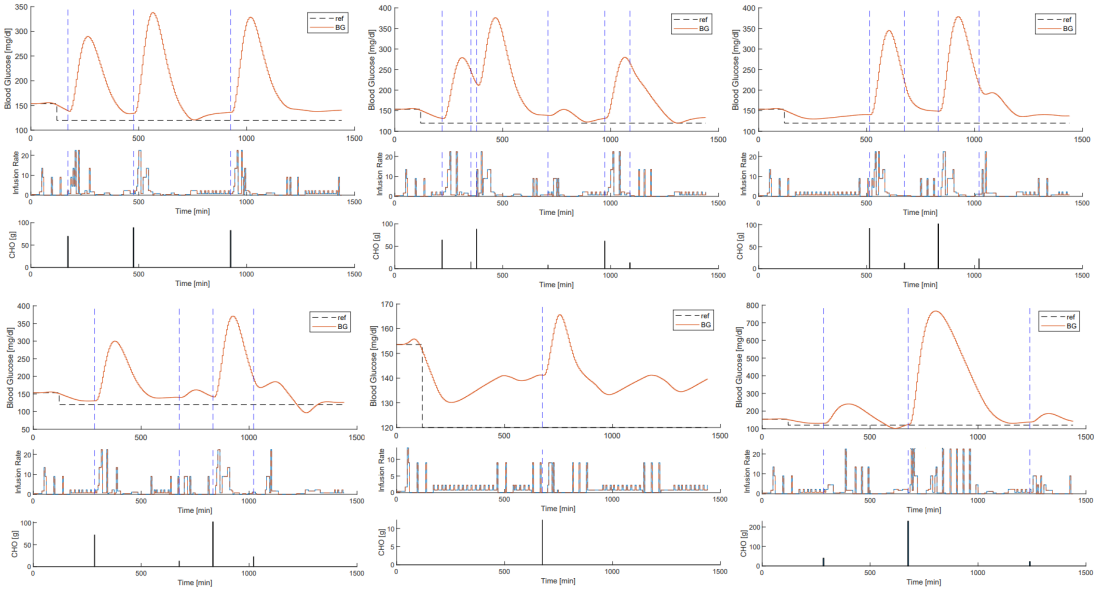

Figure 5.3: Examples of control in episodes with different meal schedules. Top left: standard 3 main meals schedule. Top middle: all 6 meals, 3 main meals, 3 snacks. Top right: lunch and dinner with 2 snacks, skipping breakfast. Bottom left: breakfast and dinner with 2 snacks, skipping lunch. Bottom middle: almost fasting (sick day). Bottom right: unusually large meal (holiday/wedding).

plot (lower right); in this case, a schedule was manually crafted to simulate an unusually large amount of CHO consumption (more than double the mean CHO amount planned for lunch), which has an extremely low probability of being generated by the algorithm. The failure to control an occasion like this adequately is understandable since the agent never encountered anything like it during training.

Observing the episodes in Figure 5.3, it's noticeable that policies tend to be rather cautious, staying for most of the time in the upper range of normoglycemia, above the reference. Due to the LQL reward function shape, the agent has no real short-term advantage in getting closer to the more dangerous region of hypoglycemia. However, this is probably something we want to encourage, considering that staying in the lower range of normoglycemia in between meals may allow the agent to have more room to react to them, decreasing the height of postprandial peaks and consequently the time in hyperglycemia. Agents trained with the same data but using the LQQL-based reward function, which favors this behavior, as explained in Section 4.1.3, indeed confirm that performances can be boosted, increasing time spent in the target range by about 3%, up to 68.5%. In Figure 5.4 an example of how the policies, trained using different reward

functions, tend to work: the first agent (LQL-based) stays most of the time in the range from 120 to 150 mg dL^{-1} in between meals, while the second (LQQL-based) tends to stay lower, close to 100 mg dL^{-1}, when possible.

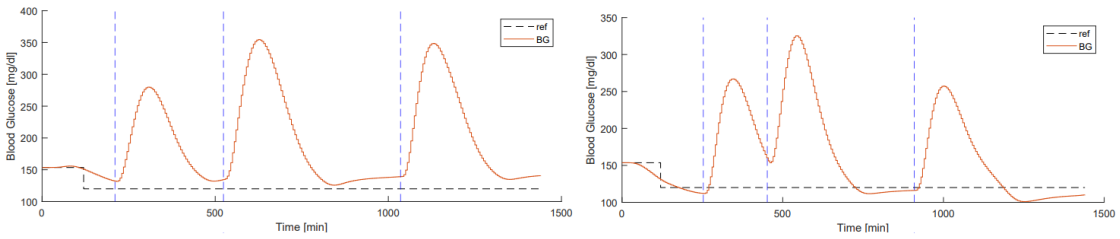

Figure 5.4: Comparison of two episodes, using two different agents: the one on the left was trained using LQL-based reward function, while the one on the right using LQQL-based function, encouraging to settle in the lower range of normoglycemia.

The robustness of the method was, finally, put to the test by considering meal sizes with larger variability (35% of the mean CHO amounts, against the previously considered 15%), which replicates what is seen in the general population of T1D patients[77]. As expected, increased variability affects, although not dramatically, the performance achievable by the approach, since the agent tends to become more cautious about strong responses to disturbances, spending more time in hyperglycemia. Furthermore, being the task harder, the length of the training process increases drastically, almost doubling. Specifics are reported in Table 5.3.

Meals variability	Cumulative reward	Ep. to converge	% Time hyperglycemia	% Time hypoglycemia	% TIR	HBGI	LBGI
15%	600	3300	30.2	1.3	68.5	8.3	0.6
35%	200	5800	34.1	0.9	65.0	9.9	0.3

Table 5.3: Comparison of performances obtained changing solely meal variability.

Since one of the purposes of this thesis is to test RL methods for AP in realistic conditions, this kind of variability in the meal schedule is used for all the following experiments presented.

5.1.3 Episodic failures and CHO consumption in hypoglycemia

An aspect which is necessary to consider when dealing with such a delicate task as diabetes management is the study of worst-case scenarios. The metrics implemented, in the aggregate, can give a good idea about the general performance of a policy, but a deeper look into problematic episodes can help considerably to evaluate how safely the agent is actually acting. In the literature, different studies have discordant definitions of episodic failure in the case of diabetes

management, setting different thresholds in deep hypoglycemia to distinguish these traumatic events. Since a proper consensus about this isn't reached, the entire distribution of the minimums reached in 1000 episodes after policy convergence is discussed here.

The latest policy discussed spends less than 1% of the time in hypoglycemia on average, with a mean LBGI of 0.3, which indicates a minimal risk. The distribution of the minimums is shown in the left plot of Figure 5.5: most of them are actually in normoglycemia, clustering around the target value of 100 $\mathrm{mg\,dL^{-1}}$, corresponding to the maximum designed in the LQQL reward function, but sometimes, 6% of the episodes, the agent drops into hypoglycemia. When it does, only the upper range of hyperglycemia is reached usually, with the exception of some rare events (less than 1.5% of the episodes), when it drops even below 30 $\mathrm{mg\,dL^{-1}}$. The agent never achieves a minimal blood glucose level below 5 $\mathrm{mg\,dL^{-1}}$, which some studies, possibly optimistically, indicate as the threshold for catastrophic episode failure[62].

It should be noted that, so far, in the training and evaluation of the agent, the simulation process doesn't take into account consuming CHO in reaction to low blood glucose levels, which is the most common strategy in real life to avoid catastrophic failures in BG control. This aspect was briefly taken into consideration, to understand its impact on policy evaluation and policy training: a system was implemented such that, if active, whenever the agent dropped below the hypoglycemia threshold of 70 $\mathrm{mg\,dL^{-1}}$, the agent would immediately consume a fixed amount of 15g of CHO, with the possibility to repeat it once per hour.

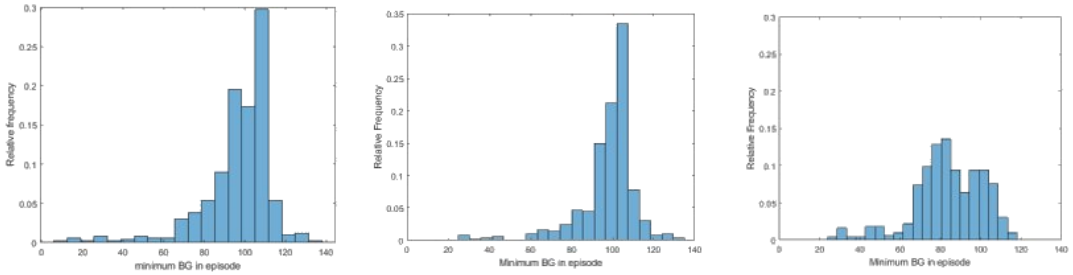

Figure 5.5: Comparison of the distributions of minimums, obtained switching on and off the system for the consumption of CHO snacks of 15g in response to hypoglycemia. On the left: policy trained and evaluated without snacks. In the middle: policy trained with system off but evaluated with system on. On the right: policy trained and evaluated with CHO consumption in hypoglycemia.

This system was used to evaluate two policies: one trained without it and one trained with it. The distributions of the minimums obtained are presented in Figure 5.5, respectively in the middle plot and the right plot, along with the comparison to the standard case, when the system is switched off both in training and evaluation. Clearly, training an agent with this

system active, results in lower and more frequent minimums, showing that, if not penalized, the method tends to abuse the safety net put in place. However, when comparing the distributions relative to the evaluations of the agent trained without the safety net, we notice that in realistic conditions (i.e. consuming small CHO snacks), the policy never drops to critical blood glucose levels and, in general, hypoglycemia events tend to be a lot milder than previously observed, when snacks aren't considered. This system, while being useful to make proper considerations about the actual worst-case scenarios when evaluating a policy, is never activated in the training of all the following agents to avoid its possible abuse as a safety net.

5.1.4 Additional features: the second derivative and COB

The agents trained so far rely exclusively on 3 features (BG concentration, BG rate, and IOB) to represent the state, which allows to maintain the dimension of the Q-values table small enough (about 5 thousand entries). Adding an additional feature, discretized into a limited number of ranges (5 to 10), greatly expands the table but is still doable, at the cost of speed in the training process.

Two features were tested in this context: the second derivative of blood glucose concentration and carbohydrates on board (COB; see Section 4.1.1 for specifics about the calculation). Although they're quite different features conceptually, the idea behind their use is similar. The amount of time spent in hyperglycemia after meals clearly depends on how fast an agent is able to react to them, delivering a proper amount of insulin as soon as they're detected. The features used so far don't change drastically in response to meals and the response of the agent tends to be delayed, especially considering that the agent has to discriminate between a small disturbance, like a snack, and a big disturbance, like a main meal, before delivering large amounts of insulin. A feature that is activated by meal occurrences and whose intensity is proportional to the amount of CHO consumed, on the other hand, has the potential to significantly reduce the subsequent postprandial peaks.

The second derivative of the blood glucose concentration changes rapidly when BG suddenly rises or drops, as opposed to the first derivative, considered so far, which takes some time to build up. The rate is still an essential feature to consider, considering that it provides the state representation a way to discriminate at what speed, and in which direction, the feature to control (BG) is heading; however, the second derivative, i.e. the instantaneous acceleration of the process, provides a way to react faster. An example of the difference between the two features is shown in Figure 5.6, showcasing an episode where all possible meals are consumed;

note the sharp and immediate increases of the second derivative signal in response both to main meals and snacks, as opposed to the smoother and delayed changes in the rate.

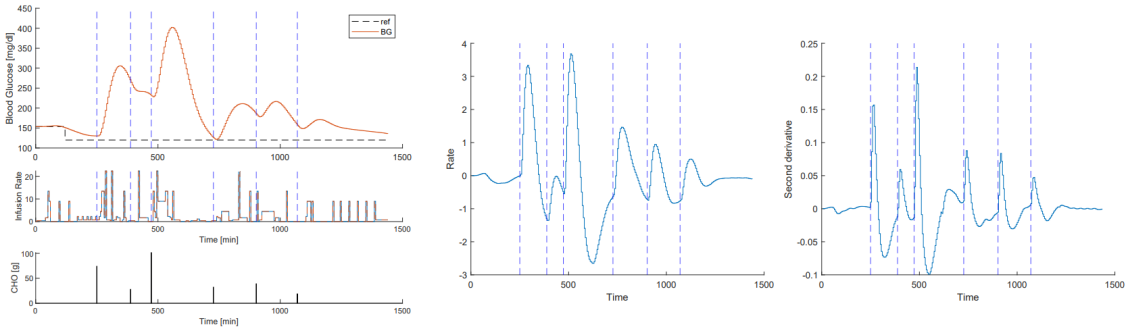

Figure 5.6: Difference between the first (middle plot) and second derivative (right plot) features, calculated with respect to the BG profile of the episode on the left. Note the difference in reactivity between the two with respect to the meals (the vertical dashed lines).

The difference in final performance and in reactivity is substantial: the agent trained with the additional feature (discretized into 8 ranges) spends way less time in hyperglycemia and reduces the mean intensity of hyperglycemic events overall. Performance metrics and comparison with the simpler agent using only BG concentration, BG rate, and IOB are reported in Table 5.4. This improvement comes, of course, at the price of training speed, reaching convergence in 4 times more episodes than in the previous case. Figure 5.7 compares the handling of episodes with identical meal schedules by the two agents: looking at the insulin profiles, it's clear that the increased reactivity to meals, by the agent making use of the second derivative, is responsible for the decrease in height and width of postprandial peaks.

State representation	Ep. to converge	Cumulative reward	% time hyper	% time hypo	% TIR	HBGI	LBGI
BG,rate,IOB	5800	600	34.1	0.9	65.0	9.9	0.3
BG,rate,IOB,2nd der.	24000	1300	27.2	0.3	72.5	8.2	0.2

Table 5.4: Comparison of performances obtained adding the second derivative feature.

It should be noted, however, that the second derivative is, possibly, taking advantage of approximations introduced in the simulation and, arguably, in real conditions would fail to discriminate adequately between meals and other disturbances induced by factors which are not taken into consideration, such as physical activity or even typical daily alterations, such as the notorious dawn phenomenon, i.e. the rise in blood sugar levels in the early morning hours, when hormones induce the liver to release large amounts of sugar into the bloodstream. In

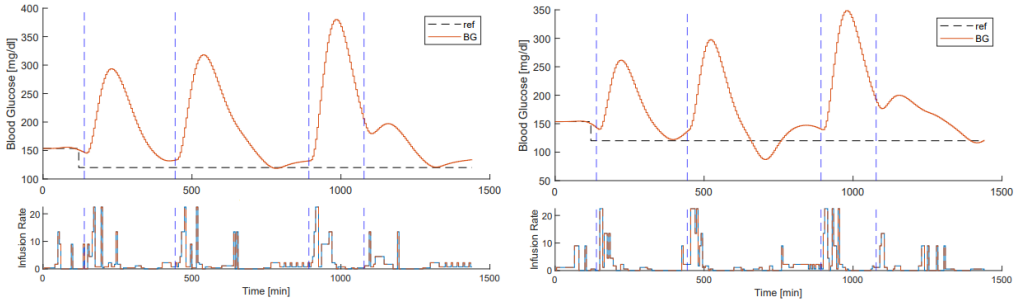

Figure 5.7: Example of two identical episodes, handled differently by the policy without second derivative feature (on the left) and with (on the right). Note in the right insulin profile the faster response to meals.

other words, there's a serious risk of overfitting the state representation to the simulator, obtaining results that are not so easily transferable to real conditions.

On the other hand, the implementation of the carbohydrates on board (COB) feature, although it shares the same intention of building a representation which allows for faster reactions by the agent, does not raise the same concerns of overfitting to the simulator, since the signal is not derived from a manipulation of the BG concentration feature, but it's explicitly delivered by the user. Clearly, the fact that the user has to interact with the system means that the approach is trading off autonomy for increased performance. Although the general focus of this work is on fully closed loop systems, this case is briefly taken into consideration, in order to understand the impact that such a feature would have, all else unchanged.

Several variations were tested, depending on the level of accuracy of the information provided by the user. The COB feature can be calculated by considering the real meal schedules, assuming perfect accuracy in CHO counting by the user, but, realistically, it has been shown that it's rather common for diabetes patients to miscalculate the amount of CHO consumed. Hence an agent was also trained by considering COB calculated on meal schedules deviating from the real ones, following a gaussian error distribution which emulates the estimates provided in A.S. Brazeau et al.[77]. Finally, instead of calculating an estimate of the carbohydrates which still have to be absorbed at every step of the episode, simple meal announcements can be considered: instantaneous signals whose intensity is proportional to the amount of CHO in a meal for a single step when the meal is consumed, and zero otherwise, as opposed to the prolonged signal provided by COB. The results obtained for all the cases mentioned are presented in Table 5.5, along with the usual comparison with the simpler state representation.

Similarly to what was observed with the introduction of the BG second derivative as a new feature, using COB or meal announcements leads to improved reactivity, reducing the height

State representation	Ep. to converge	Cumulative reward	% Time hyper	% Time hypo	% TIR	HBGI	LBGI
BG,rate,IOB	5800	600	34.1	0.9	65.0	9.9	0.3
BG,rate,IOB,COB exact	11000	1100	28.0	0.8	71.2	8.3	0.4
BG,rate,IOB,COB with errors	11500	800	29.1	1.0	69.9	8.6	0.9
BG,rate,IOB,meal ann. exact	11500	900	28.5	1.3	70.2	7.6	1.1
BG,rate,IOB,meal ann. with errors	10000	700	30.7	0.9	68.4	8.2	0.7

Table 5.5: Results obtained adding information about CHO consumption as a feature, as COB or meal announcements, for cases with and without errors in CHO calculation by the user.

of the peaks after meals by about 20-30 $\mathrm{mg\,dL^{-1}}$ on average, and hence the amount of time spent in hyperglycemia. Once again, besides the autonomy of the system, the price to pay is in terms of the number of episodes before convergence, twice as much as the simpler representation. The best performances are obtained when perfect information about the amount of CHO consumed is available, but, especially considering that they get discretized into ranges, even imprecise estimations can still have a rather positive effect, which is in line with what was observed in the literature for other control methods [86]. Keeping track of an estimation of the CHO that have yet to be absorbed, as expected, is slightly better than considering solely meal announcements: in Figure 5.8 it's easy to see that when the COB feature is used (middle plot) the reactions to meals span multiple close steps, while, when meals are only announced (right plot), a single step reaction is immediately visible, followed by further adjustments after the meal detection is confirmed by the other features.

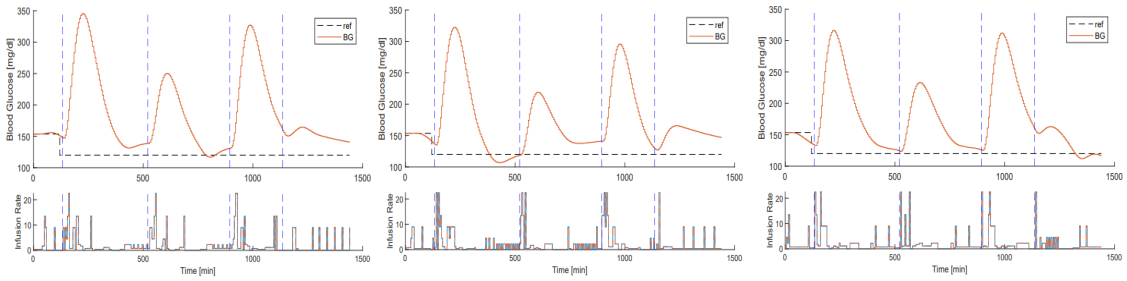

Figure 5.8: Example of the differences in handling the same meal schedule by the policy obtained with the simple BG, rate, and IOB state representation (left plot), with the additional information given by COB (middle plot), or with meal announcements (right plot). The difference in reaction speed is clearly visible by looking at insulin profiles.

Essentially, if COB is available, the agent is able to compare COB and IOB, to determine, over multiple steps, if more insulin has to be administered, while, using meal announcements, large meals are met by a single large injection of insulin followed by standard control. Overall, the gap in % TIR (time in range) between approaches with direct access to information about

the CHO in the system and approaches without is about 4-6%, which is pretty much in line with the average gap observed in similar studies[62].

5.1.5 Explainability: table interpretation

Systems that take into account only a few features have an advantage that has not been discussed yet: they are interpretable, if properly inspected. This is possible thanks to the use of visualization techniques, taking advantage of the fact that the Q-values table is small in size. Rather than considering the action values individually, which are not very informative, it is better to focus on the state values, which indicate the expected return given the observed state, and the policy, i.e. how the agent acts, being in that state. For reference, the relationships between the Q-values, the greedy policy, and the state values are reported below:

$$\pi(a|s) = P[A_t = a|S_t = s] \qquad \pi(a|s) = \begin{cases} 1 & \text{if } a = \text{argmax}_{a \in A}\, q(s,a) \\ 0 & \text{otherwise} \end{cases}$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$

When exploration is not actively pursued, as it is the case so far, since we rely only on the stochasticity of the system to explore enough, the policy deterministically follows the action associated with the largest estimated Q-value of the current state, hence the state value coincides with the maximum action value.

In this section we consider the simple state representation using BG concentration, BG rate, and IOB, to facilitate visualization and intuition. Since we're dealing with three features, multiple tables have to be reported at the same time to understand the system in its entirety. Between tables the IOB range changes, while within a table it's kept fixed and all combinations of BG concentration and BG rate are compared. The tables are plotted as heatmaps, where the color expresses either the expected return, when state values are considered, or the action that the agent is going to take, when the policy is considered. Furthermore, to understand if that state is actually ever reached by the policy to which the system has converged, in every square the average number of visits per episode (calculated over 1000 episodes) is reported. As a reminder, the episodes are 24 hours long and CGM readings are reported every 5 minutes; consequently, every episode lasts 288 steps. States which are never encountered during both training and evaluation are reported as blank: note that these states are usually impossible states due to their

contradicting features (e.g. an extremely large negative BG rate in deep hyperglycemia and a low IOB), hence their Q-values have never been updated with respect to their initialization.

Visualizing state values allows one to gain an intuition about how ”happy” an agent is to be in that state, but no information is present about its intentions, in terms of actions. Figure 5.9 shows the state values for every possible state: the brighter the color in the square, the bigger the return that the agent expects from that state. We can clearly see that the states relative to the first 3 ranges of insulin on board are actually transitory states. Upon further inspection of IOB during several episodes, the explanation for this kind of behavior is simple: these states are encountered only at the beginning of each episode, because the agent wants to maintain a larger background level of IOB throughout the episode, similarly to what is done with slow-acting insulin in manual control. As a consequence, the first three IOB ranges could probably be incorporated into one, with no loss of performance. Furthermore, an AP system is continuously

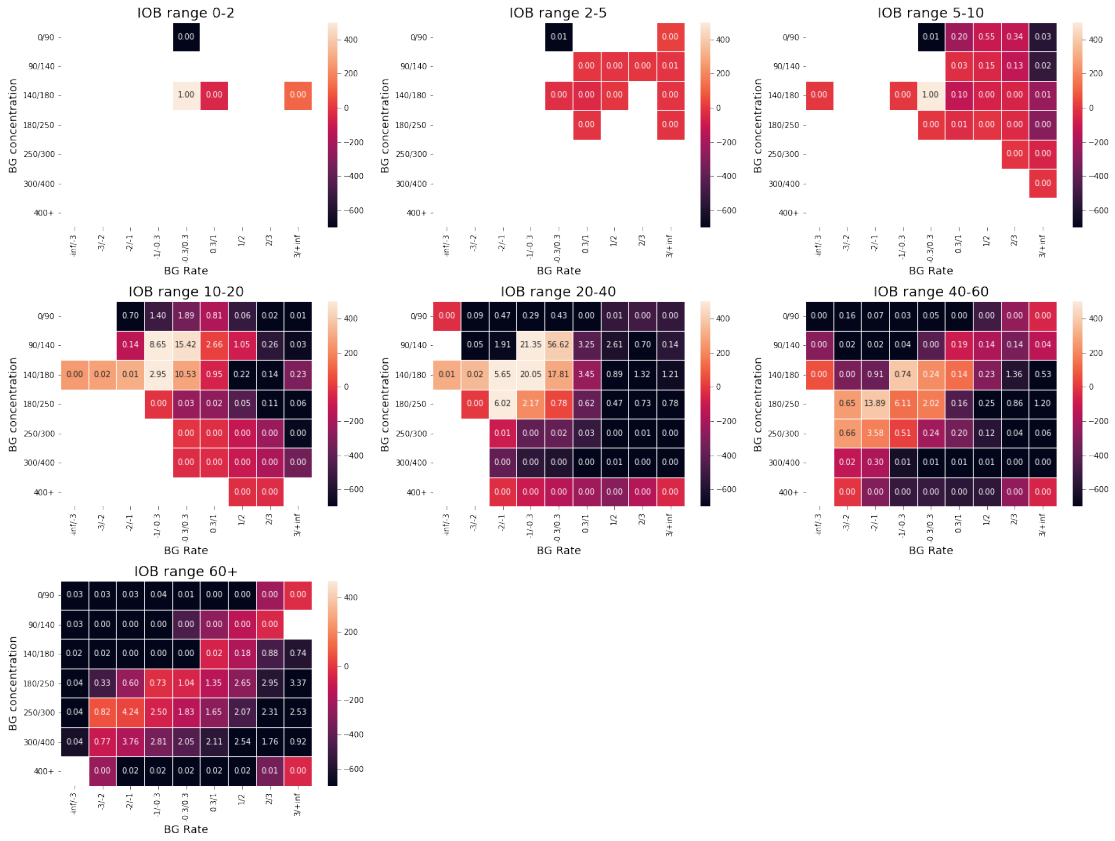

Figure 5.9: State values for every state which an agent with features BG concentration, BG rate, and IOB can encounter. Every heatmap is associated to a different IOB range, reported as the title of the table. The average number of visits per episode is outlined within every square; unencountered states are shown as blank.

working, so extremely low levels of IOB would probably hardly ever be encountered, as opposed to the episodic case when every episode starts with no insulin in the system. The fourth and fifth heatmaps are more informative: most of the time in the episodes is spent in states relative to euglycemia, neutral or slightly negative rate, and moderate levels of IOB, which are exactly the desirable states and in which the agent, rightly, expects greater rewards, as shown by the brighter color associated with these states, i.e. the large state values. Finally, looking at the heatmaps relative to the last three IOB ranges, it's interesting to see that the system is recognising that large positive rates, especially when they're not countered by large amounts of insulin, are usually resulting in bad returns and that large negative returns can also be expected when large negative rates are observed (there's too much insulin in the system and it's headed into deep hypoglycemia). This is especially clear when observing and reasoning about the large diagonals that cross both of the last two heatmaps: positive returns are only expected when the current BG concentration, rate, and IOB position the system for a rapid return to normoglycemia without overshooting into hypoglycemia. Finally, note that the little time that is spent in hypoglycemia is mostly associated with low or medium IOB amounts and slightly negative or neutral BG rates, meaning the agent indeed gave too much insulin but not so much to still be falling into deep hypoglycemia.

Unlike state values, the display of heatmaps relative to the policy does not give an idea about how good a state is, but only about what the agent intends to do. Figure 5.10 shows the actions which would be taken by the agent from every possible state: in this case, the brighter the color, the larger the dose of insulin to administer (check the colorbar in the figure, ten actions of varying intensities, ranging from 0 to 20 times the patient's BR, are available to the agent). As a reminder, the system is forced to avoid any further insulin administration in the first BG concentration range (0 to 90 $\mathrm{mg\,dL^{-1}}$), so the policy followed in that range is trivial. Once again, the tables relative to the first three IOB ranges only reveal that these states are transitory; the few times they're met, the agent simply chooses to give enough insulin to reach regions of the state space with larger IOB. The following three heatmaps show that as soon as the system is in hyperglycemia and blood glucose concentration is rising (or it's going down slowly but we're too deep in hyperglycemia), the agent has to react by increasing sharply the insulin infusion rate, in order to quickly bring back BG concentration within euglycemia. Note, however, that in euglycemia, unless large BG rates are observed, the agent often tends to still be cautious and not deliver much insulin; this type of behavior is why we often observe a delayed response to meals. This is likely due to the fact that the system, without any direct information about the amount of CHO consumed, has to understand if it's possible to control the disturbance

through weak actions (i.e. a snack) or not (i.e. a main meal). In the last heatmap, relative to the largest IOB range, observe that, since there's already a lot of insulin in the system, the agent tends to be aggressive solely in the case that BG rate is still quite large, otherwise it's mostly choosing to administer doses close to BR, waiting for the return to normality, thanks to its previous actions.

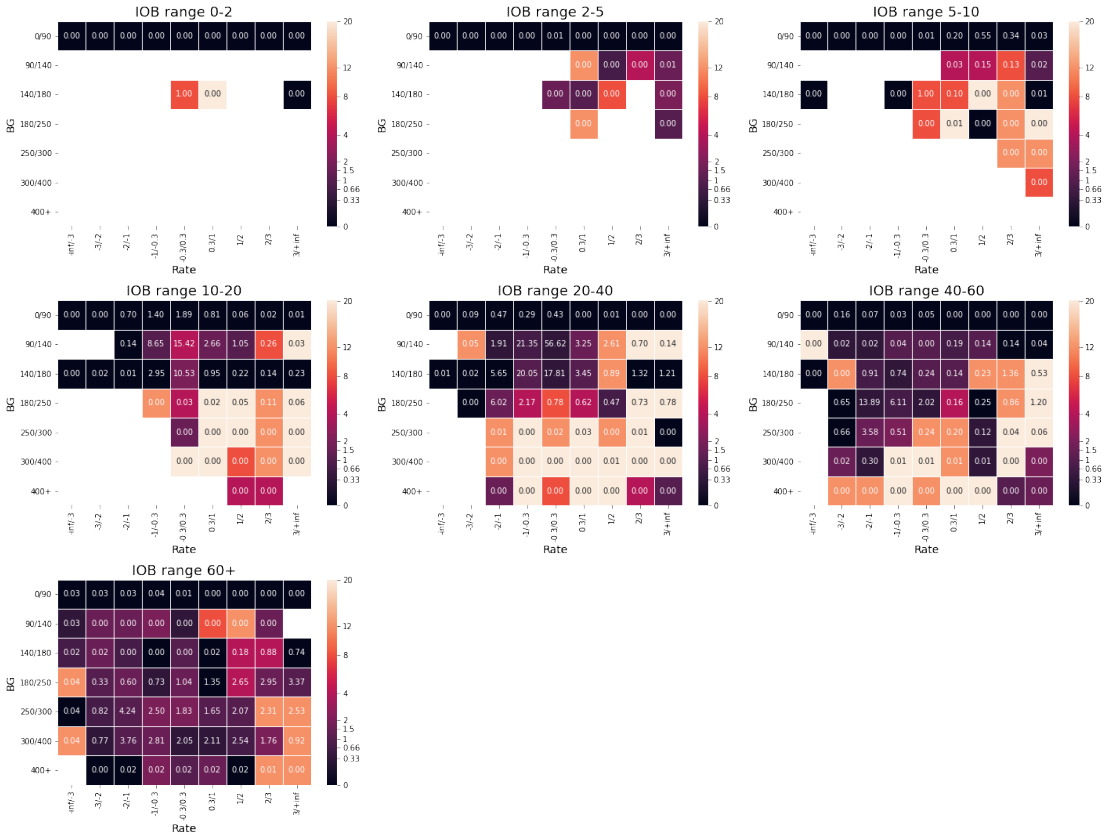

Figure 5.10: Actions the agent would take for every state, following the policy. Every heatmap is associated to a different IOB range, reported as the title of the subplots.

Even though, for the most part, the actions taken by the agent are correct and intuitive, in some cases it's possible to spot some evaluations that are simply wrong and, quite possibly, dangerous. This is due to the fact that in tabular approaches there's no built-in generalization and states which are rarely encountered can be associated with very wrong Q-values estimations. As long as the states are not visited, the effect of these wrong estimations is not noticeable. An example of this is found in the second to last heatmap, for the two states on the left, relative to euglycemia and large negative rates: in such a situation, the appropriate actions to take would

be either to not deliver insulin at all or only close to the patient's BR; however, the current estimates suggest administering large amounts of insulin, which would inevitably lead to hypoglycemia. In this case, the average number of visits per episode is zero, indicating that no dangerous hypoglycemic event was caused by them; however, they could still be potentially harmful. This has been the case for another wrong estimation, which is noticeable in the previous IOB range, relative to the lower range of euglycemia (90 to 140 $\mathrm{mg\,dL^{-1}}$) and a negative rate between -3 and -2 $\mathrm{mg\,dL^{-1}\,min^{-1}}$: this state was visited within the 1000 episodes of the evaluation process and the action taken was likely responsible for one of the rare, dangerous dives into hypoglycemia, which, realistically, would have to be manually corrected by the user, consuming CHO.

Bad Q-values estimations associated with rare states wouldn't likely happen if the method kept into account the average behavior in close states. Following this reasoning, an initial idea, could be to smooth out the tables reported, using something like a 3D gaussian filter, avoiding sudden changes in policy between close states. Unfortunately, smoothing doesn't work, since it does achieve generalization but at the cost of the ability to discriminate, which is essential in such a task. For example, the sudden change in BG rate from neutral to slightly positive warrants a drastic increase in insulin rate, in the case of main meals. Some function approximation methods, described in Sections 4.2.2 and 4.2.3, provide exactly the proper framework to satisfy such requirements.

5.2 Linear Function approximation RL approaches

5.2.1 State representations with tile coding

The approaches which are discussed here have all been implemented using the tiles3 library[87], through calls to Python functions from MATLAB®, making sure consistency of the encoding of present features between calls is observed, through pseudo hashing.

The same features that have been considered in the tabular case (BG concentration, BG rate, and IOB) are employed here, without the need for discretization. The difference is that now they're not used directly but only to find the "active" components of the feature vectors, which, combined with the weight vector, allow to calculate the approximate action values. The present features returned depend on how the state-action space is partitioned; as mentioned, linear function approximation methods' success is largely based on whether or not the states are properly represented in terms of the features; hence feature crafting is quite an important step.

In the simplest case, the state-action space is partitioned using a unique set of 4D tilings, spanning all dimensions of the problem: generalization happens taking into consideration simultaneously all variables; clearly, due to the dimensionality of the grid, provided enough tilings and tiles per tiling are specified, this system is able to discriminate meticulously. Using this kind of representation leads to good policies but takes a lot of episodes to converge (about 13000), struggling in performance early on in the training. Two examples of episodes, using the final policy reached, are shown in Figure 5.11. From the insulin profiles, the greater generalization, compared to the tabular case, can already be noticed: throughout the episodes, the infusion rate tends to remain close to the patient's basal rate, with the exception of the sudden reactions to meals, as soon as they're detected. Arguably, this is also the most "human-like" policy observed so far, in terms of interpretability (which will be thoroughly discussed in Section 5.2.2) and similarity to the standard diabetes management approaches.

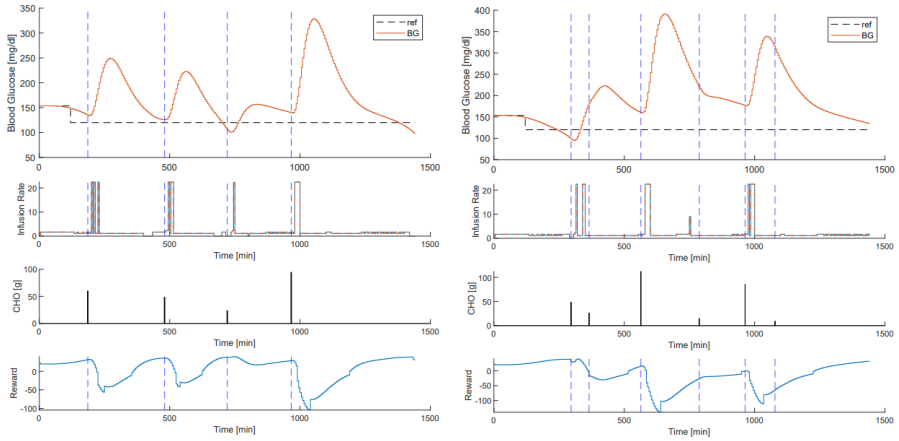

Figure 5.11: Example of the handling of two episodes by the agent employing the 4D tile coding representation.

The problem with this implementation is that there are a lot of possible combinations considering all four variables at the same time and it takes a lot of data to form an appropriate idea of the whole problem. Conversely, an implementation which partitions the state-action space by considering each feature separately, hence employing exclusively 1D tilings, is able to learn some basic notions of the problem fast (e.g. large negative/positive BG rates are usually a bad sign), but can't appropriately evaluate states and policies long-term because it totally lacks the representational power to consider the overall context, which is absolutely necessary for this task (e.g. a large negative BG rate can actually be a good thing, in the case of hyperglycemia). It can be quite interesting to consider how mono-dimensional contributions vary, to gain an intuition about what the agent is able to learn with such a representation (Figure 5.12).

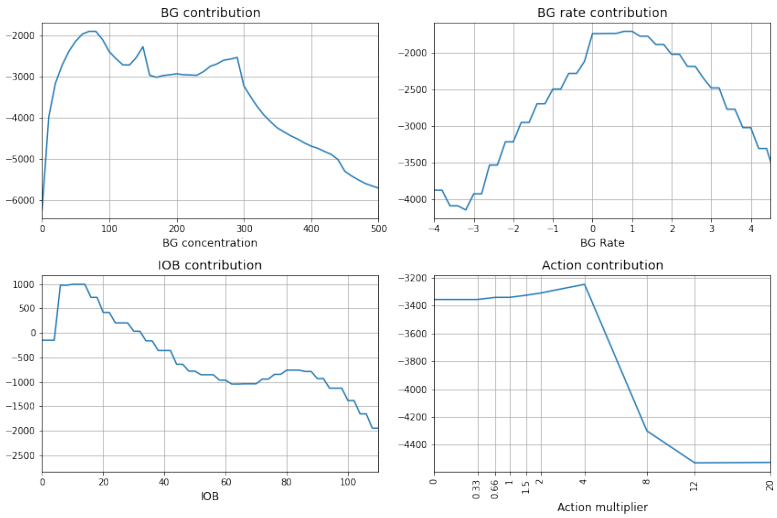

Figure 5.12: Mono-dimensional contributions to the computation of action-values, relative to each feature considered.

The shape of the contribution curve associated with the BG concentration feature closely resembles, with the exception of some bumps going into hyperglycemia, the LQQL base reward function, slightly shifted to the left: clearly, the agent has a good intuition of the fact that better rewards are to be expected close to normoglycemia, while extreme levels of BG lead to bad returns, particularly and shiftily so in hypoglycemia. The contributions of the BG rate are quite clear: the agent expects to receive the most reward when the rate is close to neutral, while extremes usually lead to trouble. The IOB contribution curve underlines that too much insulin is usually correlated with bad returns, but a decent amount of background IOB is certainly better than none, which is in line with the behavior observed for some tabular policies. Finally, the actions' contributions show that, in general, vigorous actions can have harmful effects, while actions close to the basal rate or slightly larger, can be beneficial. Agents trained employing this kind of representation, after an initial learning phase, get quickly stuck into delivering solely the "best average action" (in this case four times the patient's BR). Keep in mind, all of these considerations should be thought in a mono-dimensional sense; strong actions are indeed necessary, but in the right context, which is exactly what a system based exclusively on 1D tilings is lacking. These notions are undoubtedly rough and simplistic, but they may provide some useful initial domain knowledge for more complex methods.

Ideally, when designing a system with tile coding, both early generalization and long-term discrimination are desirable: it's only logical then, to consider implementations which divide the state-action space using both 4D tilings and 1D tilings. Intuitively, early training is mainly

driven by learning mono-dimensional concepts (and somewhat by 4D tilings, but to a lesser extent), which allow for a good starting point, while long-term refinements are due exclusively to 4D contributions. This type of representation is superior to those presented before, achieving a quite rapid improvement in performance in the initial training phases, and asymptotic performance close to tabular methods and to the implementation using exclusively 4D tilings, but requiring a lot fewer episodes for convergence. Figure 5.13 shows a comparison of the training processes relative to the implementations discussed so far. Due to the large inter-episode variability, moving averages are plotted to obtain proper visualizations.

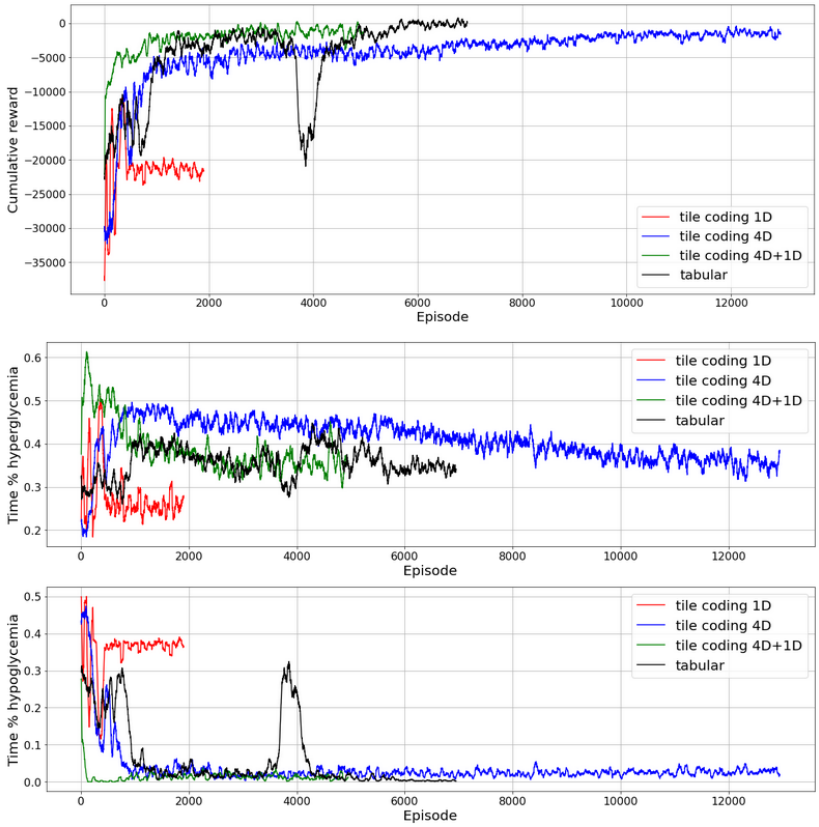

Figure 5.13: Comparison of the training process for the tile coding implementations described. Using both 1D and 4D tilings (in green) allows faster learning.

Note how the implementation that employs only 1D tilings, after the initial improvements, gets irremediably stuck, while the implementation using exclusively 4D tilings takes a long time to reach a performance close to the tabular method. Employing both kinds of tilings, instead, leads to extremely fast learning during the first couple of hundred episodes, and a gradual improvement afterward.

It's worth commenting more in detail and by itself on the training process of the implementation using both 1D and 4D tilings (shown in Figure 5.14), since it's going to be the main approach employed in all following experiments, when function approximation is considered.

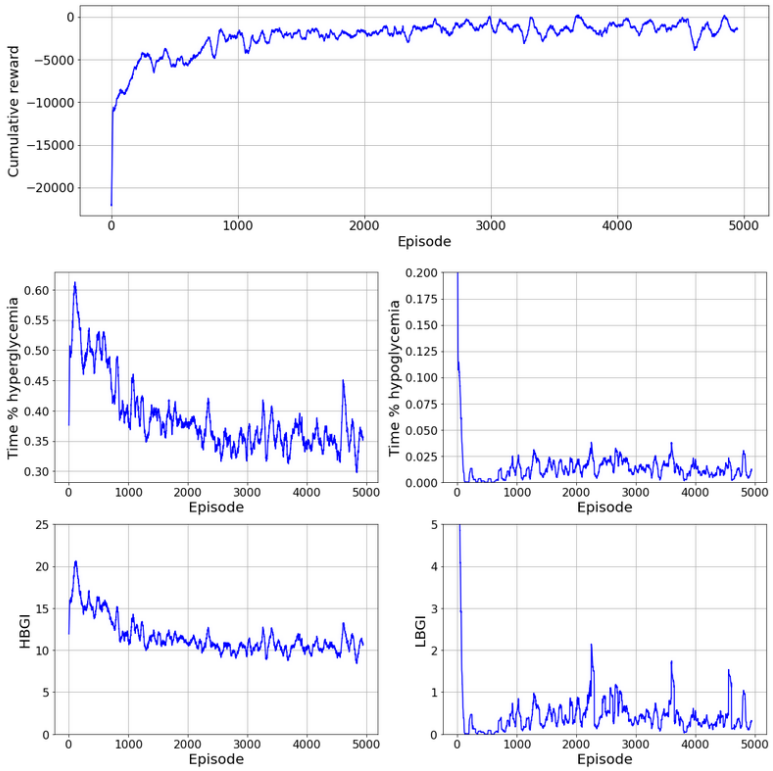

Figure 5.14: Training process of the method using both 1D and 4D tilings.

As mentioned, performance, in terms of cumulative reward, improves substantially in the first few episodes: but how is this achieved exactly? The answer relies on the fact that monodimensional contributions force the agent to become rather cautious in terms of insulin delivery, very fast. It takes only 40 episodes to drop below 10% of the time spent in hypoglycemia and only about 150 to reach 0%. From this point onward, the agent doesn't experience any particularly traumatic hypoglycemic event, as the LBGI metric implies, staying within a minimal to low-risk range. This is not to say that the agent has already reached a good policy at 150 episodes; it simply administers a little insulin to completely avoid hypoglycemia, at the cost of more time spent in hyperglycemia. Hyperglycemia is then gradually dealt with, mainly thanks to the contribution given by 4D tilings which, over time, are able to accurately discriminate in which instances vigorous actions are required. After about 2000 episodes (6 times less than the implementation considering only 4 tilings), the training process pretty much converges to

a good policy, which is close, in terms of performance, to the tabular approach previously proposed. The policy is extremely similar, as it's evident when inspecting insulin profiles, to the one reached with the 4D tilings implementation. An example of an episode is shown in Figure 5.15. In truth, the policy reached is pushing for even more uniformity in terms of actions; a proper inspection through visualization in the next section will underline the differences between the two cases, apparently so equal.

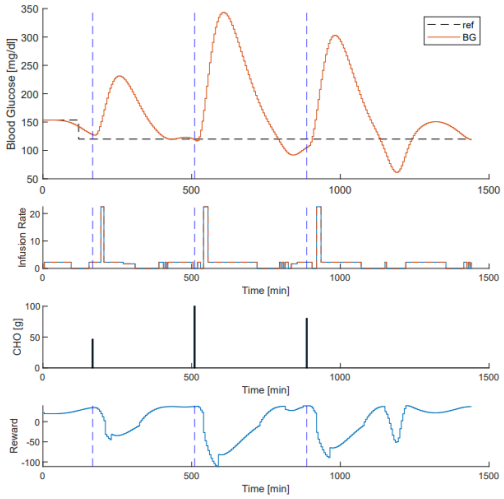

Figure 5.15: Example of the handling of an episode by the agent trained with state representation based on 1D and 4D tilings.

Some other variants of state representation through tile coding were tested and rapidly discarded. A very brief description of them is provided next.

Generalization across the action dimension is not something which is usually so obvious in tile coding for RL tasks: in many cases actions are treated as independent and are associated with completely different tiles, even when the state is the same. This is reasonable, because usually the actions available to an agent can be completely different, even opposite; however, in the context of an AP system, actions all refer to doses of insulin. A system using independent tilings associated to each action simply converges way more slowly and achieves a slightly worse performance, due to more frequent traumatic episodes.

Another idea is to introduce even more types of tilings to build the feature vector: since 1D tilings are highly general and 4D tilings are highly specific, possibly a middle ground, 2D tilings, can help speed up learning, describing the interactions between each pair (or selected pairs) of features. Unfortunately, this doesn't work: after a fast initial learning phase, typical of 1D contributions, it completely fails to make progress. Upon close inspection of the contributions

of each kind of tiling, the issue is that 4D tilings' contributions, which are supposed to dominate the others in some contexts, are basically "covered" by all other contributions, making it impossible to achieve appropriate discrimination in the policy.

5.2.2 Explainability and system evolution in tile coding

As it was done for the tabular methods, policies and state values can be inspected through proper visualization techniques, even in function approximation, due to the few features employed. Some important differences have to be specified. To begin, no discretization is required in function approximation, but it is still required to create heatmaps to some extent, albeit at much finer grain than in the tabular case, for BG concentration and rate. IOB, however, has to be fixed to a defined value for every heatmap: the middle values of the previously discretized ranges, employed in the tabular case, are chosen. We'll consider all ranges, even though we saw in Section 5.1.5 that the three initial IOB ranges could be incorporated into one; this allows for an easier visual comparison of the figures. Secondly, including the amount of time spent in each (s,a) is no longer possible, since it relied on the discretization of features in ranges. Finally, the notion of un-encountered (s,a) (both in training and evaluation), is much trickier: in the tabular case, it referred to the fact that the corresponding Q-value didn't undergo any changes with respect to its initialization; for function approximation, it was decided that a (s,a) pair is said to be un-encountered (hence appearing blank in the visualization) in the case that at least one of its present features is associated with a component of the weight vector which has never undergone any changes. The drawback of this decision is that, in some cases, due to generalization, states can suddenly be considered un-encountered and displayed as blank, when the corresponding best action changes, but at least one of the components of the Q-value has never actually been modified throughout training. Despite these limitations, it is still possible to get a good idea of the general functioning of the methods.

In this section we will analyze the state values and policies of both the method using only 4D tilings and the method using 4D tilings and 1D tilings. We start with the first one: Figure 5.16 shows the state values at convergence (after 13000 episodes). These plots must be compared to those equivalent for the tabular case, in Figure 5.9.

The two figures are quite similar, implying that the two methods arrive at similar conclusions about the returns obtainable throughout the state space. Note, for example, that the largest returns are predicted, rightly so, in normoglycemia, when rates are close to neutrality and IOB is moderate. In particular, as IOB grows, the brighter region drops down in the heatmaps,

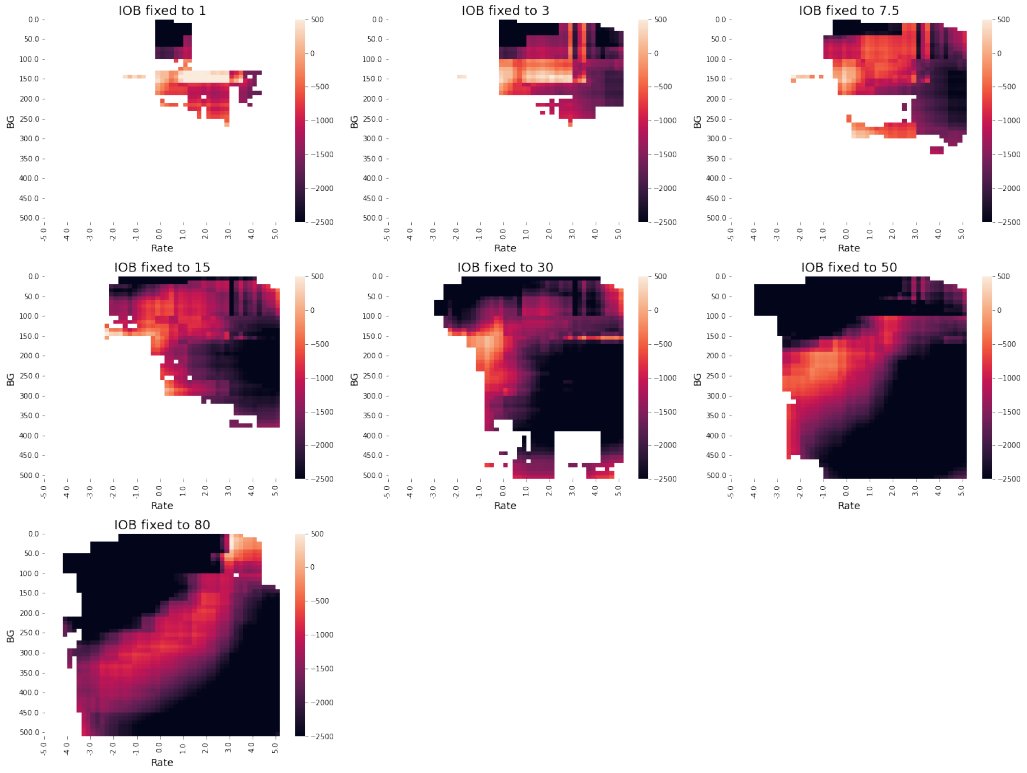

Figure 5.16: State values heatmaps, for the agent using 4D tilings for its state representation. Every heatmap is associated with a fixed IOB value. Un-encountered states, both in training and evaluation, are shown as blank.

implying that large amounts of insulin lead to positive rewards only when the agent is still in hyperglycemia and returning to the target range or the rate is still large and positive. Another similarity between the two figures is the presence of the large diagonals crossing the last two heatmaps, indicating that the agent has a clear understanding of the dangers of hypoglycemia and hyperglycemia and that it's important to appropriately balance out the different features, in order to return to euglycemia as shiftily as possible. Finally, even un-encountered states, which are mostly impossible states, are associated with identical regions of the state-space, for both methods. The main difference between the two cases is that when using tile coding, thanks to built-in generalization and a lack of discretization, the expected returns change a lot more smoothly, at least for well-explored areas of the state space. A similar argument about generalization can be made regarding the comparison of the policies reached, looking at Figure 5.17 and its tabular counterpart in Figure 5.10.

The first heatmaps, relative to low amounts of insulin in the system, are mostly transitory states encountered at the beginning of the episode or recovering from hypoglycemia, just like

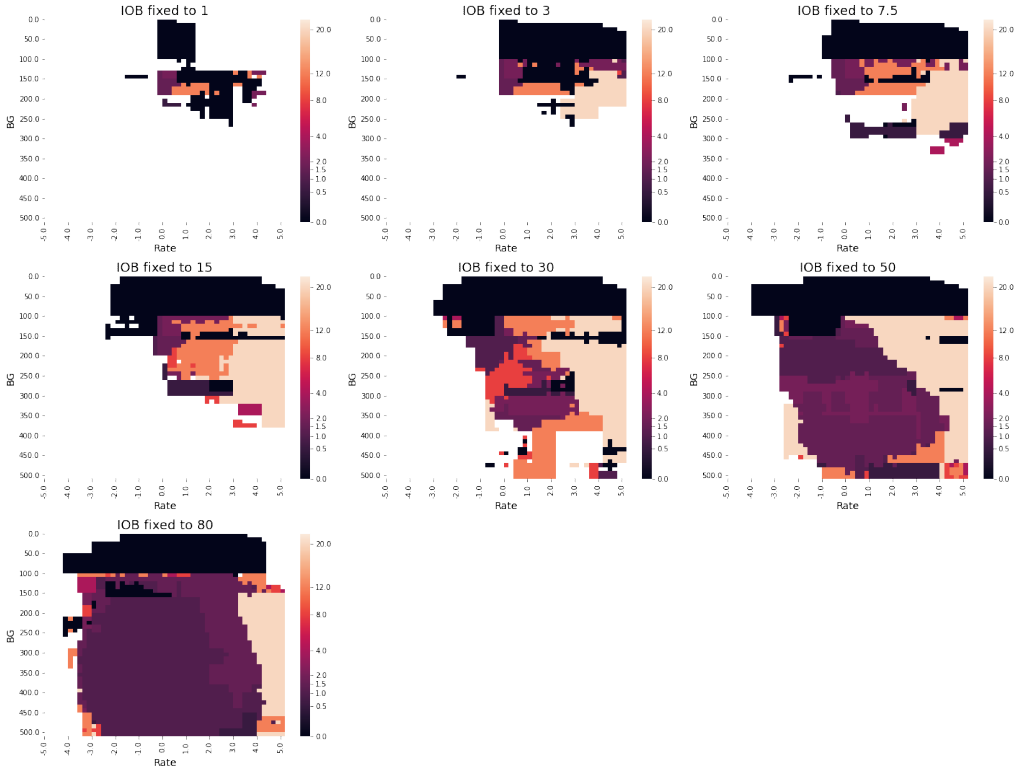

Figure 5.17: Actions the 4D tilings agent would take for every state, following its policy. Every heatmap is associated to a different IOB value, reported as title of each subplot.

for the tabular case. In the following heatmaps, for both methods, a sudden change in actions to be taken is visible, when the rate switches from neutral to positive, in order to react to meals; both policies then get more and more conservative about employing vigorous actions as IOB increases, unless the rate is still quite large. While the similarities are many, note that when tile coding is employed there's a lot more consistency in the action to be taken, between close states. These continuous large regions (especially in the last two heatmaps) are responsible for the more uniform (compared to the tabular case) and human-like insulin profiles observed in the example episodes. This is exactly the kind of result we were looking for, going from tabular methods to function approximation. Note, however, that the policy is still quite variable, especially close to the borders of un-explored regions of the state-space, which may still be problematic in some rare cases.

Moving on to the case of tile coding with both 4D and 1D tilings, state values are reported in Figure 5.18 and the policy in Figure 5.19. Not much is to be added with respect to the state values, except that the agent expects overall slightly larger returns, as it's evident by the brighter

colors, with respect to the state values of the other method. This is in line with the marginally better performance obtained by the final policy, but in truth it can also be a product of a streak of well-handled episodes, which momentarily boosted the evaluations.

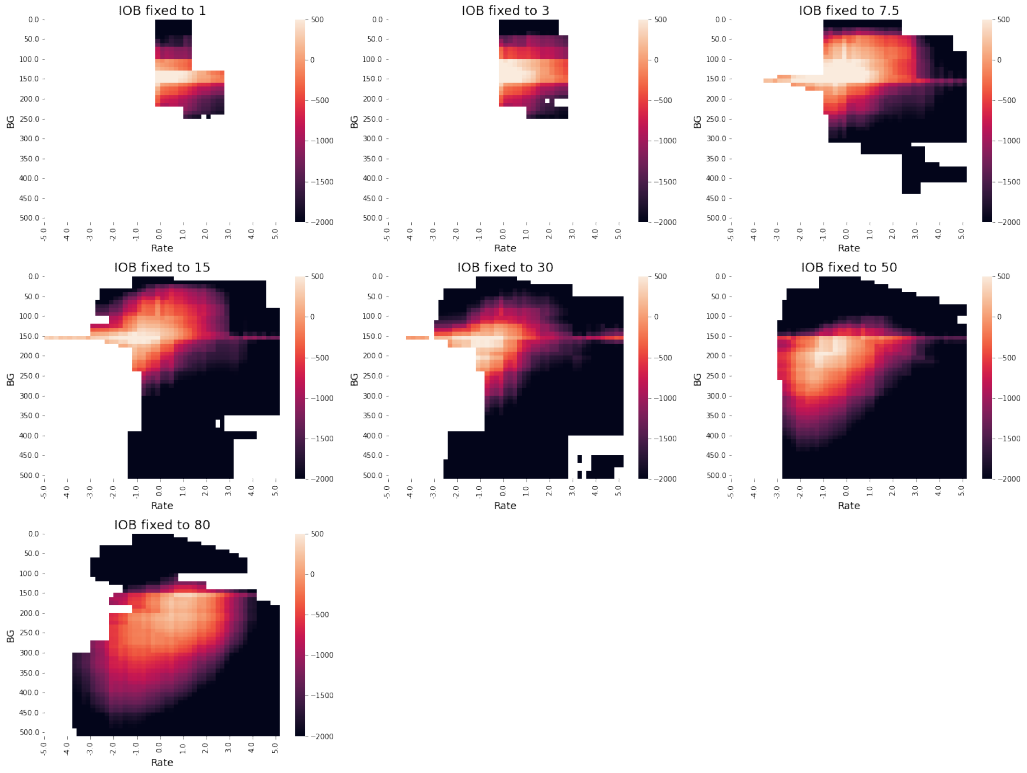

Figure 5.18: State values heatmaps, for the agent using both 4D and 1D tilings for its state representation. Every heatmap is associated with a fixed IOB value.

In the previous section, when looking at insulin profiles of example episodes, the policies obtained by using solely 4D tilings or both 4D and 1D tilings seemed almost equal. However, from Figure 5.19 it's quite clear that considering mono-dimensional generalization results in way more uniformity in the actions taken by the agent. The borders between the regions relative to strong and weak actions are positioned similarly to those of the policy previously discussed, but in this case there's no middle ground in terms of intensity of the actions employed. Basically only 3 actions are considered: no insulin at all, administering insulin close to the patient's BR or reacting as aggressively as possible to a disturbance. Moreover, unlike the previous policy, (s,a) in areas close to un-encountered regions, which are usually rare areas to visit under the current policy, tend to be associated with cautious actions, which bodes well for the safety of the policy overall.

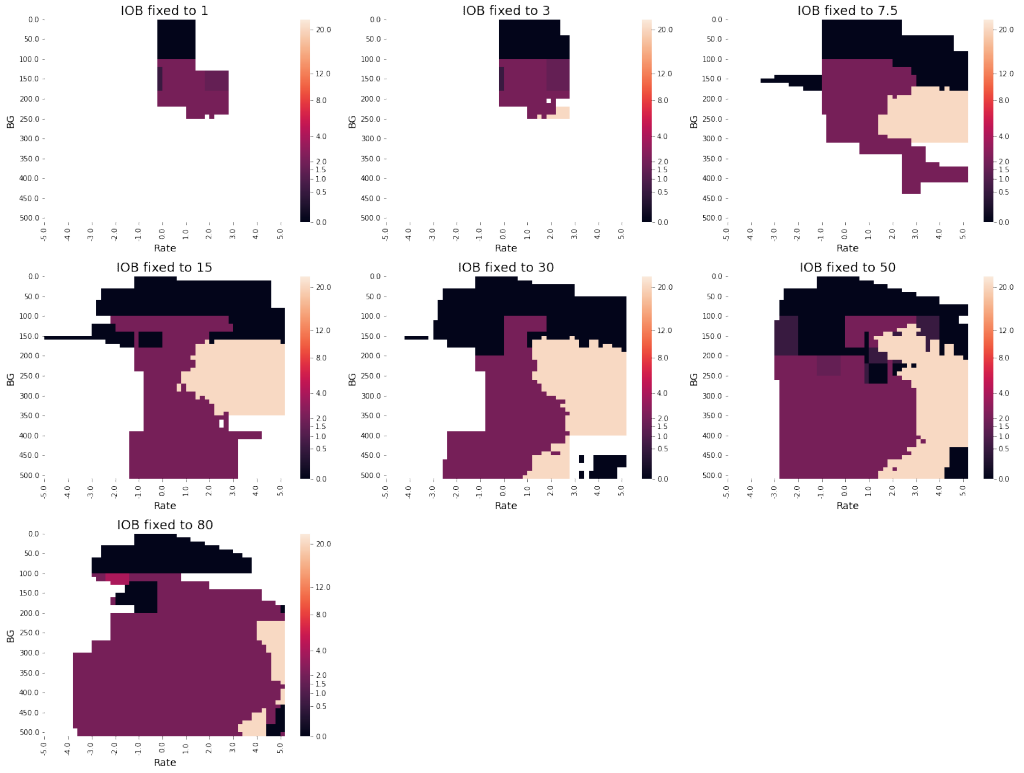

Figure 5.19: Visualization of the policy relative to the agent employing both 4D and 1D tilings.

The general uniformity described is to be attributed, mostly, to 1D generalization across actions and to how the policy is improved throughout the training process: this state representation initially forces the agent to be rather cautious about reacting aggressively, fearing hypoglycemia and considering strong actions usually dangerous. Only when the contributions associated with 4D tilings kick in, the agent is able to slowly understand in what context to use strong actions. This is what intuitively happens, but how can we be sure that the policy evolves this way, besides looking at the metrics of the training process and at the behavior and the contributions of the other methods? All we have to do is sequentially view the heatmaps associated with the actions the agent would take for each state, at different stages of the training process, not only after reaching a proper policy. This is done in Figure 5.20, inspecting the policy of an agent trained with the state representation based on 4D and 1D tilings, every 1000 episodes. Note that the agent is trained on a different seed; as a consequence, some minor differences are to be expected with respect to the policy previously inspected.

Looking at the initial policy, the early performance improvement is indeed due to the strong mono-dimensional generalizations that hypoglycemia has to be avoided and it's usually better

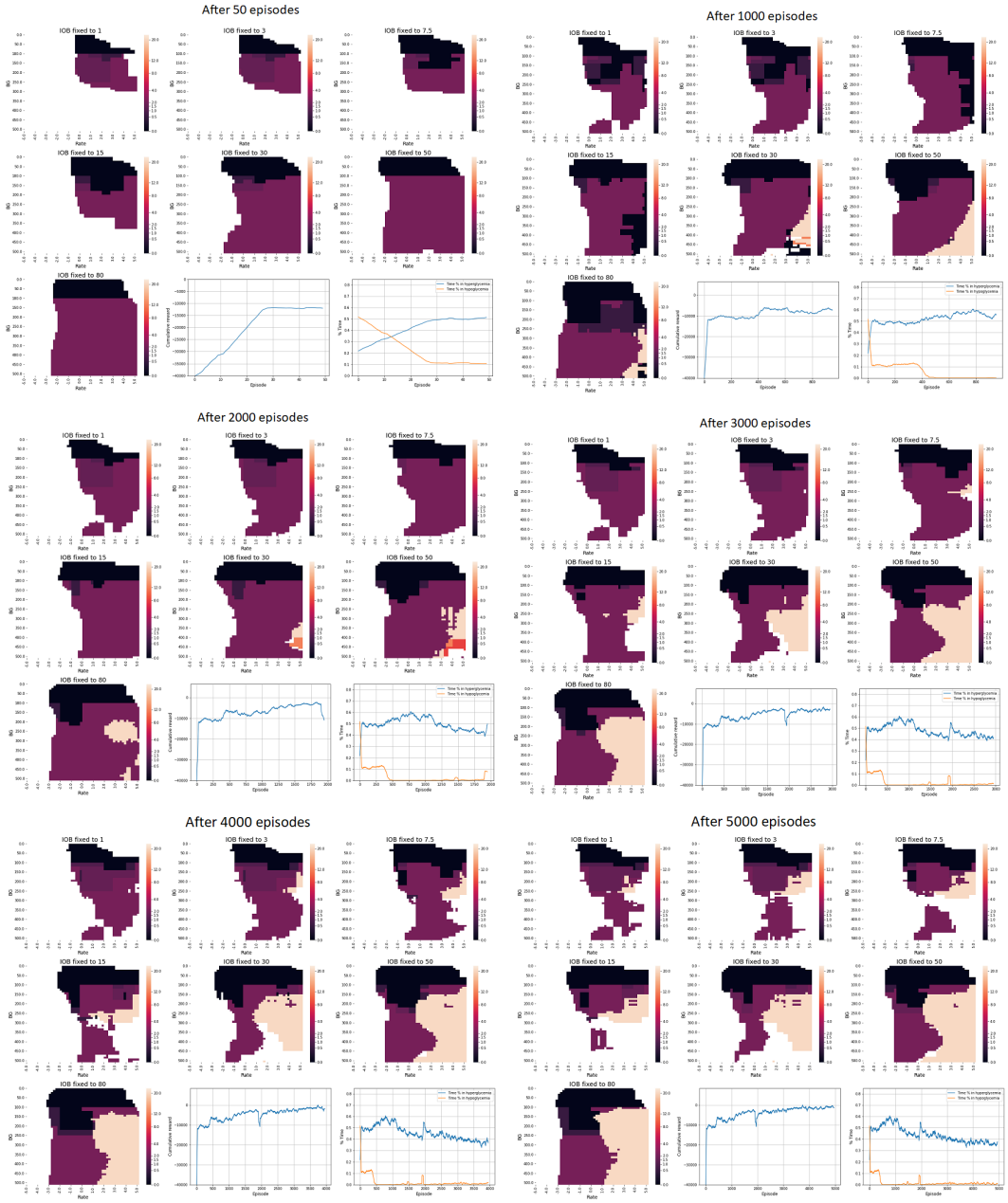

Figure 5.20: Evolution of the policy of an agent based on 4D and 1D tilings, throughout its training process. Changes are reported every 1000 episodes.

to be really cautious, administering only amounts of insulin close to the patient's BR. Throughout the training process, the agent is then trying to determine the proper region in the state space where strong actions should be employed, reacting more and more aggressively to meals.

This is evident by observing that the region relative to the strongest possible action is gradually widening as training proceeds. Eventually, a good policy is reached: the performance is in line with those previously seen, with a minimal amount of time spent in hypoglycemia and about 35% of the time spent in hyperglycemia. From there, further training doesn't improve the average performance, although it can still adjust the way the agent has to act in some rarely encountered regions of the state space. Note that, when compared to the previous agent, the final policy is more aggressive in the region of the state space relative to the last heatmap towards the end of the training process; at the same time, however, the areas associated to the avoidance of insulin delivery are also much wider, probably balancing the increased aggression.

This particular agent was picked to showcase how the system evolves because it also allows to show that training processes are not necessarily steadily improving at all times. After about 2000 episodes (plot in the middle left of Figure 5.20), a considerable drop in performance is noticeable: such drops are always associated with a disruption in policy, swaying considerably the evaluations built so far and slowing down the convergence to proper policies. The occurrence of such events is strictly linked to traumatic episodes; in fact, their frequency increases substantially if exploration is pursued actively.

Finally, ideally, the evolution of the system should be observed by looking at sequences with a much finer degree of separation, in terms of the number of episodes, between two successive visualizations. For this purpose, an animation of the training of the same agent, reporting changes in the policy every 100 iterations, can be watched in the repository of this thesis[88].

5.2.3 Repeatability and comparison with tabular approaches

Repeatability of the experiments is an aspect that hasn't been properly discussed yet. The last two policies seen are both obtained employing the same method, but, clearly, changing seed results in some slight differences, both in terms of final policy and learning speed. Due to computational limitations, it wasn't possible to test every experiment on several seeds; however, in order to have a clear idea of the differences in the training process between the tabular case and the function approximation case and to ensure that one of the main results is not due to chance, multiple agents were trained on seven different seeds, using both the tabular method and tile coding with 1D and 4D tilings, using the same features (BG concentration, BG rate, IOB) and the same action set. Figure 5.21 shows the results obtained: the median performances of the methods over time are reported by the solid lines, while the bands keep track of the worst and best performing seeds, for the metric considered, throughout the training process.

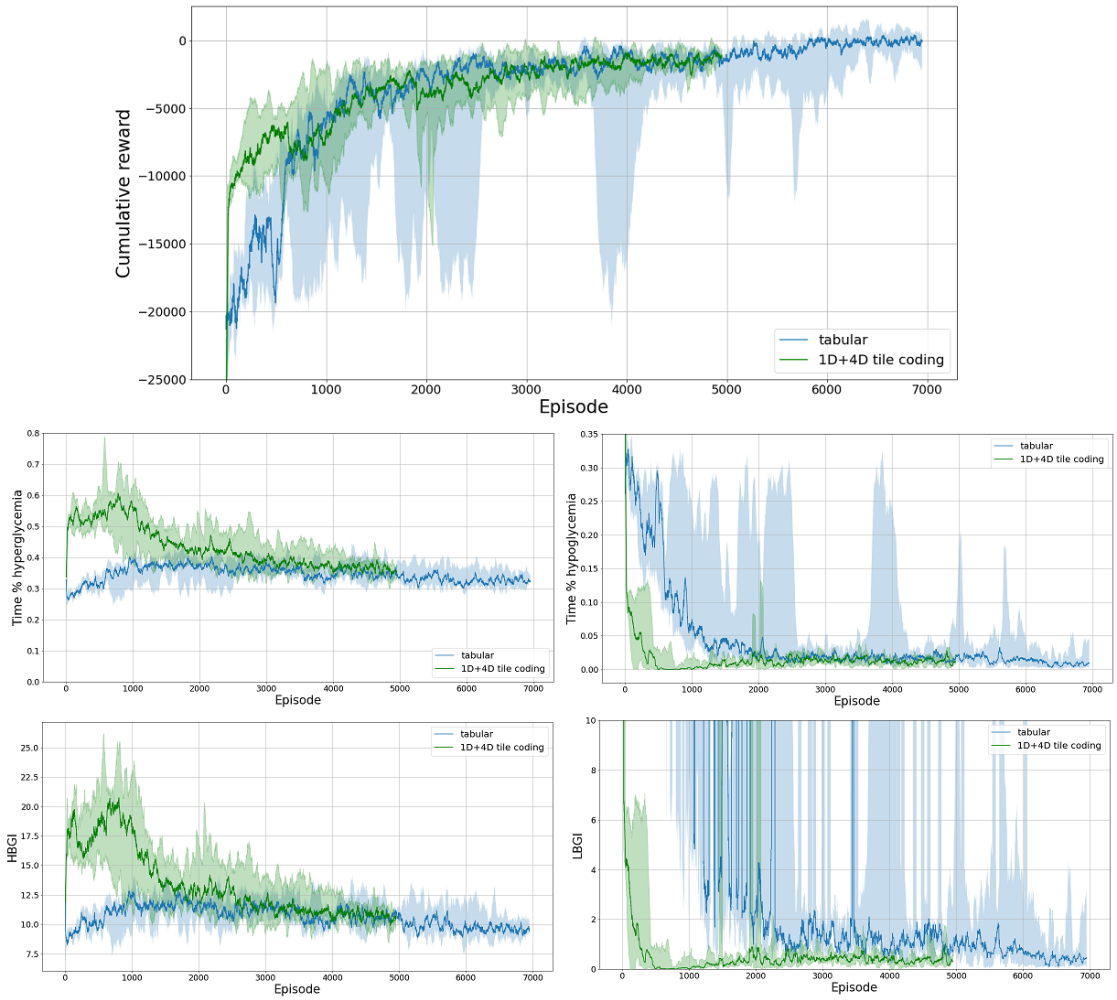

Figure 5.21: Comparison of the performance and the metrics of the tabular and the tile coding approach. Every method was run on seven different seeds. Solid lines refer to the median, bands spread from minimums to maximums.

The initial performance gap, due to mono-dimensional generalization in function approximation, is indeed always present, as visible from the graph relative to the cumulative reward. From the following two graphs, we can see that in the first couple of thousand of episodes, function approximation, being overly-cautious, struggles to deal with hyperglycemia, while tabular, instead, struggles with hypoglycemia and refrains from delivering too much insulin. After that, both tabular and function approximation runs tend to converge towards results which are similar in terms of the amount of time spent in euglycemia, although tile coding is still more cautious and often ends up spending a bit more time than necessary in hyperglycemia. For this reason, tabular runs have a slight advantage, when considering asymptotic performance: the

tabular agents spend about 3-4% more time in euglycemia at the end of training, on average. The difference in variability is also something to keep into account: tabular runs tend to be much more variable in their performance throughout learning; they also tend to be subject to sudden drops in performance and traumatic episodes, even late in the training process, as is particularly evident by the many spikes in the LBGI graph. This is linked, in part, to the wrong estimations associated to rare states, as pointed out in Section 5.1.5. Conversely, catastrophic failures in function approximation are quite rare after 200 to 500 episodes.

Overall, if safety during the training process and data efficiency are a concern, then function approximation approach seems to be more appropriate, especially in the early stages. If instead all that is desired is sheer performance of the policy reached, then, between the two, the tabular method proposed is the way to go. Of course, if this is the case, then we're essentially relying on the hope that the transfer of the final version of the policy trained on the simulator is enough to work properly without further training, which, as mentioned, can be quite variable.

In Table 5.6 some summary statistics of the distributions for the different metrics considered are reported, for both approaches, after the training process. Results are based on 1000 evaluation episodes for every seed; for every metric, the median and range between the 5th and 95th percentile are shown.

Method	Cumulative reward	% Time hyper	% Time hypo	% TIR	HBGI	LBGI
Tabular	$10\ (-5600, 4200)$	$33.3\ (19.4, 45.5)$	$0.0\ (0.0, 7.6)$	$65.6\ (53.1, 79.5)$	$9.4\ (5.0, 15.7)$	$0.1\ (0.0, 4.1)$
Tile coding (4D and 1D)	$-1400\ (-6200, 2950)$	$36.8\ (21.2, 51.4)$	$0.0\ (0.0, 5.9)$	$61,8\ (48.3, 76.7)$	$10.6\ (5.6, 16.8)$	$0.0\ (0.0, 1.7)$

Table 5.6: Summary statistics of the average performance across different seeds for the two main methods investigated.

5.3 Generalization across patients

For AP systems to be effective, the control algorithms employed are expected to be robust with respect to the substantial inter-variability in metabolic parameters that has been observed in the population of people with T1D. The results discussed so far, while promising, all refer to the same subject. In this section, the aspect of generalization to other subjects is tackled. Ten patients are considered, the first being the one considered so far. The training process is carried out without any prior knowledge of the problem, employing exactly the same methods presented: first the agents trained using Sarsa(λ) in the tabular setting will be discussed, then those using it with tile coding (with both 4D and 1D tilings). All agents are trained on the same random seed. Hyperparameters were optimized only once, on the first patient, and then fixed

for all subjects. For a detailed report of the hyperparameter optimization process, refer to the Appendix A. As a reminder, a slight degree of customization to the subject is already included in the way the action space is defined, depending on the patients' basal rate.

5.3.1 Tabular Sarsa(λ) on multiple patients

Results obtained by the tabular agents are reported in Table 5.7.

Subject	Cumulative reward	% Time Hyperglycemia	% Time Hypoglycemia	% TIR	HBGI	LBGI
1	500	31.2	1.2	67.6	8.6	0.7
2	2100	24.7	0.9	74.4	7.0	0.2
3	4000	22.1	0.1	77.8	5.1	0.2 *
4	3000	26.9	0.3	72.8	6.0	0.2
5	700	30.1	1.5	68.4	8.9	0.6
6	-4000	41.8	2.0	56.2	14.6	0.7
7	1250	31.0	0.7	68.3	8.5	0.2
8	2600	25.2	0.3	74.2	6.9	0.1
9	-7200 (1300)	61.2 (29.5)	1.6 (0.1)	37.2 (70.4)	15.4 (7.5)	113.4 (0.2)
10	500	32.3	1.1	66.6	9.2	0.4

Table 5.7: Performance and metrics of the final policies reached by the agents trained by the tabular approach.

For 8 out of 10 patients (subjects # 1,2,3,4,5,7,8,10) satisfactory policies are achieved, spending most of the time in euglycemia (a range approximately between 65 to 80 % is to be expected, depending on the patient). Time spent in hypoglycemia tends to be minimal, showing that the agents are capable of reacting to meals without overshooting to low BG levels; when it happens it's never critical, as it's clear by the LBGI metrics, well within appropriate levels of risk[79]. Note, however, a single exception: the mean LBGI relative to the 3rd patient is to be considered as reported when ignoring a single traumatic episode in the evaluation, which, if considered, would skew its value to 3.2 (LBGI, being a mean, is greatly influenced by outliers); such an episode is due to one of the notorious wrong-estimations associated with rare states in tabular approaches, which can be a threat to the safety of a policy and should be met by a direct intervention by the user, like CHO consumption. Time in hyperglycemia is in line with the experiments carried out before, if not even better; HBGI, in particular, shows that the policies, on average, are able to stay in a range of moderate risk[79, 80].

However, subjects 6 and 9 are problematic: in both cases a sub-optimal policy is achieved, struggling to contain hyperglycemia. For subject 6, the final policy is not terrible, but definitely

sub-par with respect to what was achieved for the other patients: its HBGI, in particular, is large enough for the policy to be close to high hyperglycemic risk. For subject 9, the method completely fails to reach a proper policy: not only are meals not met by an adequate reaction (notice how much time is spent in hyperglycemia), but also catastrophic episodes, where too much insulin is administered, resulting in deep hypoglycemia, are frequent. The LGBI metric explodes in value at such BG levels, when many such episodes are encountered in evaluation.

The reason for the sudden failure in the method for subject 9 was investigated: the metabolic parameters of the patient are quite unusual, when compared to the other subjects; specifically, the basal rate, upon which the available actions are based, is 30 to 50% larger than usual, resulting in a particularly aggressive action set. Similar problems in training processes were briefly observed in initial experiments of tabular methods when testing action sets that included actions too extreme for the method to handle. Based on these observations, an additional agent was trained for subject 9, using a less aggressive action set, allowing the administration of doses between 0 and 6 times the basal rate of the patient, all else unchanged. With this variation, the agent is capable of reaching the same level of performance as the other subjects (these results are reported within brackets for subject 9 in Table 5.7); the policy reached is actually one of the safest in terms of hypoglycemic risk. This case underlines that, sometimes, some components of the method might have to be adjusted, keeping in consideration the metabolic parameters of a patient: while it's true that vigorous actions usually allow for faster reactions to meals and, hence, reduced time in hyperglycemia, some subjects might not be able to cope with such actions as well as others.

The learning processes of all the tabular agents trained are visualized next; for ease of visual inspection, the graphs are shown in two distinct figures: Figure 5.22 reports it for the first five subjects and Figure 5.23 for the others.

For the first five patients (Figure 5.22), the method works quite well, even in terms of learning speed: within the first thousand episodes the performances improve substantially and hypoglycemia is reduced to less than 5% of the time; after that, the policies are gradually adjusted, perfecting the response to disturbances and reducing hyperglycemia, until reaching a plateau in performance between 1000 and 4000 iterations, depending on the subject. No improvements were achieved in the last 3000 episodes; the late variability observed is simply due to the large inter-episode variability and the availability of strong responses in the action set. The difference in performance between the subjects is strictly linked to differences in metabolic parameters and how easy it is to properly control the disturbances introduced into the system. The training process tends to be rather stable in terms of performance, without many drops.

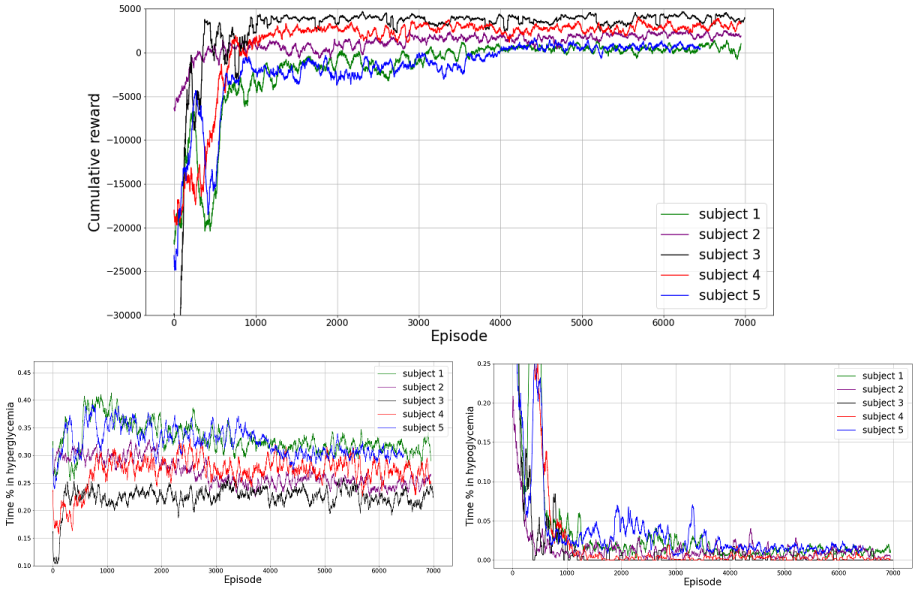

Figure 5.22: Training processes for the first five subjects in the tabular approach.

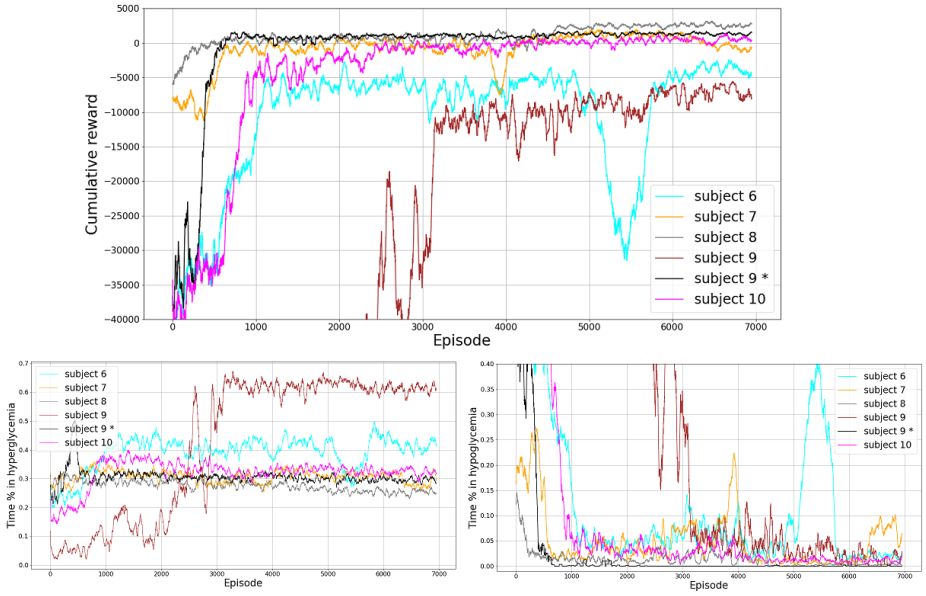

Figure 5.23: Training processes for the last five subjects in the tabular approach.Subject 9 * refers to the agent trained with a more cautious action set.

Looking at Figure 5.23, pretty much the same can be said about subjects 7, 8 and 10, except that some improvements can be noticed even late in the training for subject 8. Regarding subject 6, the training initially proceeds in line with the others but is not able to eventually

reach quite the same performance: it's possible that we're simply dealing with a rather problematic patient, and the policy reached is indeed close to the best that can be achieved. Even if this was the case, it's important to note that overall the training process is associated with higher variability than usual, and a sudden drop in performance, lasting over 1000 episodes. As mentioned before, for subject 9, the scenario is even worse, when considering the usual action set: for over 3000 episodes the agent spends most of the time in deep hypoglycemia and it's struggling to make progress. Eventually it does, but the policy reached is far from being acceptable. Additional training after 7000 iterations (not included in the graph) doesn't lead to further improvements. On the other hand, when the more conservative action set is employed, the training process proceeds smoothly even for subject 9, in fact pretty much on par with the other subjects, both in terms of learning speed and performance achievable.

It's clear that, overall, the tabular approach proposed is capable of reaching adequate policies for BG control for several T1D patients. However, some components may have to be tweaked to make it work properly at all times.

5.3.2 Tile coding Sarsa(λ) on multiple patients

Results obtained in 1000 episode evaluations for every agent based on tile coding, implemented with 4D and 1D tilings, are reported in Table 5.8.

Two approaches were tested: without any active exploration (as discussed so far) and with minimal active exploration. The reason for the introduction of exploration is that for some patients (# 4, 8 and 9), the usual approach got repeatedly stuck in suboptimal policies, completely avoiding the actions related to the administration of large doses of insulin in short amounts of time. This wasn't ever the case for subject 1 in previous experiments. This issue is due to a combination of factors: mono-dimensional generalization, as seen previously in Figure 5.12, pushes to avoid strong actions, unless it's deemed necessary by 4D contributions; in the case where a policy completely avoids the use of such actions in every part of the state space, there's no way for a method which is not exploring to generalize back the concept that they might be useful. This is problematic, because, as mentioned, classic ϵ-greedy exploration frequently leads to prolonged degradation of performance, due to the large TD-errors it produces in some episodes, when exploring at the "wrong" moment, leading the agent into deep hypoglycemia. The speed of training, due to the long-term consequences and the delay between action and effect in our task, is drastically reduced using active exploration, especially when frequent exploration is used in the early stages of training, when most of the actions taken by the agent

Subject	Active exploration	Cumul. reward	% Time Hyper	% Time Hypo	% TIR	Δ % TIR wrt tabular	HBGI	LBGI
1	No	-1300	36.6	0.8	62.6	-5.0	10.5	0.3
1	Yes	-1850	37.9	0.8	61.3	-6.3	11.7	0.8
2	No	2000	25.4	1.9	72.7	-1.7	6.7	0.5
2	Yes	3300	18.7	3.0	78.3	+3.9	4.8	0.8
3	No	5300	17.6	0.7	81.7	+3.9	3.9	0.2
3	Yes	5000	17.4	1.0	81.6	+3.8	3.8	0.3
4	No	-1300	34.9	6.8	58.3	-14.5	8.3	1.5
4	Yes	2300	28.4	0.2	71.4	-1.4	6.8	0.1
5	No	-600	32.6	2.7	64.7	-3.7	9.3	1.4
5	Yes	100	31.6	0.7	67.7	-0.7	9.4	0.5
6	No	-3650	40.9	1.4	57.7	+1.5	14.2	0.6
6	Yes	-3650	39.6	2.1	58.3	+2.1	14.2	2.4
7	No	-950	30.4	6.7	62.9	-5.4	8.8	1.4
7	Yes	-150	33.6	1.9	64.5	-3.8	9.6	0.4
8	No	-550	36.2	0.1	63.7	-10.5	10.3	0.1
8	Yes	50	26.4	7.9	65.7	-8.5	7.0	1.4
9	No	-700	39.5	1.6	58.9	+21.7 (-11.5)	8.9	0.4
9	Yes	3500	23.1	0.6	76.3	+39.1 (+5.9)	5.6	0.2
10	No	-1300	35.1	2.1	62.8	-3.8	10.8	0.7
10	Yes	-1200	32.2	4.7	63.1	-3.5	9.4	1.5

Table 5.8: Performances and metrics of the final policies reached by the agents trained by tile coding approaches. % TIR wrt tabular within brackets for subject 9 refers to the comparison to the tabular method employing more cautious actions.

are already bad. Tile coding, however, as seen in the previous sections, allows for a rapid improvement in performance in the first hundred episodes, resulting in overly-cautious policies. As a consequence, agents can be trained with the ability to take infrequent exploratory actions, but only after the initial generalization; the probability ϵ of taking random actions was set to quickly decay over time, proportionally to the inverse of the number of the episodes already met, until it was practically negligible at the end of training. The exploration is minimal, in the sense that most episodes simply always follow the actions associated with the largest q-values, but sometimes the agent is "reminded" that using stronger actions can be beneficial. Overall, this variation allows to achieve better policies for subjects where the previous usual method got stuck, although it clearly doesn't solve the problem completely, since for some patients (subject # 8) achieving similar performance to the tabular case is still impossible.

For most of the patients, tile coding methods are able to achieve performances close to the tabular method, in terms of time in normoglycemia and cumulative reward, however some key differences should be noted regarding the other metrics. In general function approxima-

tion methods, due to generalization, when refining the policy are more inclined to reach lower minimums (between 55 and 70) in between meals. Note that this is why the percentage of time spent in hypoglycemia is sometimes substantially larger than the tabular counterparts; however, the LGBI metric shows that the policies are still within the range of low-risk for hypoglycemia.

In the following ten figures (Figure 5.24 to Figure 5.33) the comparisons of the training processes between tabular Sarsa(λ) (in blue) and tile coding Sarsa(λ) methods with and without exploration (in green and in red, respectively) are shown. For every subject, cumulative reward (upper subplot) and percentages of time in hyperglycemia (lower left) and hypoglycemia (lower right) are considered. Note that the same random seed is used for all agents.

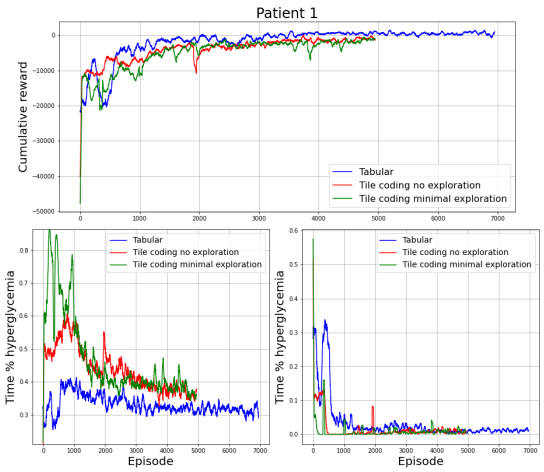

Figure 5.24: Training processes comparison for subject 1.

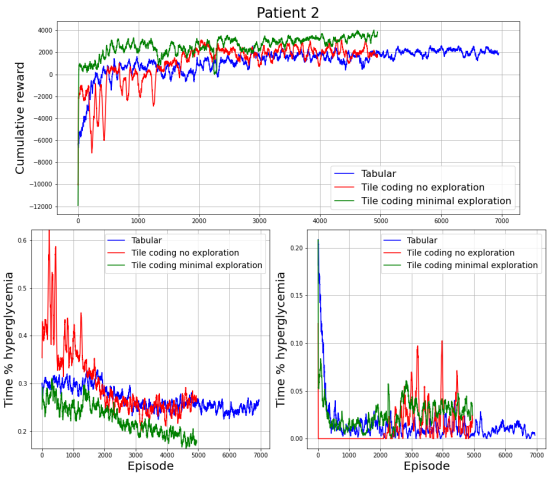

Figure 5.25: Training processes comparison for subject 2.

Overall, most considerations that were made in Section 5.2.3 for the differences in the training processes of the first patient are valid for all subjects. Early generalization in tile coding quickly reduces the amount of time spent in hypoglycemia early in training, usually way before the tabular approach (100-200 episodes against 500-1000 episodes). The two approaches then struggle in training for opposite reasons. Function approximation agents tend to be overly cautious and spend more time than necessary in hyperglycemia, gradually determining in which part of the state space to use strong actions (training processes and final policies are similar to what was discussed for system evolution in Section 5.2.2); note that agents making use of minimal active exploration are usually dealing with hyperglycemia faster and don't get stuck as easily. Conversely, tabular agents respond too aggressively to meals, and most of the training is spent gradually refining the response in order to avoid overshooting into hypoglycemia. Convergence to proper policies is usually achieved in 1000 to 4000 episodes for both methods.

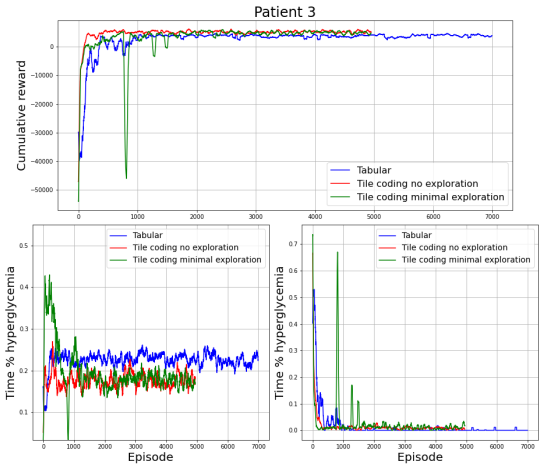

Figure 5.26: Training processes comparison for subject 3.

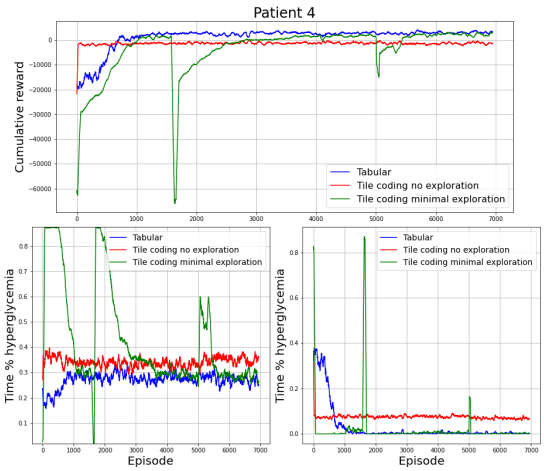

Figure 5.27: Training processes comparison for subject 4.

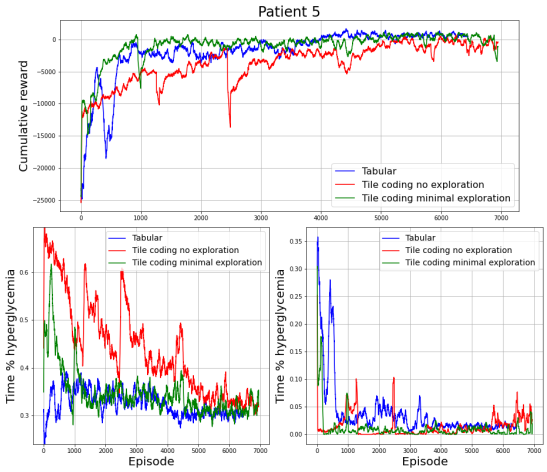

Figure 5.28: Training processes comparison for subject 5.

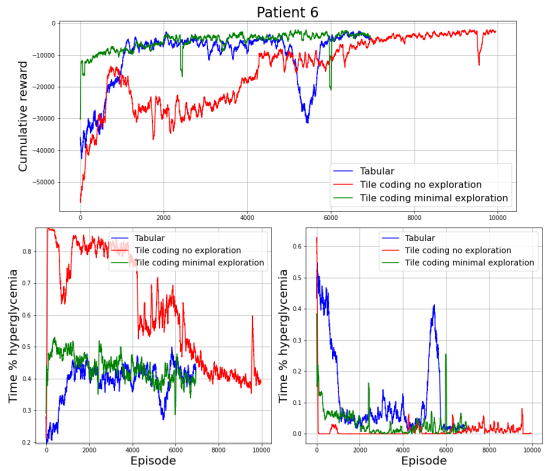

Figure 5.29: Training processes comparison for subject 6.

There are some clear exceptions to the general observations made: training processes for subject 6 tend to be problematic and lengthy, achieving sub-par performances with respect to the other subjects; subject 8 is also particularly anomalous both for the singular late improvement in the tabular method, which isn't replicated in function approximation, and the fast avoidance of hypoglycemia, with respect to tile coding, in the tabular setting.

Finally, for subject 9, it was especially interesting to observe that both tile coding methods autonomously generalize the use of prudent actions (4 to 8 times the patient's BR) in response to meals, instead of using the strongest available actions. This validates the idea that using a more conservative action set was necessary in the tabular setting for this subject; in fact, all these ap-

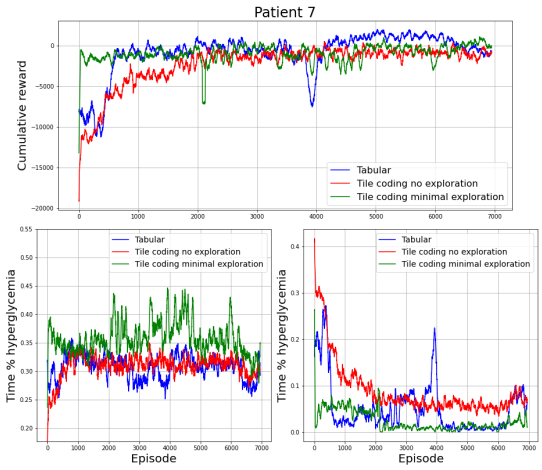

Figure 5.30: Training processes comparison for subject 7.

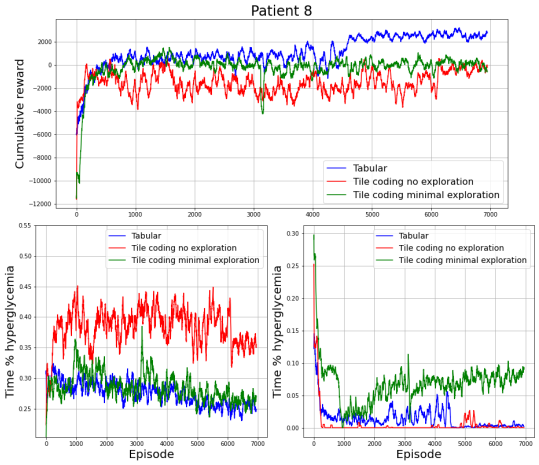

Figure 5.31: Training processes comparison for subject 8.

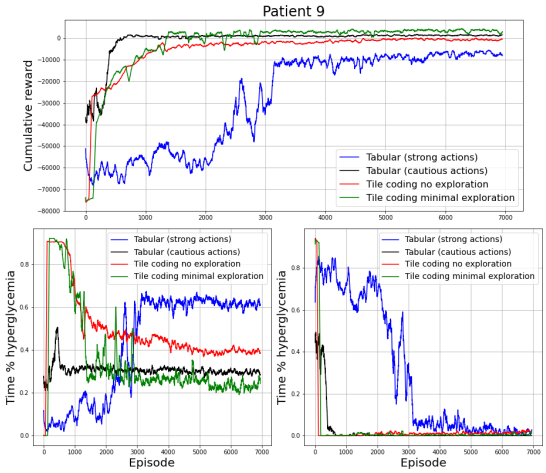

Figure 5.32: Training processes comparison for subject 9. In black the tabular agent with more cautious actions.

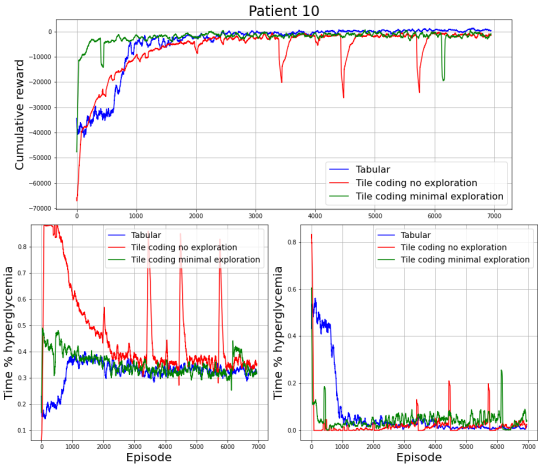

Figure 5.33: Training processes comparison for subject 10.

proaches vastly outperform the tabular method which uses the standard action set considered.

5.3.3 Policy transfer and personalization

So far, the policies reached were evaluated only on the subjects on which they were trained. However, it's also possible to evaluate them on the other patients: the Q-values table (in the tabular method case) or the weight vector (when using tile coding) can simply be transferred to be used in a context where data is generated using different metabolic parameters, representing other subjects. Having access to the parameters of ten subjects, the ten tabular policies

discussed in Section 5.3.1 and the ten linear function approximation policies discussed in Section 5.3.2 (those using minimal exploration are considered, as they appear to work better) are evaluated for every patient. 250 evaluation episodes are run for every possible combination of subject and policy; no learning is involved in the process. Note that Q-values are associated with actions that represent multipliers of a patient's basal in both the tabular and function approximation cases, hence, while the shared policy can be the same (e.g. select the action relative to 2 times the BR), the insulin doses administered aren't the same across patients.

The scope of these experiments is twofold: to understand if the policies are personalized to the patients' needs and if they're transferable to other subjects. Results are reported as 10x10 matrices of the three metrics % TIR, HBGI and LBGI, represented as heatmaps. In a matrix M, every entry $m_{i,j}$ is associated with a policy trained on patient j and evaluated on patient i.

Policy transfer for the tabular case is discussed first: results are shown in Figure 5.34. Note that for subject 9 the policy making use of a more conservative action set is considered.

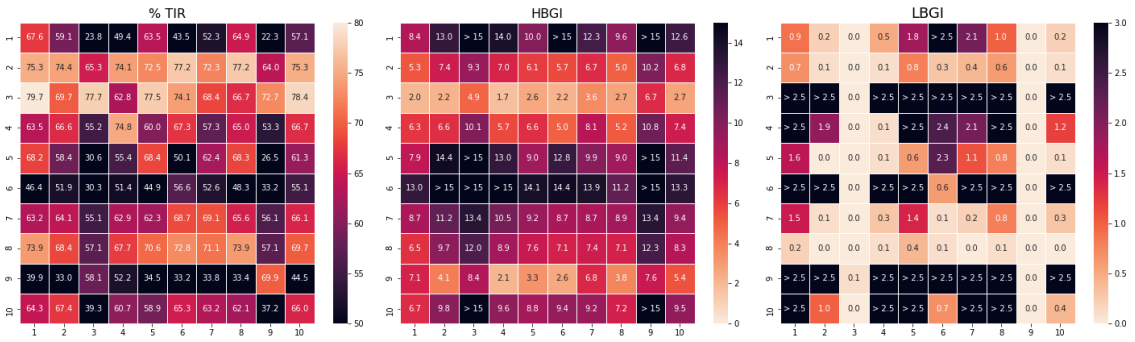

Figure 5.34: 10x10 Heatmaps of %TIR, HBGI, and LBGI for tabular policy transfer.

Looking at the heatmaps, understanding which policies are adequate for every subject isn't straightforward: the results obtainable differ widely between subjects, making it hard to rely only on colors, and comparisons must be done within a row; furthermore, it's necessary to jump between heatmaps to determine if there are some aspects of a policy that make it unfit for safe utilization. To aid this selection process, it's possible to set some rules by which we define a policy as being adequate for a subject. So far, all the policies obtained (both for tabular and function approximation) have never breached into the high risk zone of hyperglycemia (HBGI > 15) or the moderate risk zone of hypoglycemia (LBGI > 2.5)[80]; this property provides appropriate thresholds to set with regard to hyperglycemic and hypoglycemic risks. Furthermore, it's likely a good idea to restrict the definition of adequate policies only to those within 5% of TIR to the maximum obtained, for every subject. If we use these three simple rules, we can

obtain a mask for the heatmaps, shown in Figure 5.35, reporting in white the entries associated with policies that can be adequate for safe use, and in black unfit policies.

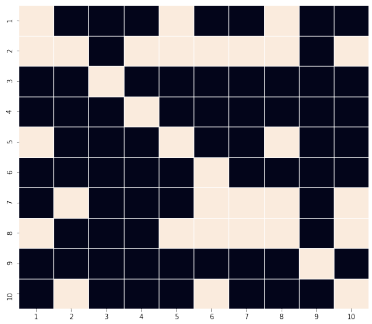

Figure 5.35: Adequacy mask M for tabular policy transfer heatmaps. An entry m_{ij} in white is associated with a policy trained on subject j and evaluated on subject i, which is fit for use with subject i.

Note that all the policies on the diagonal are adequate. This means that all the policies reached using a subject's metabolic parameters are adequate for use with that subject. Applying the mask crafted to the heatmaps shown in Figure 5.34, the more readable Figure 5.36 is obtained: making comparisons at each row and jumping between heatmaps, it's clear that not only are the policies on the diagonal adequate, but they're usually the best among all the policies trained, implying that they're personalized for the patient. The only clear exception to this observation is subject 2: several policies are adequate for safe use with this patient and some are possibly better than the one trained on his/her data; for example, subject 6's policy is achieving slightly better results when followed by subject 2.

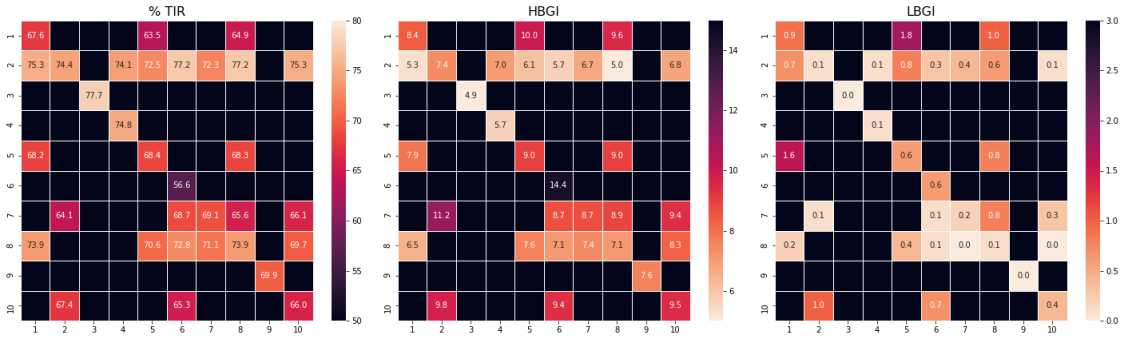

Figure 5.36: Heatmaps of tabular policy transfer metrics with adequacy mask imposed. Only metrics relative to policies deemed adequate are shown.

Exactly the same procedure can be followed for the linear function approximation case: heatmaps reporting % TIR, HBGI, and LBGI for every possible pair of policy and subject are shown in Figure 5.37.

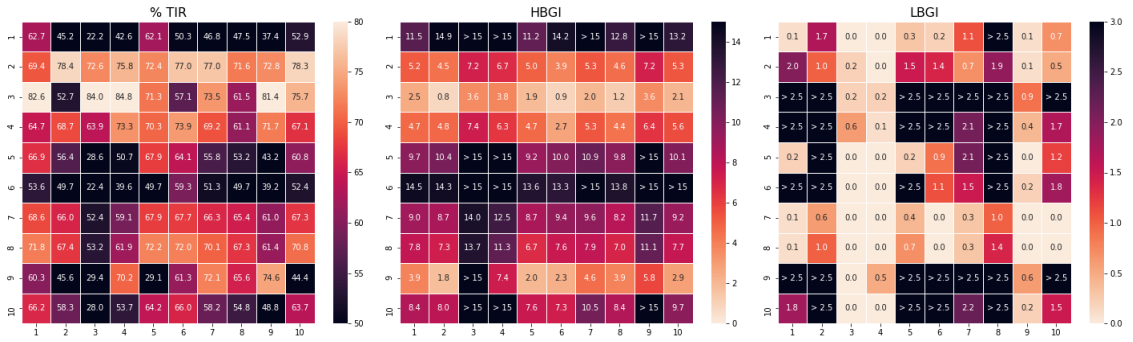

Figure 5.37: 10x10 Heatmaps of %TIR, HBGI, and LBGI for linear function approximation (tile coding, 4D and 1D tilings) policy transfer.

The adequacy mask is then crafted using the same rules used for the tabular case (HBGI < 15, LBGI < 2.5 and % TIR within 5% to the maximum obtained for every subject) and is shown in Figure 5.38.

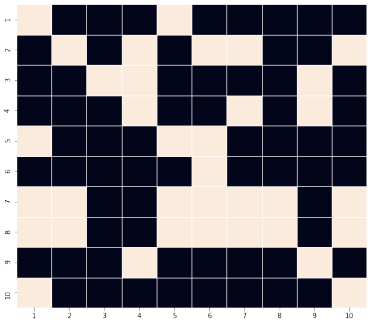

Figure 5.38: Adequacy mask M for linear function approximation policy transfer heatmaps. An entry m_{ij} in white is associated with a policy trained on subject j and evaluated on subject i, which is fit for use with subject i.

Note that, as it was for the tabular case, all policies trained on a subject are labeled as adequate for appropriate use for that subject, i.e. all entries in the diagonal satisfy the requisites imposed. When comparing it with Figure 5.35, in both cases subjects 2, 7, and 8 are the patients that are able to often use other subjects' policies safely; however, there's not a clear indication as to which method provides better transferability overall. The mask is used to filter out the unfit policies in the heatmap plots, obtaining Figure 5.39.

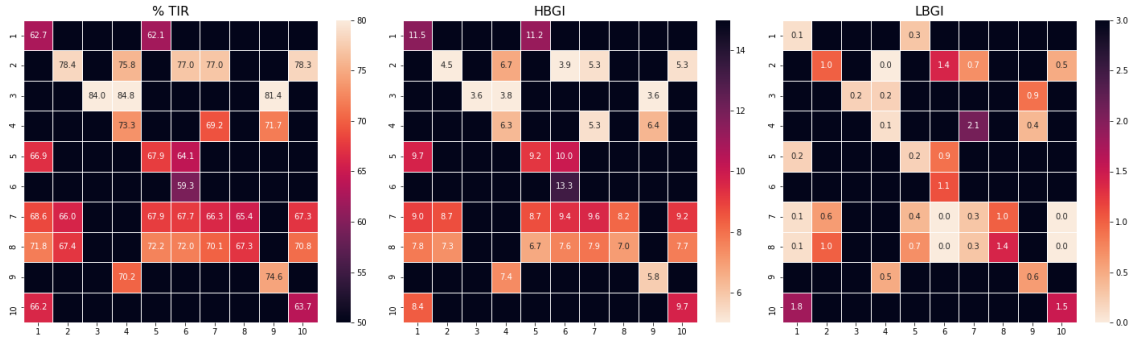

Figure 5.39: Heatmaps of linear function approximation policy transfer metrics with adequacy mask imposed. Only metrics relative to policies deemed adequate are shown.

While in most cases it's still true that for each subject his/her own policy is usually close to the best performance, linear function approximation seems to have more difficulty than the tabular method with respect to the personalization of the policy: for three out of ten subjects (7, 8, 10) better performances are achieved by policies trained on other individuals, as opposed to the only case spotted in the tabular case.

Overall, it's clear that the methods employed grant a good degree of personalization to the patients' needs but also sometimes allow for transferability of policies between patients. This is an aspect that should be investigated more in depth in the future, especially in view of methods that safely initialize the training process with knowledge from past data.

6 Conclusion

6.1 Main results

This thesis aimed to evaluate reinforcement learning methods based on the Sarsa(λ) algorithm as a control algorithm in an artificial pancreas system, employing the UVa/Padova Type 1 Diabetes simulator. Evaluation was carried out analyzing how performances vary depending on the way the problem is represented, describing and interpreting differences within final policies and training processes, and considering how the system is able to generalize to different subjects. To my knowledge, approaches based on Sarsa(λ) and state representations built by tile coding have not been properly investigated in previous studies of the AP literature.

Results indicate that Sarsa(λ) approaches based on both classic tabular RL and linear function approximation RL are able to reach appropriate policies to handle realistic meal schedules, closely emulating the distributions observed in the general T1D population, for all in silico patients considered. As expected, given the usual large variability observed in T1D patients, the percentage of time in normoglycemia which is possible to achieve varies widely between subjects (from 56% to 82%); however, appropriate policies reached are always within acceptable hyperglycemic and hypoglycemic risk levels, as measured by the HBGI and LBGI metrics. It was also briefly shown that it's possible to boost performances by considering hybrid closed loop systems that include in the state representation features derived from external information about CHO consumption; the magnitude of the improvement is associated with the accuracy

of the information encoded in the new feature, although even approximate information can be beneficial. Furthermore, as demonstrated by the results obtained transferring policies between subjects, the methods employed resulted in a good degree of personalization of the control algorithm to the patients' needs, especially in the tabular case. However, reaching optimal policies is often complicated, especially due to the difficulties that exploration entails in such a task.

Tracking training processes through different metrics showed that linear function approximation methods and tabular methods usually struggle for opposite reasons with glycemic control. Approaches based on tile coding, in fact, tend to be overly-cautious about delivering large amounts of insulin, resulting in large percentages of episode-time spent in hyperglycemia in the early stages of learning, followed by a gradual increased aggression towards disturbances. Conversely, agents trained in tabular approaches tend to administer too much insulin in the initial phases of learning and gradually balance their responses to meals as training proceeds. The two methods, although the policies they reach are usually quite different, tend to converge to similar performances at the end of the training process, for most subjects.

Finally, explainability of the agent's decision-making process is an aspect that is rarely discussed in the context of RL, but in safety-critical systems, such as the artificial pancreas, its discussion should be one of the main focuses, as it is clearly a prerequisite to possible future studies on real patients and could even help to manually craft or adjust policies obtained with different methods. For this reason, taking advantage of the small number of features employed in the state representation, in this work some visualization techniques were proposed that allowed to easily gain an intuition about the reasoning behind the decisions made by the agents and the expected return in every state encountered. This allowed us to spot possible inaccuracies in the policies and to keep track of the evolution of a RL system during the training process.

6.2 Future outlook and ideas

This work showed that Sarsa(λ) can surely be appropriate as a control algorithm in an AP system, but it wasn't an exhaustive review of the methods where it could be employed. In particular, tile coding is a very specific instance of linear function approximation, and several other expressive methods are available to obtain linear features; for example, it would be interesting in future studies to compare the performances and policies obtainable by using polynomials features, fourier basis features, and RBF features [21].

It should also be noted that many other further steps have to be taken in order to build

RL-based AP systems that are ready to be used in real conditions. First of all, simulation can certainly be useful for preliminary studies as the one presented, to filter out methods which are surely not adequate to be employed, but eventually RL methods should likely take in consideration other sources of disturbances to glycemic control, like stress, physical activity, hormonal factors, and many others. Furthermore, if AP systems will be built with the idea to continually learn from new data, adapting to changes in habits and/or physiology of a patient, then two topics will have to be carefully examined in the future in order to guarantee the safety of the patients: the introduction of prior knowledge and the issue of exploration.

Several ways can be envisioned to introduce prior knowledge. Transferring the policies of agents previously trained in simulation or the policies followed by other patients is the most obvious way to provide a decent starting point for new patients. A particularly attractive idea, in this context, could be to cluster T1D patients into groups, based on their metabolic parameters and other personal factors (e.g. gender, age, weight, prior illnesses), and identify a minimal set of adequate safe policies that work particularly well for each group. A new patient's policy could then be initialized to the policy of a specific group, chosen based on the similarity of his/her own parameters to the groups' average parameters. To achieve personalization, a refinement process could then be carried out based on the newly observed data. In this type of scenario, it's likely that active exploration should be actively pursued to avoid remaining stuck in the initial policy, optimized for other patients, but not optimal for the current subject. In this context, exploration is probably easier than what has been observed throughout this thesis, since the initial transferred policy should be quite good already. Even so, for safety reasons, exploration should be guided by prior knowledge, allowing vigorous actions only in proper contexts, to avoid traumatic events and possible unusual and disruptive TD-errors that can have unintended detrimental effects on the policy. For example, some simple logic-based rules, taking into consideration the main features described in the thesis, could likely be crafted to determine when it's deemed safe for an agent to explore.

From all previous observations and most of the results obtained throughout the thesis, it's clear that RL-based AP systems could greatly benefit from being paired with complementary safety modules. The framework provided by RL methods is certainly attractive and adequate to be employed for glycemic control, but, given the complexity of the task, especially if the policy is actively changing during the use of an AP system, safety modules could aid training processes and provide an additional layer of protection.

References

[1] I. D. Federation. (2021) Idf diabetes atlas 10th. [Online]. Available: https://diabetesatlas.org/

[2] N. Cho, J. Shaw, S. Karuranga, Y. Huang, J. da Rocha Fernandes, A. Ohlrogge, and B. Malanda, "Idf diabetes atlas: Global estimates of diabetes prevalence for 2017 and projections for 2045," *Diabetes Research and Clinical Practice*, vol. 138, pp. 271–281, 2018.

[3] World Health Organization. Diabetes fact sheet. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes

[4] B. E. Klein, R. Klein, P. E. McBride, K. J. Cruickshanks, M. Palta, M. D. Knudtson, S. E. Moss, and J. O. Reinke, "Cardiovascular disease, mortality, and retinal microvascular characteristics in type 1 diabetes: Wisconsin epidemiologic study of diabetic retinopathy," *Archives of internal medicine*, vol. 164, no. 17, pp. 1917–1924, 2004.

[5] D. Control, C. Trial, E. of Diabetes Interventions, C. R. Group *et al.*, "Prolonged effect of intensive therapy on the risk of retinopathy complications in patients with type 1 diabetes mellitus: 10 years after the diabetes control and complications trial," *Archives of ophthalmology*, vol. 126, no. 12, pp. 1707–1715, 2008.

[6] M. Vettoretti, G. Cappon, G. Acciaroli, A. Facchinetti, and G. Sparacino, "Continuous glucose monitoring: current use in diabetes management and possible future applications," *Journal of diabetes science and technology*, vol. 12, no. 5, pp. 1064–1071, 2018.

[7] B. H. McAdams and A. A. Rizvi, "An overview of insulin pumps and glucose sensors for the generalist," *Journal of clinical medicine*, vol. 5, no. 1, p. 5, 2016.

[8] H. Thabit and R. Hovorka, "Coming of age: the artificial pancreas for type 1 diabetes," *Diabetologia*, vol. 59, no. 9, pp. 1795–1805, 2016.

[9] K. Kumareswaran, M. L. Evans, and R. Hovorka, "Closed-loop insulin delivery: towards improved diabetes care," *Discovery medicine*, vol. 13, no. 69, pp. 159–170, 2012.

[10] S. A. Weinzimer, G. M. Steil, K. L. Swan, J. Dziura, N. Kurtz, and W. V. Tamborlane, "Fully automated closed-loop insulin delivery versus semiautomated hybrid control in pediatric patients with type 1 diabetes using an artificial pancreas," *Diabetes care*, vol. 31, no. 5, pp. 934–939, 2008.

[11] R. S. Surwit, M. S. Schneider, and M. N. Feinglos, "Stress and diabetes mellitus," *Diabetes care*, vol. 15, no. 10, pp. 1413–1422, 1992.

[12] S. R. Colberg, R. J. Sigal, J. E. Yardley, M. C. Riddell, D. W. Dunstan, P. C. Dempsey, E. S. Horton, K. Castorino, and D. F. Tate, "Physical activity/exercise and diabetes: a position statement of the american diabetes association," *Diabetes care*, vol. 39, no. 11, pp. 2065–2079, 2016.

[13] S. D. Favero. (2022) Lectures on control of biological systems. [Online]. Available: https://en.didattica.unipd.it/off/2020/LM/IN/IN2546/000ZZ/INQ0091285/No

[14] G. P. Forlenza, S. Deshpande, T. T. Ly, D. P. Howsmon, F. Cameron, N. Baysal, E. Mauritzen, T. Marcal, L. Towers, B. W. Bequette *et al.*, "Application of zone model predictive control artificial pancreas during extended use of infusion set and sensor: a randomized crossover-controlled home-use trial," *Diabetes Care*, vol. 40, no. 8, pp. 1096–1102, 2017.

[15] P. Herrero, P. Georgiou, N. Oliver, D. G. Johnston, and C. Toumazou, "A bio-inspired glucose controller based on pancreatic β-cell physiology," *Journal of diabetes science and technology*, vol. 6, no. 3, pp. 606–616, 2012.

[16] I. Contreras, J. Vehi *et al.*, "Artificial intelligence for diabetes management and decision support: literature review," *Journal of medical Internet research*, vol. 20, no. 5, p. e10775, 2018.

[17] C. Pérez-Gandía, A. Facchinetti, G. Sparacino, C. Cobelli, E. Gómez, M. Rigla, A. de Leiva, and M. Hernando, "Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring," *Diabetes technology & therapeutics*, vol. 12, no. 1, pp. 81–88, 2010.

[18] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[20] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–36, 2021.

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018.

[22] D. Silver. (2015) Lectures on reinforcement learning. [Online]. Available: https://www.davidsilver.uk/teaching/

[23] D. B. Keenan, J. J. Mastrototaro, G. Voskanyan, and G. M. Steil, "Delays in minimally invasive continuous glucose monitoring devices: a review of current technology," *Journal of diabetes science and technology*, vol. 3, no. 5, pp. 1207–1214, 2009.

[24] M. K. Bothe, L. Dickens, K. Reichel, A. Tellmann, B. Ellger, M. Westphal, and A. A. Faisal, "The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas," *Expert review of medical devices*, vol. 10, no. 5, pp. 661–673, 2013.

[25] F. J. Doyle III, L. M. Huyett, J. B. Lee, H. C. Zisser, and E. Dassau, "Closed-loop artificial pancreas systems: engineering the algorithms," *Diabetes care*, vol. 37, no. 5, pp. 1191–1197, 2014.

[26] G. M. Steil, K. Rebrin, C. Darwin, F. Hariri, and M. F. Saad, "Feasibility of automating insulin delivery for the treatment of type 1 diabetes," *Diabetes*, vol. 55, no. 12, pp. 3344–3350, 2006.

[27] L. H. Messer, G. P. Forlenza, J. L. Sherr, R. P. Wadwa, B. A. Buckingham, S. A. Weinzimer, D. M. Maahs, and R. H. Slover, "Optimizing hybrid closed-loop therapy in adolescents and emerging adults using the minimed 670g system," *Diabetes Care*, vol. 41, no. 4, pp. 789–796, 2018.

[28] K. A. Wintergerst, D. Deiss, B. Buckingham, M. Cantwell, S. Kache, S. Agarwal, D. M. Wilson, and G. Steil, "Glucose control in pediatric intensive care unit patients using an insulin–glucose algorithm," *Diabetes technology & therapeutics*, vol. 9, no. 3, pp. 211–222, 2007.

[29] M. D. Breton and B. P. Kovatchev, "Impact of blood glucose self-monitoring errors on glucose variability, risk for hypoglycemia, and average glucose control in type 1 diabetes: an in silico study," *Journal of Diabetes Science and Technology*, vol. 4, no. 3, pp. 562–570, 2010.

[30] E. Renard, A. Farret, J. Kropff, D. Bruttomesso, M. Messori, J. Place, R. Visentin, R. Calore, C. Toffanin, F. Di Palma *et al.*, "Day-and-night closed-loop glucose control in patients with type 1 diabetes under free-living conditions: results of a single-arm 1-month experience compared with a previously reported feasibility study of evening and night at home," *Diabetes Care*, vol. 39, no. 7, pp. 1151–1160, 2016.

[31] G. P. Forlenza, S. Deshpande, T. T. Ly, D. P. Howsmon, F. Cameron, N. Baysal, E. Mauritzen, T. Marcal, L. Towers, B. W. Bequette *et al.*, "Application of zone model predictive control artificial pancreas during extended use of infusion set and sensor: a randomized crossover-controlled home-use trial," *Diabetes Care*, vol. 40, no. 8, pp. 1096–1102, 2017.

[32] Y. Li, K. H. Ang, and G. C. Chong, "Pid control system analysis and design," *IEEE Control Systems Magazine*, vol. 26, no. 1, pp. 32–41, 2006.

[33] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering *et al.*, "Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes," *Physiological measurement*, vol. 25, no. 4, p. 905, 2004.

[34] L. Magni, D. M. Raimondo, L. Bossi, C. Dalla Man, G. De Nicolao, B. Kovatchev, and C. Cobelli, "Model predictive control of type 1 diabetes: an in silico trial," 2007.

[35] B. Grosman, E. Dassau, H. C. Zisser, L. Jovanovič, and F. J. Doyle III, "Zone model predictive control: a strategy to minimize hyper-and hypoglycemic events," *Journal of diabetes science and technology*, vol. 4, no. 4, pp. 961–975, 2010.

[36] C. Cobelli, E. Renard, and B. Kovatchev, "Artificial pancreas: past, present, future," *Diabetes*, vol. 60, no. 11, pp. 2672–2682, 2011.

[37] S. Del Favero, J. Place, J. Kropff, M. Messori, P. Keith-Hynes, R. Visentin, M. Monaro, S. Galasso, F. Boscari, C. Toffanin *et al.*, "Multicenter outpatient dinner/overnight reduction of hypoglycemia and increased time of glucose in target with a wearable artificial

pancreas using modular model predictive control in adults with type 1 diabetes," *Diabetes, Obesity and Metabolism*, vol. 17, no. 5, pp. 468–476, 2015.

[38] L. Leelarathna, S. Dellweg, J. K. Mader, J. M. Allen, C. Benesch, W. Doll, M. Ellmerer, S. Hartnell, L. Heinemann, H. Kojzar *et al.*, "Day and night home closed-loop insulin delivery in adults with type 1 diabetes: three-center randomized crossover study," *Diabetes Care*, vol. 37, no. 7, pp. 1931–1937, 2014.

[39] J. Kropff, S. Del Favero, J. Place, C. Toffanin, R. Visentin, M. Monaro, M. Messori, F. Di Palma, G. Lanzola, A. Farret *et al.*, "2 month evening and night closed-loop glucose control in patients with type 1 diabetes under free-living conditions: a randomised crossover trial," *The lancet Diabetes & endocrinology*, vol. 3, no. 12, pp. 939–947, 2015.

[40] H. Thabit, M. Tauschmann, J. M. Allen, L. Leelarathna, S. Hartnell, M. E. Wilinska, C. L. Acerini, S. Dellweg, C. Benesch, L. Heinemann *et al.*, "Home use of an artificial beta cell in type 1 diabetes," *New England Journal of Medicine*, vol. 373, no. 22, pp. 2129–2140, 2015.

[41] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[42] E. Atlas, R. Nimri, S. Miller, E. A. Grunberg, and M. Phillip, "Md-logic artificial pancreas system: a pilot study in adults with type 1 diabetes," *Diabetes care*, vol. 33, no. 5, pp. 1072–1076, 2010.

[43] R. Nimri, E. Atlas, M. Ajzensztejn, S. Miller, T. Oron, and M. Phillip, "Feasibility study of automated overnight closed-loop glucose control under md-logic artificial pancreas in patients with type 1 diabetes: the dream project," *Diabetes technology & therapeutics*, vol. 14, no. 8, pp. 728–735, 2012.

[44] C. Ziegler, A. Liberman, R. Nimri, I. Muller, S. Klemenčič, N. Bratina, S. Bläsig, K. Remus, M. Phillip, T. Battelino *et al.*, "Reduced worries of hypoglycaemia, high satisfaction, and increased perceived ease of use after experiencing four nights of md-logic artificial pancreas at home (dream4)," *Journal of Diabetes Research*, vol. 2015, 2015.

[45] M. Breton, A. Farret, D. Bruttomesso, S. Anderson, L. Magni, S. Patek, C. Dalla Man, J. Place, S. Demartini, S. Del Favero *et al.*, "Fully integrated artificial pancreas in type 1 diabetes: modular closed-loop glucose control maintains near normoglycemia," *Diabetes*, vol. 61, no. 9, pp. 2230–2237, 2012.

[46] B. Kovatchev, S. Patek, E. Dassau, F. J. Doyle III, L. Magni, G. De Nicolao, C. Cobelli, and J. D. R. F. A. P. Consortium, "Control to range for diabetes: functionality and modular architecture," 2009.

[47] E. Dassau, F. Cameron, H. Lee, B. W. Bequette, H. Zisser, L. Jovanovič, H. P. Chase, D. M. Wilson, B. A. Buckingham, and F. J. Doyle III, "Real-time hypoglycemia prediction suite using continuous glucose monitoring: a safety net for the artificial pancreas," *Diabetes care*, vol. 33, no. 6, pp. 1249–1254, 2010.

[48] M. Tejedor, A. Z. Woldaregay, and F. Godtliebsen, "Reinforcement learning application in diabetes blood glucose control: A systematic review," *Artificial intelligence in medicine*, vol. 104, p. 101836, 2020.

[49] B. P. Kovatchev, M. Breton, C. Dalla Man, and C. Cobelli, "In silico preclinical trials: a proof of concept in closed-loop control of type 1 diabetes," 2009.

[50] C. D. Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, and C. Cobelli, "The uva/padova type 1 diabetes simulator: new features," *Journal of diabetes science and technology*, vol. 8, no. 1, pp. 26–34, 2014.

[51] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering *et al.*, "Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes," *Physiological measurement*, vol. 25, no. 4, p. 905, 2004.

[52] E. Lehmann and T. Deutsch, "A physiological model of glucose-insulin interaction in type 1 diabetes mellitus," *Journal of biomedical engineering*, vol. 14, no. 3, pp. 235–242, 1992.

[53] M. Oroojeni Mohammad Javad, S. Agboola, K. Jethwani, I. Zeid, and S. Kamarthi, "Reinforcement learning algorithm for blood glucose control in diabetic patients," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 57571. American Society of Mechanical Engineers, 2015, p. V014T06A009.

[54] D. J. Luckett, E. B. Laber, A. R. Kahkoska, D. M. Maahs, E. Mayer-Davis, and M. R. Kosorok, "Estimating dynamic treatment regimes in mobile health using v-learning," *Journal of the American Statistical Association*, 2019.

[55] A. Jafar, A. El Fathi, and A. Haidar, "Long-term use of the hybrid artificial pancreas by adjusting carbohydrate ratios and programmed basal rate: A reinforcement learning approach," *Computer Methods and Programs in Biomedicine*, vol. 200, p. 105936, 2021.

[56] B. P. Kovatchev, "Metrics for glycaemic control—from hba1c to continuous glucose monitoring," *Nature Reviews Endocrinology*, vol. 13, no. 7, pp. 425–436, 2017.

[57] T. Zhu, K. Li, P. Herrero, and P. Georgiou, "Basal glucose control in type 1 diabetes using deep reinforcement learning: An in silico validation," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 4, pp. 1223–1232, 2020.

[58] A. Gleave, M. Dennis, S. Legg, S. Russell, and J. Leike, "Quantifying differences in reward functions," *arXiv preprint arXiv:2006.13900*, 2020.

[59] E. Daskalaki, P. Diem, and S. G. Mougiakakou, "Model-free machine learning in biomedicine: Feasibility study in type 1 diabetes," *PloS one*, vol. 11, no. 7, p. e0158722, 2016.

[60] S. Yasini, M. Naghibi-Sistani, and A. Karimpour, "Agent-based simulation for blood glucose control in diabetic patients," *International Journal of Applied Science, Engineering and Technology*, vol. 5, no. 1, pp. 40–49, 2009.

[61] A. Noori, M. A. Sadrnia *et al.*, "Glucose level control using temporal difference methods," in *2017 Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2017, pp. 895–900.

[62] I. Fox, J. Lee, R. Pop-Busui, and J. Wiens, "Deep reinforcement learning for closed-loop blood glucose control," in *Machine Learning for Healthcare Conference*. PMLR, 2020, pp. 508–536.

[63] S. Lee, J. Kim, S. W. Park, S.-M. Jin, and S.-M. Park, "Toward a fully automated artificial pancreas system using a bioinspired reinforcement learning design: In silico validation," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 2, pp. 536–546, 2020.

[64] J. Garcia and F. Fernández, "Safe exploration of state and action spaces in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 515–564, 2012.

[65] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[66] K. Driessens and S. Džeroski, "Integrating guidance into relational reinforcement learning," *Machine Learning*, vol. 57, no. 3, pp. 271–304, 2004.

[67] L. Torrey and M. E. Taylor, "Help an agent out: Student/teacher learning in sequential decision tasks," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-12)*, 2012, pp. 41–48.

[68] C. Gehring and D. Precup, "Smart exploration in reinforcement learning using absolute temporal difference errors," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1037–1044.

[69] D. Kangin and N. Pugeault, "Combination of supervised and reinforcement learning for vision-based autonomous control," in *International Conference on Learning Representations*, 2018.

[70] E. Daskalaki, P. Diem, and S. G. Mougiakakou, "Personalized tuning of a reinforcement learning control algorithm for glucose regulation," in *2013 35th Annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE, 2013, pp. 3487–3490.

[71] C. Cobelli, C. Dalla Man, G. Sparacino, L. Magni, G. De Nicolao, and B. P. Kovatchev, "Diabetes: models, signals, and control," *IEEE reviews in biomedical engineering*, vol. 2, pp. 54–96, 2009.

[72] C. Dalla Man, R. A. Rizza, and C. Cobelli, "Meal simulation model of the glucose-insulin system," *IEEE Transactions on biomedical engineering*, vol. 54, no. 10, pp. 1740–1749, 2007.

[73] C. Dalla Man, D. M. Raimondo, R. A. Rizza, and C. Cobelli, "Gim, simulation software of meal glucose—insulin model," 2007.

[74] R. Visentin, C. Dalla Man, B. Kovatchev, and C. Cobelli, "Incorporating nonlinear response to hypoglycemia into the type 1 diabetes simulator," in *11th Diabetes Technology Meeting*, 2011.

[75] A. Chan, L. Heinemann, S. M. Anderson, M. D. Breton, and B. P. Kovatchev, "Nonlinear metabolic effect of insulin across the blood glucose range in patients with type 1 diabetes mellitus," *Journal of diabetes science and technology*, vol. 4, no. 4, pp. 873–881, 2010.

[76] R. Visentin, C. Dalla Man, B. Kovatchev, and C. Cobelli, "The university of virginia/padova type 1 diabetes simulator matches the glucose traces of a clinical trial," *Diabetes technology & therapeutics*, vol. 16, no. 7, pp. 428–434, 2014.

[77] A. Brazeau, H. Mircescu, K. Desjardins, C. Leroux, I. Strychar, J. Ekoé, and R. Rabasa-Lhoret, "Carbohydrate counting accuracy and blood glucose variability in adults with type 1 diabetes," *Diabetes research and clinical practice*, vol. 99, no. 1, pp. 19–23, 2013.

[78] D. J. Cox, L. Gonder-Frederick, L. Ritterband, W. Clarke, and B. P. Kovatchev, "Prediction of severe hypoglycemia," *Diabetes Care*, vol. 30, no. 6, pp. 1370–1373, 2007.

[79] B. P. Kovatchev, M. Straume, D. J. Cox, and L. S. Farhy, "Risk analysis of blood glucose data: A quantitative approach to optimizing the control of insulin dependent diabetes," *Computational and Mathematical Methods in Medicine*, vol. 3, no. 1, pp. 1–10, 2000.

[80] A. Scaramuzza, V. Cherubini, S. Tumini, R. Bonfanti, P. Buono, F. Cardella, G. d'Annunzio, A. P. Frongia, F. Lombardo, A. C. M. Monciotti *et al.*, "Recommendations for self-monitoring in pediatric diabetes: a consensus statement by the isped," *Acta diabetologica*, vol. 51, no. 2, pp. 173–184, 2014.

[81] A. Facchinetti, S. Del Favero, G. Sparacino, J. R. Castle, W. K. Ward, and C. Cobelli, "Modeling the glucose sensor error," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 3, pp. 620–629, 2013.

[82] C. Ellingsen, E. Dassau, H. Zisser, B. Grosman, M. W. Percival, L. Jovanovič, and F. J. Doyle III, "Safety constraints in an artificial pancreatic β cell: an implementation of model predictive control with insulin on board," *Journal of diabetes science and technology*, vol. 3, no. 3, pp. 536–544, 2009.

[83] M. Schiavon, C. Dalla Man, Y. C. Kudva, A. Basu, and C. Cobelli, "Quantitative estimation of insulin sensitivity in type 1 diabetic subjects wearing a sensor-augmented insulin pump," *Diabetes care*, vol. 37, no. 5, pp. 1216–1223, 2014.

[84] J. Loch and S. Singh, "Using eligibility traces to find the best memoryless policy in partially observable markov decision processes." in *ICML*, vol. 98, 1998, pp. 323–331.

[85] T. J. Perkins, "Reinforcement learning for pomdps based on action values and stochastic optimization," in *AAAI/IAAI*, 2002, pp. 199–204.

[86] Q. Sun, M. V. Jankovic, and S. G. Mougiakakou, "Impact of errors in carbohydrate estimation on control of blood glucose in type 1 diabetes," in *2018 14th Symposium on Neural Networks and Applications (NEUREL)*. IEEE, 2018, pp. 1–5.

[87] Richard S. Sutton. tiles3: Tile coding library. [Online]. Available: http://incompleteideas.net/tiles/tiles3.html

[88] A. D. Rossi, "Rl 4 ap thesis: system evolution gif," 2022. [Online]. Available: https://github.com/Vaeliss/RL_4_AP_Thesis/blob/main/animations/system_evolution_4D_1D.gif

A Appendix

A.1 Hyperparameter optimization

In the tabular setting, considering as features BG concentration, BG rate, and IOB, three main hyperparameters of Sarsa(λ) (learning rate α, type of traces, and the trace decay parameter λ) were optimized for subject 1. Optimization was carried out by looking at a single hyperparameter at the time. For proper comparison, every run had the same random seed.

For the learning rate α, only fixed learning rates were considered, since, ideally, we'd like a

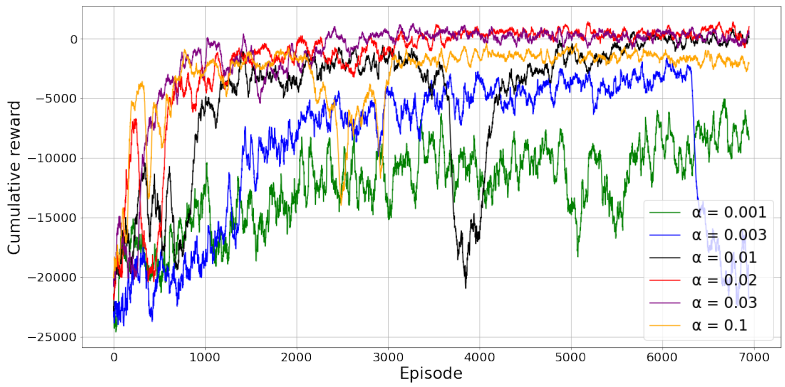

Figure A.1: Cumulative reward throughout training for different values of the learning rate α, all else equal.

RL system in an AP framework to adjust the Q-values estimations and the policy if the habits of the user change over time. A decaying learning rate wouldn't be able to deal with such a non-stationary behavior. Values between 0.001 and 0.1 were tested: Figure A.1 shows the corresponding training processes, keeping track of the cumulative reward throughout the episodes. Low rates (0.001 and 0.003) entail slow learning and are clearly inferior to the others. Conversely, a too large learning rate (0.1), although initially it allows for quick learning, eventually reaches sub-optimal asymptotic results. The remaining three values tested (0.01 to 0.03) are quite close in terms of final performance and speed of learning. Figure A.2 depicts the percentages of time spent in hypoglycemia and hyperglycemia for the three values: arguably 0.02 is the best among the three, being the most stable after the first thousand of episodes, and keeping % time in hypoglycemia low enough.

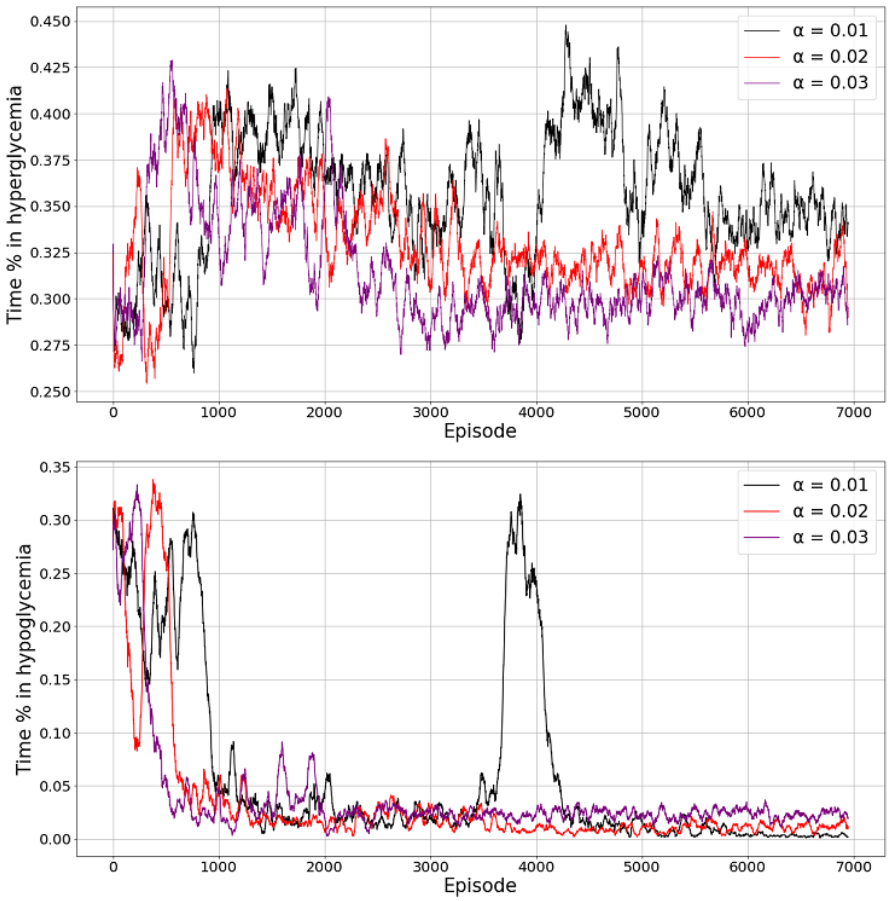

Figure A.2: Time percentages in hyperglycemia and hypoglycemia throughout training for different values of learning rate α, all else equal.

There are different ways to keep track of how fast traces should vanish and how they're updated; in the task considered, the behavior of traces is strictly connected to the amount of time needed for insulin absorption. Accumulating and replacing traces [21] were considered for the optimization process. Multiple values for the trace decay parameter λ were tested for both cases. Training processes and metrics relative to accumulating traces are shown in Figure A.3, while those relative to replacing traces are shown in Figure A.4.

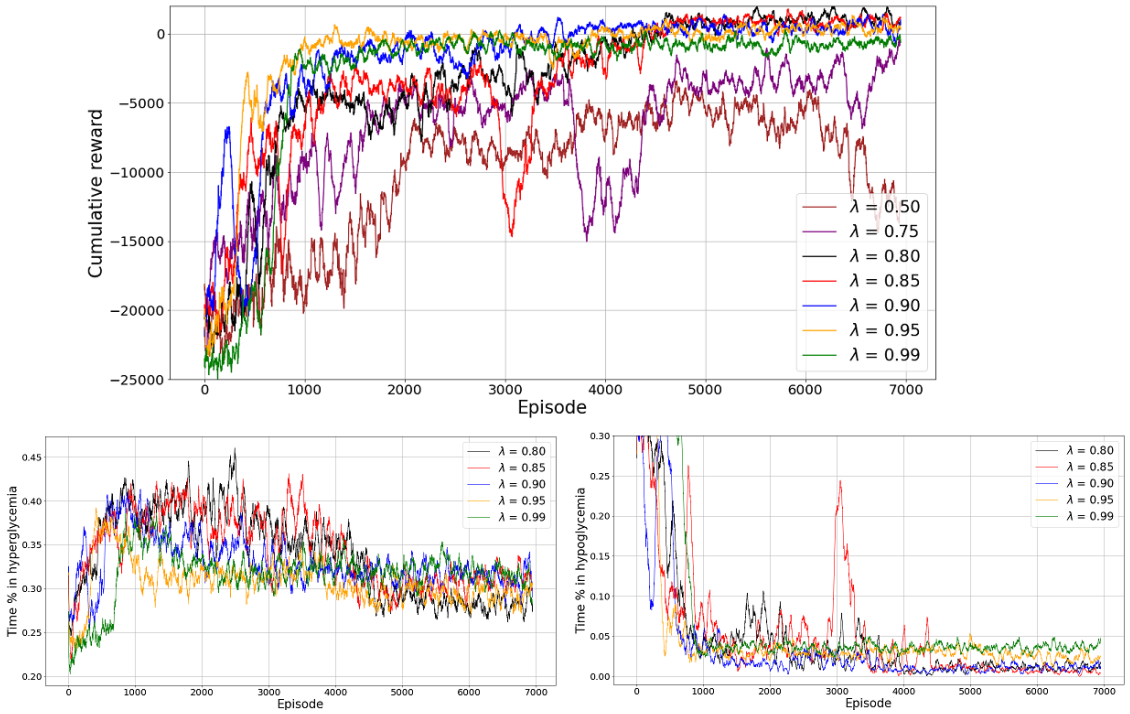

Figure A.3: Cumulative reward and time percentages in hyperglycemia and hypoglycemia, using different values of the trace decay parameter λ and accumulating traces.

For accumulating traces, low values of λ (0.5 to 0.75) lead to slow and sub-optimal learning, as it's possible to see by the first plot. On the other hand, looking at the two following plots, too large λ values (0.95 and 0.99) are associated with larger percentages of time spent in hypoglycemia, compared to the others. Overall values between 0.8 and 0.9 seem to be appropriate.

In the case of replacing traces (Figure A.4), larger λ values are clearly needed to achieve good results: any value below 0.9 doesn't seem to be working properly. This is logic, since the signal relative to the encounter of each (s,a) pair is not accumulated over time, hence for such low λ values the traces would vanish too quickly, compared to the amount of time that insulin has effect in the system. λ values close to 1 (0.95, 0.98, 1) yield the best results in terms both of speed

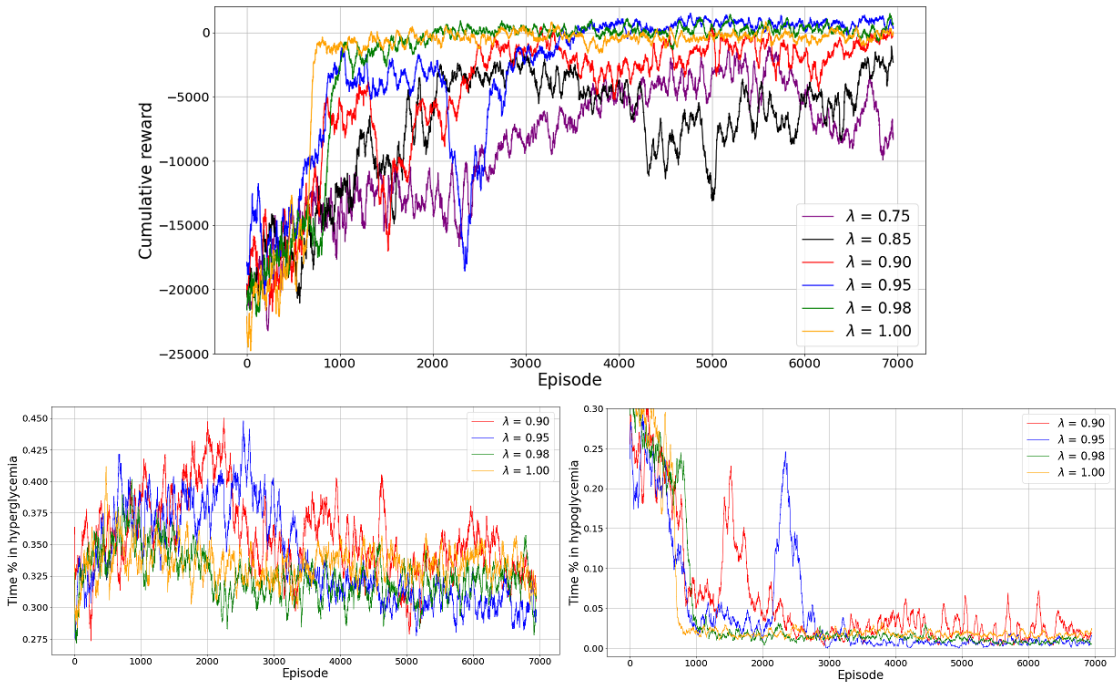

Figure A.4: Cumulative reward and time percentages in hyperglycemia and hypoglycemia, using different values of the trace decay parameter λ and replacing traces.

and overall performance. In case of $\lambda = 1$, vanishing is only determined by the discount factor γ (which has always been set to 0.99). When compared to the accumulating traces case, initial learning seems, in general, to be slightly slower (within the first 500 episodes the cumulative reward is quite poor); however, the best final performances are comparable.

A detailed comparison of the results obtained at convergence for the two types of traces, depending on λ, is presented in Figure A.5.

From the three plots it's clear that larger values of λ are needed for replacing traces and that the window for the selection of a proper λ is narrower than it is in the case of accumulating traces. For the optimal values of λ in the two cases, the final policy reached is quite similar in terms of performance, for all metrics considered. A larger window for selecting the proper trace decay parameter λ, comparable performance, and slightly faster learning appear to indicate that accumulating traces are the best overall choice. In the accumulating traces case, values of λ between 0.8 and 0.9 seem all to be acceptable; looking back at the speed of learning in Figure A.3, among the three, although we're possibly sacrificing a little asymptotic performance, the value 0.9 seems to be the most appropriate, as it's quite clearly leading to faster learning over the other two.

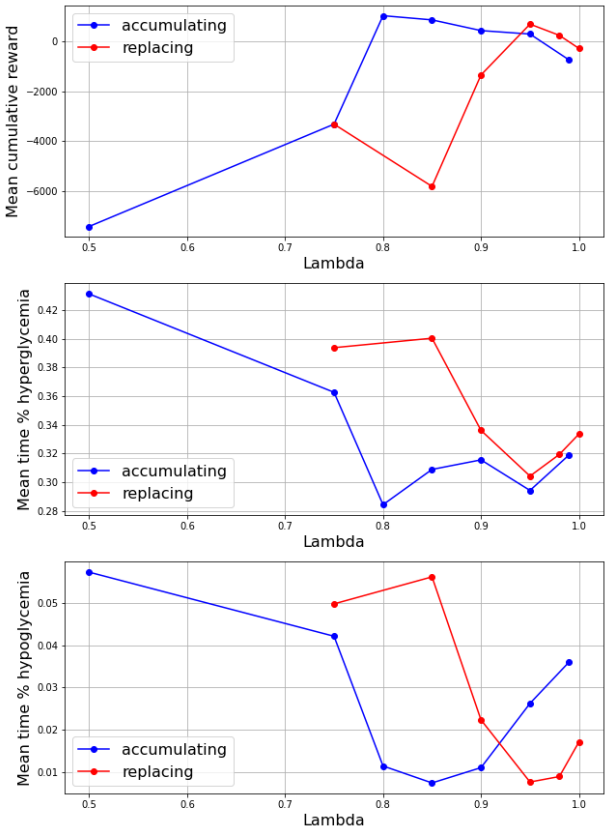

Figure A.5: Comparison of the final results obtained for all the runs, depending on the type of trace and on the trace decay parameter λ.

The generalization of the tabular method to multiple patients, discussed in Section 5.3.1, is carried out by fixing the hyperparameters for Sarsa(λ) to the best-values observed in this section: learning rate $\alpha = 0.02$, accumulating traces, and trace decay parameter $\lambda = 0.9$.

Hyperparameters for the tile coding approach are fixed based on the observations made in this section and according to the guidance provided in the tiles3 library documentation[87].

Acknowledgments

I would like to thank my supervisor Gian Antonio Susto, as well as my co-supervisors Simone Del Favero, Mirco Rampazzo, Eleonora Manzoni, and Francesco Zanini, for their guidance throughout this entire project. Their observations, suggestions, kind support, and constructive feedback have been crucial to the realization of this research work.
Thanks a lot to you all.