
Human-in-the-loop, Interpretable, Sample-efficient Reinforcement Learning for Behaviour Change Interventions

UNDERGRADUATE THESIS

*Submitted in partial fulfillment of the requirements of
BITS F421T Thesis*

By

Advait Rane

ID No. 2017A7PS0135G

Under the supervision of:

Mina Khan

&

Dr. Basabdatta Sen Bhattacharya



BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

December 2020

Declaration of Authorship

I, Advait Rane, declare that this Undergraduate Thesis titled, 'Human-in-the-loop, Interpretable, Sample-efficient Reinforcement Learning for Behaviour Change Interventions' and the work presented in it are my own. I confirm that:

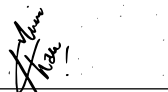
- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 18/12/2020

Certificate

This is to certify that the thesis entitled, “*Human-in-the-loop, Interpretable, Sample-efficient Reinforcement Learning for Behaviour Change Interventions*” and submitted by Advait Rane ID No. 2017A7PS0135G in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.



Supervisor

Mina Khan
PhD Candidate,
MIT Media Lab
Date:

Co-Supervisor

Dr. Basabdatta Sen Bhattacharya
Asst. Professor,
BITS-Pilani Goa Campus
Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

Abstract

Bachelor of Engineering (Hons.)

Human-in-the-loop, Interpretable, Sample-efficient Reinforcement Learning for Behaviour Change Interventions

by Advait Rane

Habits help sustain behaviour change, but forming habits is difficult. Habit formation can be supported with computer-generated interventions. However, for these interventions to be helpful they should be personalised and context-specific. For a healthy interaction with the user the system should learn to adapt to the user fast, and the user should be able to understand and control its behaviour. Thus, sample-efficiency and interpretability are integral to the system. As a part of this thesis, we relied on Reinforcement Learning (RL) to learn the most beneficial interventions. We built an RL environment and designed datasets to test the above desiderata in an RL model. We evaluated the use of different sequence models to learn user behaviour patterns to make learning sample-efficient. We further evaluated the performance of different RL algorithms to determine the best choice in terms of sample efficiency and integration of human guidance. To test the models, we used computer-usage data to learn to give calming interventions during computer usage. We also built and deployed a Chrome extension which gives calming interventions based on the user's preferences while browsing the internet.

Acknowledgements

I would like to thank my supervisor Mina Khan and my co-supervisor Dr. Basabdatta Sen Bhattacharya for their continual support throughout my thesis. I'm very grateful to me fellow collaborator P. Srivatsa without whom this work would not have been possible. I would also like to the the MIT Media Lab for letting me pursue this project remotely in the wake of the COVID-19 pandemic. Lastly, I would like to thank my family who have greatly supported me through this period.

Contents

Declaration of Authorship	i
Certificate	ii
Abstract	iii
Acknowledgements	iv
Contents	v
List of Figures	viii
Abbreviations	ix
1 Introduction	1
1.1 Habit Formation for BCI	1
1.2 Reinforcement Learning	1
1.3 Computer usage	3
2 Literature Review	4
2.1 Behaviour Support and MHealth	4
2.2 Sequence Modelling	5
2.3 Safety in RL	6
3 Reinforcement Learning	7
3.1 RL Environment	7
3.1.1 Input	7
3.1.2 Observation Generation	8
3.1.3 Reward Generation	9
3.1.4 Visualizations	10
3.1.5 Extensions	10
3.2 RL Agents	13
3.2.1 Simple Agent	13
3.2.1.1 Exploration	14

3.2.1.2	Exploitation	14
3.2.1.3	Visualization	15
3.2.2	TensorFlow Agents	15
4	Chrome Extension	17
4.1	Options Page - User Input	17
4.2	Intervention Popup	18
4.2.1	Breathing Patterns	18
4.2.2	Ambient Video	19
4.2.3	Journal	20
4.3	Future Work	20
5	Sequence Modelling	21
5.1	Models	21
5.1.1	Recurrent Neural Networks	21
5.1.2	Convolutional Neural Networks	21
5.1.3	Transformer	22
5.1.4	LSTM Auto-Encoder	22
5.1.5	SeriesNet	22
5.1.6	VQ VAE	23
5.2	Data	23
6	RL Test Cases	25
6.1	Guided Exploration	25
6.2	Personalizing	27
6.3	Multiple Objectives	28
6.4	Interpreting User Guidance	29
6.5	Sensitivity Over Reward Scale	30
7	Results	31
7.1	Reinforcement Learning	31
7.1.1	Guided Exploration	31
7.1.2	Personalizing	31
7.1.3	Interpreting User Guidance	32
7.1.4	Sensitivity Over Reward Scale	32
7.2	Sequence Modelling	32
7.2.1	High Frequency Dataset	33
7.2.2	Different Frequencies	33
7.2.3	Higher Number of States	33
7.2.4	Discontinuity as a Patch	33
7.2.5	Added Noise	33
7.2.6	Discontinuity as Missing Data	34
7.2.7	Linear Trend	34
8	Conclusion	35

A RL Environment Input Files

36

Bibliography

38

List of Figures

1.1	RL setting, shows how the agent interacts with the environment. Source - [25] . . .	2
3.1	RL Environment visualizations	11
3.2	RL Environment Javascript visualizations	12
3.3	Plot of mean value of a Beta distribution over the episodes	16
4.1	PAL extension options page	18
4.2	PAL extension breathing patterns	19
4.3	PAL extension ambient screen	19
4.4	PAL extension journal option	20
5.1	Time series data generated	24
6.1	Matching ideal and user interventions	26
6.2	Different ideal and user interventions	26
6.3	Overlapping ideal and user interventions	27
6.4	Change in intervention	27
6.5	Change in routine	27
6.6	Past routine	28
6.7	Erratic routine	28
6.8	Avoiding user annoyance. X indicates user annoyance.	29
6.9	Limited number of actions. * indicates a time at which the user has limited the number of interventions.	29
A.1	The observations file, which shows 7 episodes with identical routines for 6 time steps. The states are labelled as S1-S5	36
A.2	The objectives file, which shows that the user has an objective to reach state S6 everyday. Since the start time and end time are missing, the state will be attained right after the corresponding action(intervention) is received.	36
A.3	The correct interventions which direct the environment. This file states that the correct action corresponding to objective 1 is between time points 2 and 3.	36
A.4	The user given preferred intervention file, which directs the agents exploration. The user given intervention does not match the correct one, saying that the intervention should be given between time points 0 and 1.	37
A.5	The user rewards file. This indicates that the agent will get a reward of value 1 if the user is in state S6 at time step 3. The reward for each episode has to be listed separately.	37

Abbreviations

RL	R einforcement L earning
ML	M achine L earning
BCI	B ehaviour C hange I nterventions
HIL	H uman I n the L oop
UCB	U pper C onfidence B ounds
CNN	C onvolutional N eural N etwork
RNN	R ecurrent N eural N etwork
AE	A uto- E ncoder

Chapter 1

Introduction

1.1 Habit Formation for BCI

Behaviour change is important for mental and physical well-being. However, changing one's behaviour systematically requires effort and is often difficult to sustain. One can set goals to achieve and bring about behaviour change. However, purely goal-directed behaviour change approaches lead to relapse patterns. While there is higher adherence to the goals in the short-term, retention of the changed behaviour in the long-term is low [31].

Habits, on the other hand, are automatic actions directed by the context[32]. Once habits are formed, they are easier to sustain. Habits can thus lead to higher long-term retention of changed behaviour and habit formation can support behaviour change maintenance[16, 31].

Habit formation can be assisted by computer-generated interventions. However, habits are a highly context-specific and personal actions. For interventions to be beneficial, they should be directed by the context and the user's preferences. They should be given in personalised and fine-grained contexts. Furthermore, the user should have a complete control and understanding of these interventions and their generation. Thus, for a computer to give habit formation interventions, it should take the context and the user's guidance as input and determine the most beneficial interventions to give while adapting to the user fast and allowing the user to interpret and modify its functioning. Sample-efficiency, interpretability, and Human-in-the-Loop learning are thus important desiderata for such a system.

1.2 Reinforcement Learning

Reinforcement Learning is an approach to train an agent to take the most rewarding actions in an environment based on the state of the environment. At each time step the agent receives the

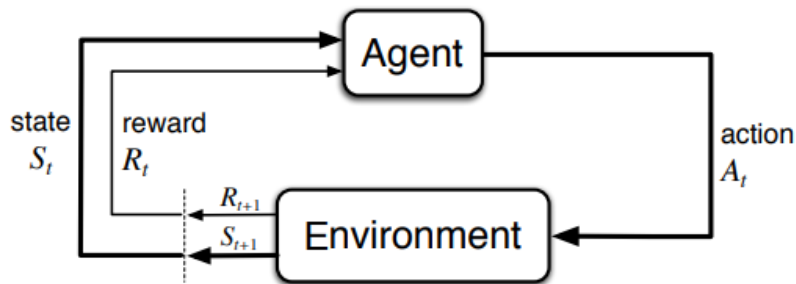


FIGURE 1.1: RL setting, shows how the agent interacts with the environment. Source - [25]

state from the environment and returns an action, following which the environment gives the agent a reward and the next state in the next time step.

It can be formalised as a Markov Decision Problem (MDP), $\langle S, A, R, P, \rho_0 \rangle$. S is a finite set of states of the environment. A is a finite set of actions that the agent can take. R is the reward function which gives the reward at each time step, $R_t = R(s_t, a_t, s_{t+1})$. P is a transition probability function, $P(s, a, s') = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$, which gives the probability of the next state being s' if the action a is taken in the state s . A sequence of states and the corresponding actions is called a trajectory, τ . ρ_0 is a distribution over S for the start state. A discounting factor $\gamma \in [0, 1]$ is used to promote immediate rewards and discount future rewards. The agent aims to maximise the discounted return of the trajectory it takes, $R(\tau) = \sum_{t=0}^{\infty} \gamma^t R_t$. To do this, it learns a policy $\pi(a, s) = P[A_t = a | S_t = s]$ which directs the trajectory. Furthermore, given a policy we can define a value function for each state, $V^\pi(s) = \mathbb{E}_\tau \pi [R(\tau) | s_0 = s]$ and an action-value function (or Q-function) for a state-action pair, $Q^\pi(s, a) = \mathbb{E}_\tau \pi [R(\tau) | s_0 = s, a_0 = a]$. The optimal value function for the optimal policy is given by $V^*(s) = \max_\pi V^\pi(s)$ and the optimal action-value function by $Q^*(s, a) = \max_\pi Q^\pi(s, a)$.

There are several approaches to learn the optimal policy. Broadly, RL algorithms can be divided into model-based and model-free algorithms. Model-based algorithms learn a model of the environment, i.e. the transition function, which can be used for planning as well as sample data generation to learn the policy. These algorithms are usually more sample-efficient. Model-free techniques learn a control policy directly without modelling the environment. Policy learning can be done directly by policy optimisation algorithms which optimise the policy to maximise the value function. It can also be done indirectly by Q-learning which learns the action-value function and chooses the actions which maximise it. Policy optimisation algorithms are usually on-policy, i.e. the trajectory used to train is collected using the recent version of the policy. Q-learning is typically off-policy, i.e. it uses trajectories collected irrespective of the trajectory.

In this thesis, we primarily explored contextual bandits, which have been the state-of-the-art approach for behaviour change interventions, and deep Q-learning. We combined these techniques

with human guidance to give the user control and Bayesian estimation to make the intervention generation simple to interpret. We analysed model-based approaches as sample-efficiency was a key consideration to adapt to the user fast.

1.3 Computer usage

Our computer usage has significantly increased over the last year through the COVID-19 pandemic. Increased computer time, whether due to work or leisure activities, may have become unavoidable for many. However, this increased usage comes with a few challenges. We have had to learn to set boundaries between our work and home lives, and take care of our mental health as stress has increased. We believe that our approach to behaviour change interventions can be especially useful in terms of regulating computer usage.

To this end, we incorporated behaviour change interventions in Chrome browser usage and even evaluated the use of such a system for computer usage in general. We developed a Chrome browser extension through which a user can set micro-interventions for themselves on specific websites. This would enable users to take calming breaks in the middle of work or social media usage on the Chrome browser. We conducted a user survey to understand the kinds of breaks users preferred to take in the middle of computer usage. Furthermore, to extend these interventions to general computer usage, we combined our approach with data obtained from a computer activity tracker called ActivityWatch[2] which allows users to track their computer usage. Our approach would allow users to not only track their computer usage but also set and receive context-specific interventions to take breaks. Thus, the capabilities of our approach to behaviour change interventions to help users build calming habits which alleviate stress and improve mental health become evident in the context of computer usage.

Chapter 2

Literature Review

2.1 Behaviour Support and MHealth

Computers and machine learning have been explored as tools to aid behaviour tracking and behaviour change. Tools like tracking devices and mobile/computer applications have been developed and studied to help users like Quantified-Selfers track their behaviour [6, 5, 13]. These approaches require manual-tracking or self-analysis and do not implement generated interventions. They identify the lack of context-tracking[6] and misalignment of user and system goals[13] as problems hindering behaviour change. They also discuss ways to improve user-engagement to support goal-based behaviour change.

Computer-tailored interventions have been studied for health-behaviour change in specific use cases such as to stop smoking, increasing physical activity, promoting a healthy diet, and receiving regular mammography screening[15]. In the use case of online computer usage, rotating interventions are more effective as opposed to static interventions but they also result in higher attrition[14]. This study also highlights that users prefer to have control over the intervention-generation system. Micro-interventions which direct users to popular web applications to alleviate stress have also been studied[23]. These were augmented with ML-based recommender systems to determine which intervention would be ideal based on user-information such as personality features and contextual-information obtained from phone data.

Reinforcement Learning (RL), and specifically contextual bandits, has been identified as a means to learn to give Just-in-time-adaptive-interventions (JITAI). Contextual bandits-like policies were learned to determine the type of intervention message(positive or negative) to send diabetes patients to increase physical activity based on patient demographics and past activity[34]. RL can be leveraged to learn personalised goals and interventions in mobile fitness and mobile health(MHealth) applications[17, 36]. [36] promotes physical activity by determining daily goals

for number of steps using inverse reinforcement learning. An online, actor-critic, contextual bandit is used to determine whether to give an activity suggestion based on simplified contexts in [17]. Further improvements to RL algorithms in MHealth applications have been suggested, such as sharing information between similar users[38] and initialising the online algorithm with previously collected data[37]. Algorithmic improvements to contextual bandits have also been suggested to determine the effectiveness of generated interventions[33].

Beyond these applications, RL has also been leveraged for interventions in healthcare[35]. RL has been used to determine the most effective Dynamic Treatment Regimes (DTR) also known as adaptive interventions. DTRs, similar to an RL setting, involve a course of actions (such as drug dosage or treatment type) to be taken at a time point based on the patients treatment history and current health.

2.2 Sequence Modelling

Sequence modelling is relevant to this work as a means to model user behaviour pattern sequences. This would allow the algorithm to predict user behaviour with or without interventions. Since the RL agent acts upon the user, sequence modelling could be used to create a model of the environment i.e. the user. We thus have a short review of different types of sequence models. A complete review of sequence modelling literature is beyond the scope of this thesis.

There is a vast literature in the area of sequence modelling. Recurrent Neural Networks (RNNs) can be used to process sequential data in applications like machine translation and speech processing[27, 9]. To learn longer-term dependencies Long-short term memory cells, or LSTMs, can be used[10]. Convolutional Neural Networks (CNNs) have also been used for time-series modelling and text-to-speech translation[24, 22]. These approaches use dilated causal convolutions to increase the receptive fields of CNNs and learn dependencies over longer sequences. State-of-the-art language models use Transformers[30] to model long-term dependencies using attention. Attention models have the added benefit of providing interpretable attention values. Furthermore, transformer training is parallelizable as opposed to RNNs.

More recent sequence modelling algorithms combine different techniques such as attention and Variational Auto-Encoders (VAE). Learning latent representations in sequences can be useful to model the sequence as well as further use those sequences for decision-making. The Vector-Quantised VAE (VQ-VAE)[21] provides a way to learn these underlying representations in videos, speech, etc. The Temporal Difference VAE (TDVAE)[11] provides a means to learn an environment model which learns an abstract state to represent the world. Beyond these

approaches, the Temporal Fusion Transformer (TFT) [19] combines recurrent layers with self-attention layers to learn temporal dependencies at different timescales in an interpretable manner.

2.3 Safety in RL

Safety in the context of RL refers to learning the optimal policy in a safe manner without causing any harm. Safety is an important concept in our work since we are using RL to learn the optimal intervention policy, but it should not learn these at the cost of causing harm to the user’s routine or mental health.

Safe RL has been extensively studied and surveyed in literature [8]. This work highlights the modification of the exploration process by incorporating external knowledge as a way to introduce safety while learning the optimal policy. Safe exploration is a key idea in RL safety, and there are several approaches to inform the exploration process with human advice[28, 20, 29]. RL models also need to be evaluated for safety before training. Safety Gym[1] and AI safety Gridworlds[18] have suites of environments that measure how well agents respect safety constraints such as safe interruptibility and constrained exploration.

Chapter 3

Reinforcement Learning

3.1 RL Environment

Developing the RL environment was the first step we undertook to set up the framework in which to train and evaluate our agents. There are several open-source RL environments such as those in the OpenAI gym[3]. These environments can be used to train and compare different RL algorithms on tasks as simple as the cart-pole problem and as complex as Atari arcade games or 2D/3D robot simulations. However, these environments would not allow us to evaluate the model's behaviour in the real-world behaviour change setting. We thus developed an environment to simulate a user's behaviour based on an input routine and preferred interventions. We developed this environment to work with the OpenAI gym environment interface as well as Tensorflow's tf.agents environment interface so that we could use these interfaces to train RL algorithms. We outline the functioning of the PAL environment in the following subsections.

3.1.1 Input

In the simplest usage, the input to the environment would provide a user's routine, the ideal interventions to bring about a change in behaviour, the interventions suggested by the user as guidance, the user's behaviour change objectives, and the user-given rewards. Each of these inputs is given through a CSV file.

The user's routine is passed as a grid with each episode representing a row and each column representing a unique time step. The content of each cell in the grid represents the activity the user engages in the corresponding episode at the given time step. These activity observations can also be generated or inferred as we will describe later. The intervention files are split into two types. The first kind has interventions as rules, for example, the user stops using their laptop at 5:00PM everyday if they receive an intervention between 4:50PM and 5:00PM while

using a laptop. The second type has interventions as instances, for example, an intervention in the third episode at 10:00AM when the user is browsing Facebook would remind them to take a more fulfilling break. The ideal interventions and the user-given interventions are thus split into two CSV files each. The ideal interventions directs the environment, whereas the user-given interventions would direct the agent's exploration process. Furthermore, the rules can be added, updated, or deleted as the episodes progress. The interventions are linked to the user's behaviour change objectives. These objectives can be specified as rules which indicate the changed behaviour for individual episodes(days) or sets of episodes. These can be positive objectives(for example, take a calming break after using the Mail application everyday) or negative objectives(for example, do not use the Messages application between 5:00PM to 6:00PM on weekends. Finally, the user given rewards are specified as instances at which rewards are to be given to the agent if the user successfully performed the desired activity. Positive rewards promote an activity(for example, reward the agent with +10 if the user takes a calming break at 10:00AM) and negative rewards would lead to avoidance of those activities(for example, reward the agent -10 if the user ignored the intervention and used the Messages app at 5:05PM).

This input is parsed into a suitable form by a data parser script. The data parser converts the input into pandas Dataframes with pre-defined formats so that they can be processed by the environment and agents as needed. Refer to appendix A for an example of the input files.

3.1.2 Observation Generation

The environment simulates a user's behaviour by generating a routine with appropriate behaviour changes and providing the current activity as observations to the agent along with the user-given rewards if any at each time step. The functioning of the environment is divided into two simulators at the lowest level. The output of these simulators is combined to generate per time step observations which are then passed to the environment interface to further pass to the agent.

The routine simulator simulates a routine at the start of an episode which represents the behaviour of the user in the absence of any behaviour change interventions. This routine is sampled from the input routine provided to the environment based on the episode. The routine generation was not done using a generative ML model so as to be a better reflection of actual user behaviour and real-world conditions. While this would impose a limitation on the amount of training data, our model evaluations would be indicative of how the model would actually perform in an active learning setting with limited data for each new user.

The change simulator samples the interventions and their corresponding behaviour changes from the input data based on the episode at the start of each new one. It further divides the changes into three kinds. Time-based changes are those which are directed by a start time and end time of a desired or undesired activity. Context-based changes are those that are directed by

contexts apart from time, e.g. take a break after reading mails irrespective of the time of the day. Quantity-based changes specify the maximum or minimum number of times an activity needs to be done, e.g. go to Twitter no more than three times a day.

The output of these simulators is simulated at the start of each episode. The data simulator then combines these to generate the routine. The data simulator currently implements the time- and context-based changes but not the quantity-based changes. At each time step it generates the current activity (or current state). When it receives an action denoting an intervention, it checks if this intervention produces a change in the routine based on the change simulators output. If so, it makes the corresponding change in the routine simulated by the routine simulator. For positive objectives, the change would involve adding the action into the routine based on the start and end time(time-based) or in relation to another activity(context-based). For negative objectives, the change replaces the undesired activity with the next activity the user would engage in the simulated routine.

The entire process is mediated by a data mediator. The mediator first employs the data parser to pass the input to the simulators. It then runs the simulators to get the state at each time step, which it communicates to the environment interface. It also passes actions from the environment interface to the data simulator. Thus the observations are generated on a per time step basis.

During the implementation of the simulators, I primarily contributed to the data simulator as well as the integration of the different parts.

3.1.3 Reward Generation

The reward generation is done directly in the environment interface. In our current environment structure, the only rewards the agent receives are from the user. The environment does not determine any rewards based on the changes it makes. This has been done so the user has complete control over any rewards that the agent receives. If the rewards were to be simulated inside the environment, the misalignment of this simulator and the users goals would create problems in terms of the effectiveness of the RL as well as its safety.

The user rewards parsed by the data parser are directly passed to the environment interface by the data mediator. The environment gives the agent rewards based on the instances specified in the parsed user rewards. The environment simply has to compare the current observations in terms of the episode, time step, and activity with the desired state in the user rewards and give the reward if the conditions are satisfied. Thus, the rewards are given by the environment but they are solely directed by the user.

3.1.4 Visualizations

We added several visualisations to the environment to help interpreting the agents behaviour based on the input. There are three main visualisations which show the trajectory of the agent in the environment.

The first visualisation shows the entire history for all the trajectories. It displays the state at each time step, accompanied by an action if taken and a reward if received at that time step. The second plot shows only the actions and rewards. This plot helps to identify which actions correspond to which rewards and whether the agent learns to repeat those actions while reducing unwanted actions. The third plot shows the agents actions along with ideal interventions which direct the environment. This plot is helpful to identify the actions the agent learns as compared to the actions it should learn. Figure 3.1 shows the plots for a single run with simple data.

Besides these we also plot the average loss and the returns of the agent across the episodes in the training.

Furthermore, we also developed Javascript visualisations for the first three plots which run on a browser in an interactive manner. This provides an easier way to visualise the episodes, especially for larger episodes with a sparse but meaningful information. The visualisations allow users to zoom into specific time windows to observe the user activities and the agent actions and rewards. Figure 3.2 shows the Javascript visualizations.

3.1.5 Extensions

The environment dynamics described above work well for simple test cases. For example, for small episodes the observations file can be created easily but for longer episodes having a CSV with as many columns as time steps might be difficult. We incorporated extensions to this simple environment structure to provide for more complex dynamics. We added functionality to convert data from different sources into an intermediate observations CSV file which can be passed to the environment.

Activity Watch^[2] is an open-source computer/mobile application which tracks users device usage. It is a time tracker, which essentially tracks the time spent on different applications during computer usage. Since we wanted to apply our system to the case of user computer usage we extended our environment to easily integrate with Activity Watch. Activity Watch provides a JSON file output which logs the users' device usage along with the application name, window title, and time spent. We provide a script to convert this JSON file into an observations CSV file, where each activity is given by the application being used at that time point. This script runs automatically if the user indicates the the source of the observations data is Activity Watch.

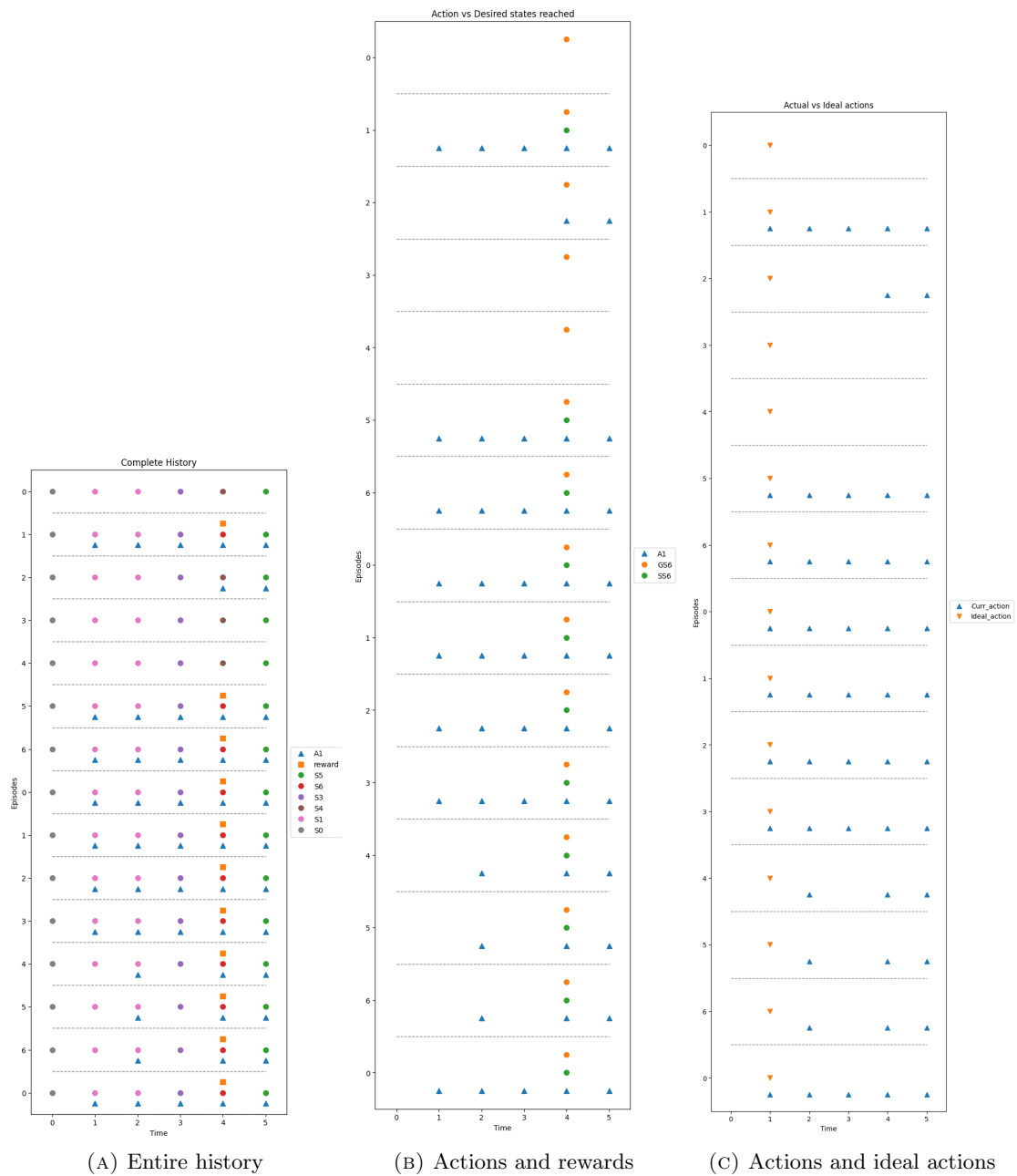
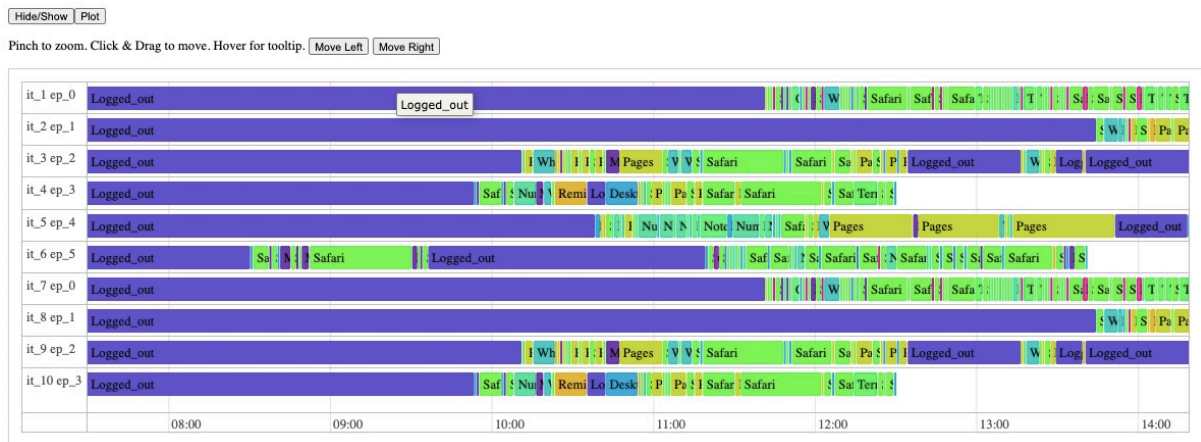


FIGURE 3.1: RL Environment visualizations

States visualization



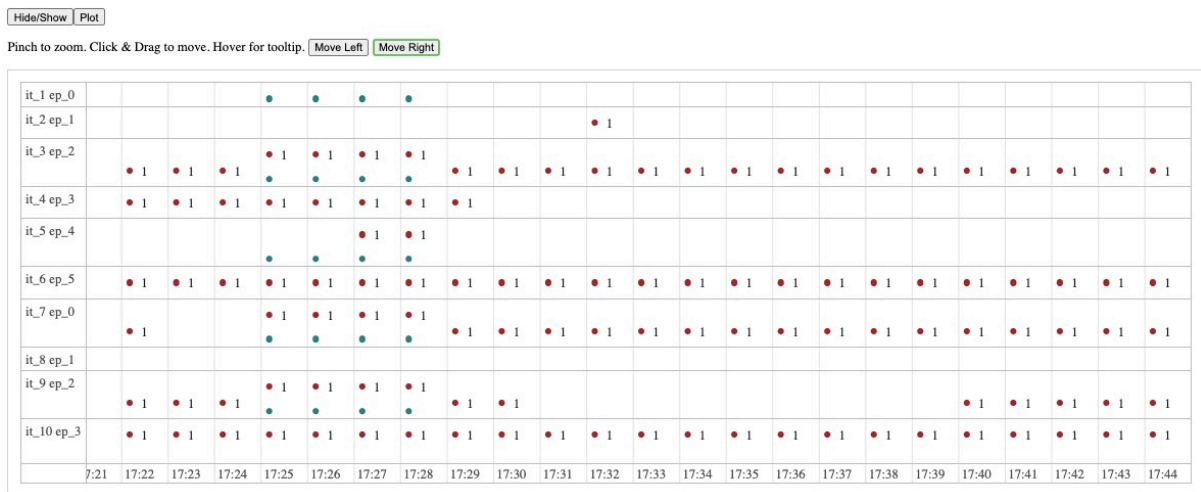
(A) Entire history in Javascript visualizations

Goal states action visualization



(B) Actions and rewards in Javascript visualizations

Agent action visualization



(C) Actions and ideal actions in Javascript visualizations

FIGURE 3.2: RL Environment Javascript visualizations

Besides this, we also added a functionality for the user to generate their own routines. This can generate simple repeating routines by combining sinusoidal waves. This functionality is more for experimental purposes, as it allows us to evaluate the performance of different agents on routines of varying complexities. These routines are generated in sets of episodes combined by concatenating them one after another. For each set, sin waves of different frequency and magnitude are combined by addition. Discontinuity and noise can be added to these generated routines.

The conversion of Activity Watch data conversion was primarily worked on by my colleague.

3.2 RL Agents

We evaluated the performance of a several RL agents on the environment we developed. To begin, we created a simple agent which was primarily guided by the user's preference. To evaluate more complex algorithms we used the TensorFlow Agents library[12]. We integrated our environment with this by wrapping it inside a `tf_agents` environment.

3.2.1 Simple Agent

We first attempted to create a model which was solely directed by the human user. This would ensure alignment with the user's goals with controlled exploration around the user's preferences if they were unsuccessful. We divided the policy into two separate modules, one for exploration and one for exploitation. The exploration module explores around the user's preference and the exploitation module exploits actions which have previously resulted in higher rewards. The functioning of both these modules is similar to probabilistic models. We used a probabilistic approach for these modules for the following reasons-

1. Probabilistic models learn meaningful patterns even in limited data. This is important to deal with sample efficiency
2. Probabilistic models with explicit distributions for different variables are interpretable, through probabilistic measures like credibility intervals and entropy. This allows us to model uncertainty simply.

In the construction of this model, I primarily contributed to the Exploitation module whereas my colleague worked on the exploration module.

3.2.1.1 Exploration

The Exploration module uses the user preferences to represent appropriate intervention time as Gaussian distributions over time. The user preference indicates the interval in which the user would prefer an intervention. The Gaussian is centered at the middle of the interval specified by the user. Since the entire interval should indicate a good intervention, the standard deviation of the Gaussian is set to cover this interval. Therefore, the Gaussians are constructed as,

$$\mu = \frac{start_time + end_time}{2} \quad (3.1)$$

$$\sigma = end_time - start_time \quad (3.2)$$

For each intervention specified by the user, a Gaussian represents the probability of an intervention at that time being successful. These exploitation distributions thus give a high probability for actions at the times specified by the user.

3.2.1.2 Exploitation

The Exploitation module represents every possible intervention as a binomial distribution with a beta prior. A binomial distribution is sufficient since we have just one action, which is to give an intervention. All possible interventions are covered by having beta distributions for an intervention at a time, during an activity, or in an episode. A state is given by a combination of these three. For previously seen states, separate beta distributions are maintained. For unseen states, if any component(activity A, time T, or episode E) of it has been seen before the probability distributions can be combined naively by assuming the components to be independent variables,

$$\mathbf{P}(action = 1|A, T) = \frac{\mathbf{P}(action = 1|A)\mathbf{P}(action = 1|T)}{\mathbf{P}(action = 1)} \quad (3.3)$$

These Beta distributions are initialised with minimally informative or uninformative priors, i.e. Jefferey's prior or Bayes' prior. Probabilistic updates are made to get the posterior from the prior based on the observed rewards. If an action is successful, the first parameter of the corresponding Beta distributions is updated and if it is unsuccessful the second parameter is updated. This the exploitation model can be represented as,

$$\mathbf{P}(action = 1|context) \sim Binomial(p) \quad (3.4)$$

$$p \sim \text{Beta}(\alpha, \beta) \tag{3.5}$$

To update the Beta distributions,

$$\begin{aligned} \alpha_{new} &= \alpha + 1, \text{ if reward} > 0 \\ \beta_{new} &= \beta + 1, \text{ otherwise} \end{aligned}$$

These Beta prior/posterior distributions can be further used to determine the uncertainty of the estimate. We provide two measures to determine the uncertainty,

1. Credibility interval: This gives an interval within which the value of the modelled variable lies with a certain probability. We use a 95 percent credibility interval by default. The smaller the credibility interval, the lower the uncertainty.
2. Entropy: The entropy represents the degree of uncertainty in the value of the modelled variable. The lower the entropy, the lower the uncertainty of our estimate.

These uncertainty measures are important as they are used to choose between the exploration module or exploitation module. We set a threshold on the uncertainty below which we should use the output of the exploitation module. If the exploitation module has a higher uncertainty, we prefer to perform a limited exploration around the user's preferences.

3.2.1.3 Visualization

We added visualizations to these modules to make it easier to interpret the working and evaluate performance. These visualizations plot the different probability distributions along with their mean or standard deviations to indicate the trend in the updated distributions over time. Figure 3.3 shows an example of a plot which shows the value of the mean of a particular Beta distribution over the episodes.

3.2.2 TensorFlow Agents

As mentioned previously, we integrated the environment with TF agents environments. This allowed us to use the different RL agents implemented in the library with our environment. We thus evaluated the performance of several agents for the task.

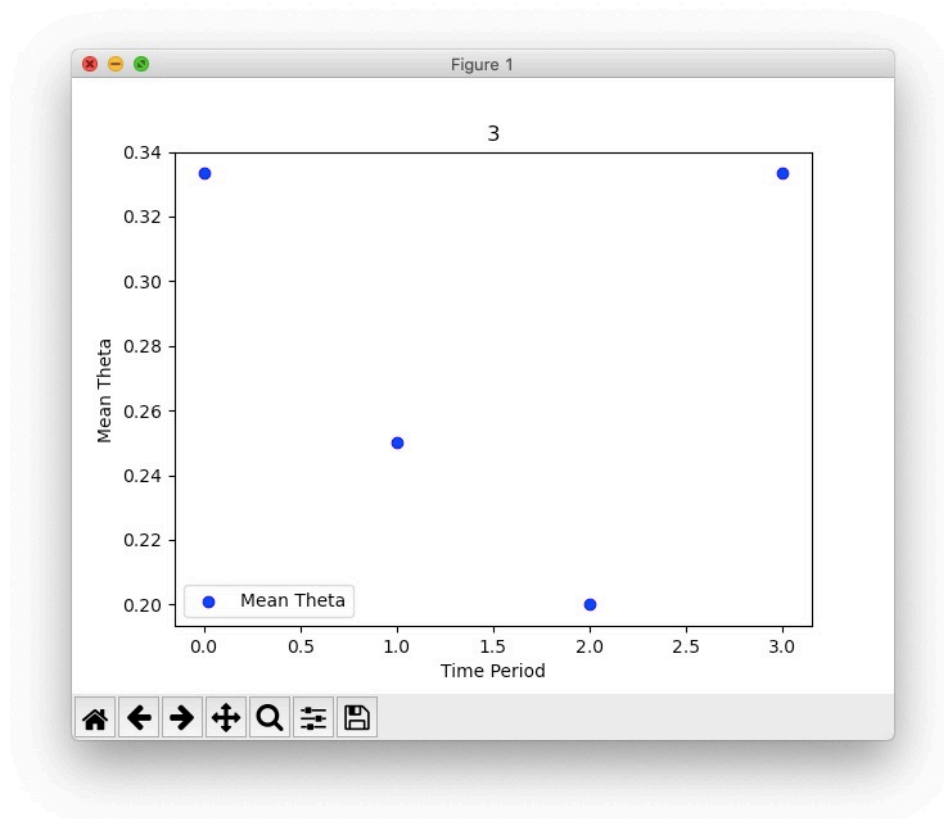


FIGURE 3.3: Plot of mean value of a Beta distribution over the episodes

Since most of the related work in RL for behaviour change interventions used contextual bandits, we evaluated the performance of different contextual bandit algorithms. We trained contextual bandits with Thompson Sampling, Linear UCB, and Neural linear UCB algorithms. Contextual Bandits represent the RL problem as a problem of choosing the correct action based on the context in independent observations. Thompson sampling represents the different actions with Binomial-Beta distributions. The UCB(Upper Confidence Bounds) algorithms use confidence bounds to explore actions which have a lower confidence so as to perform efficient exploration. The UCB algorithms can be combined with a neural network to process the context(observation) in the neural UCB algorithm.

Furthermore, we also evaluated the performance of Q-learning algorithms. We evaluated a simple Deep Q Network(DQN) which represents and learns the Q-function as a neural network. We further evaluated the performance of the DQN C-51(Rainbow) agent which makes several improvements on DQN in traditional RL problems.

Chapter 4

Chrome Extension

We developed a Google Chrome Browser extension to give interventions to users while browsing the internet. Since we identified computer usage as a viable area for our behaviour change interventions, we wanted to develop a system which could collect computer usage data and deliver interventions to users. We decided to develop a browser extension to target the most widely used application of computer usage i.e. browsing the internet. Internet browsing provides a good context for behaviour change interventions as the internet browsing often leads to unhealthy habits [4]. Furthermore, this is also a rich source of data on which we can train and evaluate our models.

The PAL extension has two main objectives,

1. Give the user calming interventions on specific websites as set by the user.
2. Store the user's browsing information (with their permission) on a firebase server.

Since I primarily worked on the first objective, it has been detailed in the following sections. The extension keeps track of the time spent on each open tab. If the user sets a website as an intervention context, the intervention appears on the screen when using that website after a user-specified interval.

4.1 Options Page - User Input

The options page makes for the initial interaction between the user and the extension. Here the user can set which websites the user would like an intervention on. The user also provides an interval after which they would like the intervention. This input might be taken to correspond to the user-given guidance for an RL model.

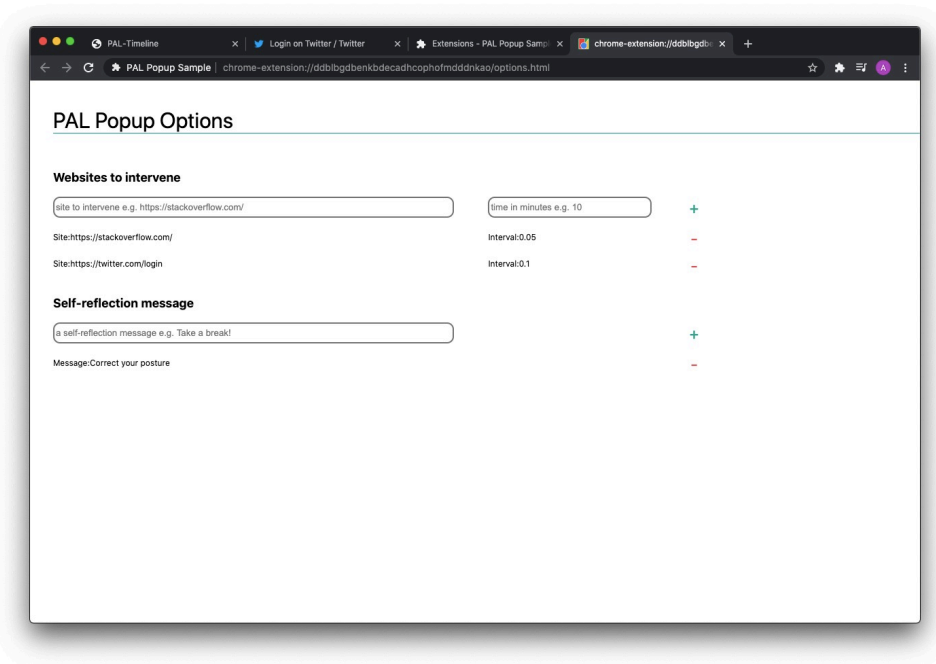


FIGURE 4.1: PAL extension options page

The intervention given by the extension is a calming screen, detailed in the next section. This screen has an accompanying message to the user. The user can customise this message to their preference.

4.2 Intervention Popup

The intervention is introduced as a calming break to the user. This aligns with our goal of alleviating stress while using the computer. The intervention appears as an overlay over the browser screen. It has different calming exercises help the user take a meaningful break.

4.2.1 Breathing Patterns

Controlled breathing helps with alleviating stress. The intervention overlay has an animation which directs the user to control their breathing pattern. The user can select one of the given patterns or customise the breathing pattern. The user can also pause or play the animation video whenever they like.

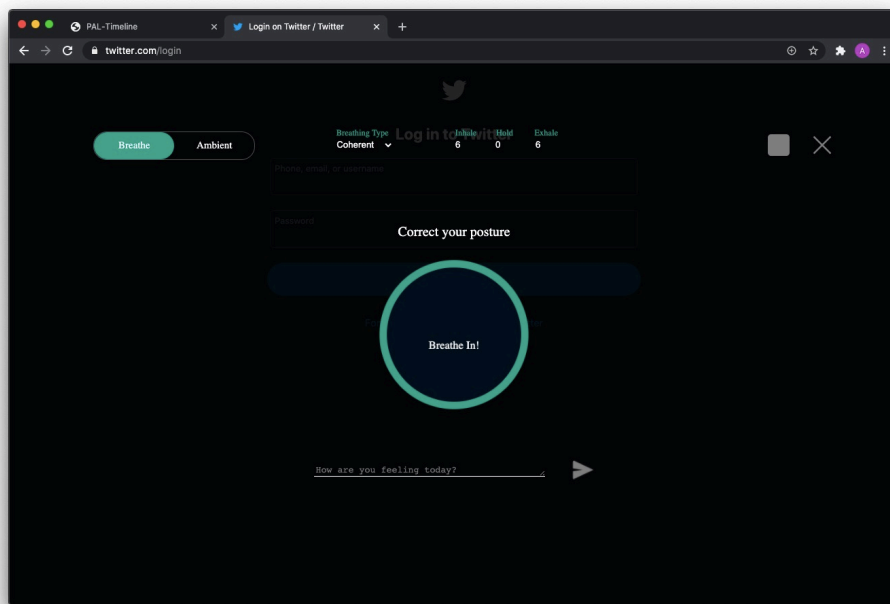


FIGURE 4.2: PAL extension breathing patterns

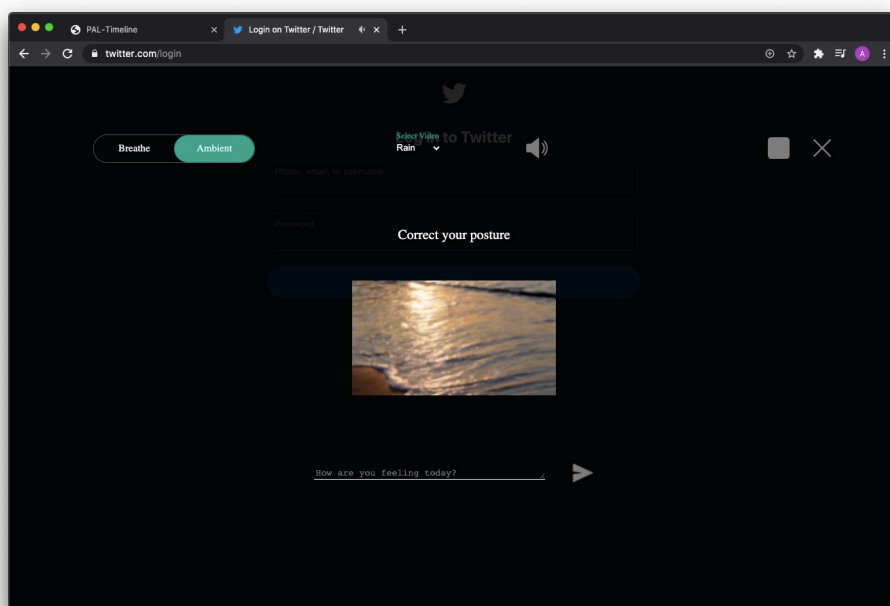


FIGURE 4.3: PAL extension ambient screen

4.2.2 Ambient Video

Alternatively, the user can watch and listen to an ambient video. This allows the user to engage in a more passive break. Currently the intervention has a limited option for the ambient video, but these can be extended in future works. The video too can be paused, played, or muted.

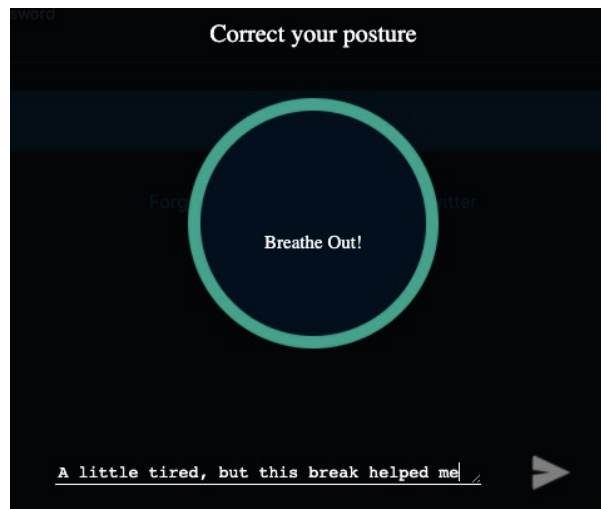


FIGURE 4.4: PAL extension journal option

4.2.3 Journal

The user also has the option to journal during the break. The intervention screen has a text input field which prompts the user to write message about how they feel. This input resembles a text message to induce familiarity. Journaling has been recognised as a healthy habit to cope with stress[26]. Our intervention prompts the user to talk about their emotional state.

4.3 Future Work

The PAL extension, while helpful to give users calming breaks, does not learn to adapt to the user. To add learning capabilities, the extension must be combined with RL agents described in chapter 3. Thus, the calming interventions will receive rewards from the user if they are appropriate and the system will learn to give the user interventions which the user prefers over time.

Furthermore, the Chrome extension only allows us to observe and act on the browser. To tackle computer usage more effectively, the system must be expanded beyond the browser. This can be done by creating a computer application which can give interventions like the extension and store the computer usage data for training the agent.

Chapter 5

Sequence Modelling

5.1 Models

As discussed in chapter 1, model based RL approaches are more sample efficient. The model can be used to learn the environment dynamics and support planning as well as to generate imaginary trajectories which can be used for training in the absence of more data. Our environment represents user behaviour over time. Thus a time series model or sequence model can be used to learn an environment model. We thus evaluated the performance of different sequence models. My colleague and I worked together to evaluate the performance of the different models.

5.1.1 Recurrent Neural Networks

We evaluated the performance of Long-Short Term Memory(LSTM) as well as Gated Recurrent Units(GRU) cells for recurrent networks. We evaluated the performance of small two-layer networks. The first layer has a recurrent connection, either a LSTM or a GRU layer. The second layer is a fully connected layer which predicts the final output. The recurrent layer output has 128 hidden units and the FC layer output has the dimensions of the output. Dropout is applied on the middle layer to avoid over-fitting. The models are optimised with an Adam optimiser. The models are implemented using TensorFlow and Keras.

5.1.2 Convolutional Neural Networks

We evaluated the performance of a small CNN to compare with RNNs. We used a three-layer network with one CNN layer and two fully-connected layers. Since we were training on a time series, the convolutional layers were one-dimensional with a kernel size of 5 and 32 filters. The hidden layer was 128 dimensional. The final layer was used to predict the next two time points

in the series. The dimensionality of the last layer matched the dimensions of the output for two points. The Adam optimiser was used to train the model. It was implemented in TensorFlow and Keras.

5.1.3 Transformer

We trained a vanilla Transformer on time series data to evaluate the performance of simple attention based sequence models. The Transformer implementation uses positional encoding, multi-head self-attention encoder layers, and masked multi-head decoder layers as described in [30], however for a simple implementation we used a single attention head. The Encoder and Decoder both have two layers. The models used 128-dimensional hidden layers and 6-dimensional embeddings for the data. The model is optimised using an Adam optimiser and implemented in TensorFlow and Keras.

5.1.4 LSTM Auto-Encoder

We also trained an Auto-Encoder for sequence-to-sequence modelling to predict the next time steps given the previous observations. We used a small AE with a two-layer encoder and two-layer decoder. The first layer in both the encoder and the decoder is a recurrent LSTM layer. This layer has a 64-dimensional output with a Dropout applied on it. The Auto-Encoder has the additional capability of learning meaningful latent representation in this space which can be decoded to get the output. The AE uses the previous five time steps to predict the next two time steps, similar to the CNN. The models are optimised with an Adam optimiser. The models are implemented using TensorFlow and Keras.

5.1.5 SeriesNet

The SeriesNet is a CNN which is used for time-series data like speech generation. This CNN uses dilated convolutions. These increase the receptive field of the convolutional layer, thus improving the networks capability to model longer-term dependencies.

The model has three identical blocks in series. Each block comprises three one-dimensional convolutional layers. The first dilated CNN layer feed into two separate CNN layers. One CNN is used to get the output of the block. The other is used to get a skip connection to the final output of the model. The output of each block has a residual connection. The dilation of the blocks doubles every layer, which allows the receptive field to grow very large. Each block uses eight convolutional filters. The final output of all the blocks is calculated by a CNN layer on the

concatenated skip connections. Additionally, the SeriesNet also has a single-layer LSTM which processes the input to directly add to the final output.

The model uses the previous 20 time steps to predict the next 20 time steps. It optimised with Adam and implemented in TensorFlow and Keras.

5.1.6 VQ VAE

The VQ VAE is an Auto-Encoder with a Vector Quantization (VQ) layer on the latent representation between the encoder and decoder. This allows us to learn a discrete latent space representation of the time series. The encoder has three one-dimensional convolutional layers which pad the input to get an output of as many time steps as the input. These layers convert the input to a 64-dimensional vector. The encoder then has a residual convolutional layer. The output of the encoder is projected to a discrete number of 64-dimensional vectors by the VQ layer. The VQ layer uses exponential moving averages to update the discrete vectors. The output is further passed to the decoder with three transposed one-dimensional convolutional layers along with a residual convolutional layer after the first transposed layer. This gives the final output of the model. The VQ VAE is trained to take the previous 20 time steps as input and predict the next 5 steps. The Adam optimiser is used for training. The model is implemented using Sonnet[7].

5.2 Data

We evaluated the performance of the aforementioned sequence models on generated time-series data. This data is generated by combining sinusoidal waves in different combinations of number, frequency, and amplitude. Discontinuity, noise, and trends can be added to the generated series. Generated data is used to evaluate the performance of the models on data of varied and controlled complexity.

We used the data generation mechanism to generate seven different kinds of datasets. The simplest way to increase variation is by increasing the frequency of the repeating data. The difference between the frequencies of combined waves can be manipulated to create a time series with long and short term dependencies. The amplitude can be increased to increase the number of states while maintaining the overall frequency of the data. Noise can be added in two ways. The first is to introduce a discontinuity, representing a lack of data. The second is to add small Gaussian noise to each time point. Another form of discontinuity is introduced by incorporating completely missing data, i.e. splicing out portions of the time-series. Finally an overall trend can be added to the time series, such as a linear or quadratic trend. Figure 5.1 shows all the generated timeseries datasets.

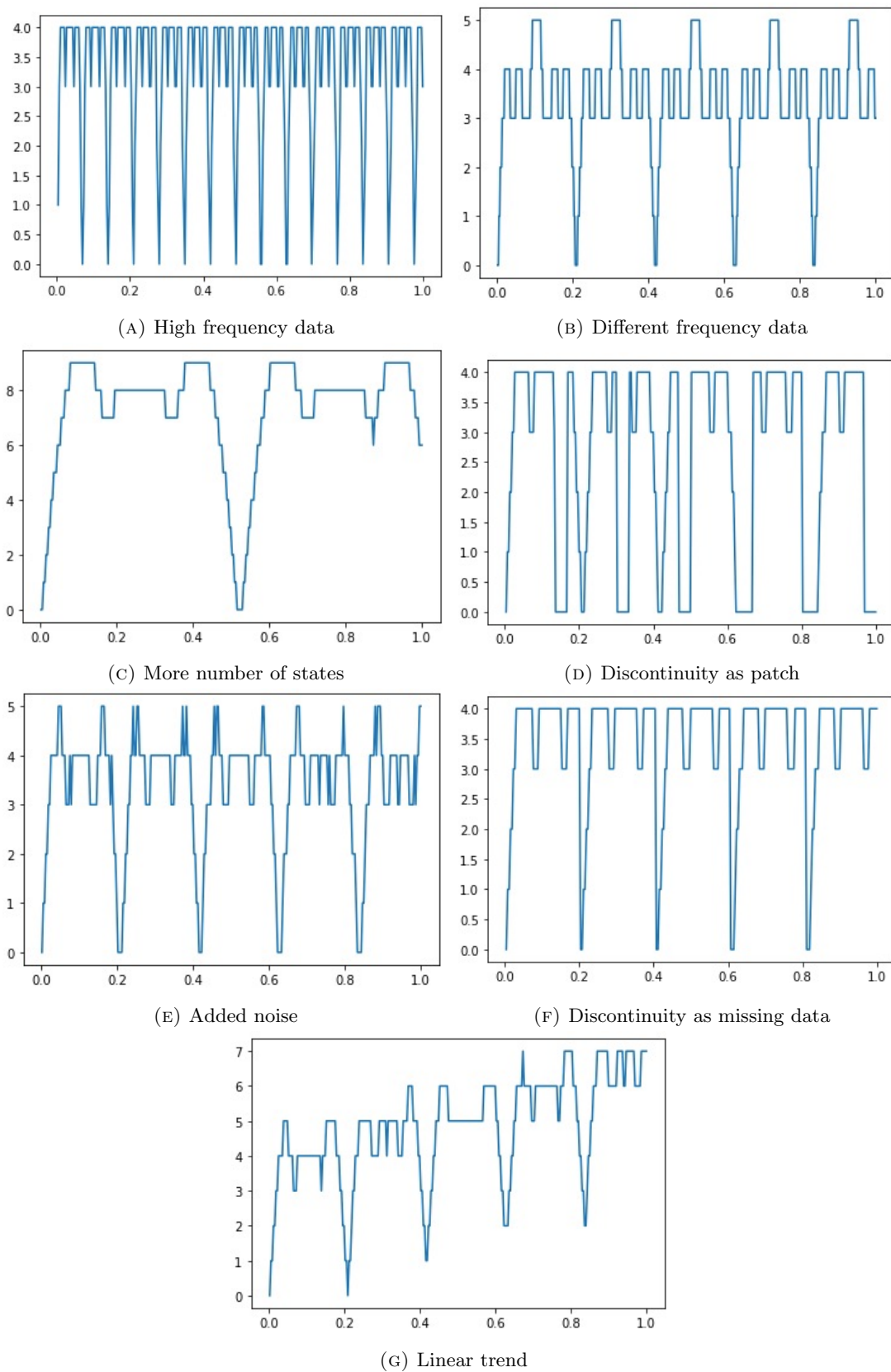


FIGURE 5.1: Time series data generated

Chapter 6

RL Test Cases

Since behaviour change is deeply related to the psyche of the user, we decided to evaluate our solution in terms of its interaction with the user. We drew on literature in RL safety to identify relevant points of consideration. We primarily drew inspiration from the DeepMind safety gridworlds[18], in terms of framing and visualizing our test cases. However, unlike the Gridworlds these test cases are specific to our problem and environment.

We designed simplified versions of real world situations and represented these as a grid for easier interpretation. This grid represents a routine where each row is an episode and each column is a time point. The ideal interventions are represented by highlighted cells. Time based interventions are represented by grey and activity based interventions are represented by matching the colour of the activity. These grids can also be looked as the observations input file passed to the environment, along with additional information. The different tests are described in the following sections. For the purpose of brevity, although we discuss all the test cases we show only those which convey the relevant information.

6.1 Guided Exploration

The primary goal of our agent is to explore safely around the user’s preferred actions to improve on them. However, the agent should only try to improve if the user’s goals are not being met. The primary objectives are those set by the user and not the agent. Safe exploration is a relevant consideration in terms of-

1. Reducing user annoyance due to random exploration with too many actions. Excessive interventions might lead to annoyance and lack of retention

S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5

FIGURE 6.1: Matching ideal and user interventions

S0	S1	S1	S3	S4	S5
S0	S1	S1	S1	S4	S5
S0	S1	S1	S3	S4	S5
S0	S1	S1	S1	S4	S5

FIGURE 6.2: Different ideal and user interventions

2. Avoiding giving interventions which cause the user to enter harmful behaviour patterns. For example, an intervention at a particular time may lead the user to taking an unfulfilling break, say scrolling on social media for too long, and this can affect the user's productivity as well as mental health. These situations need to be avoided.

To test the performance of the agents' exploration we designed situations in the environment where the ideal correct intervention does not match with the intervention given by the user. If the agent avoids exploration it will perform sub-optimally, even receiving negative rewards from the user. Extensive exploration would also lead to sub-optimal performance since the ideal interventions are sparse. The extent of the difference between the ideal intervention and the user given intervention would determine the extent of exploration required. Thus, we designed three test cases with increasing difference.

In the simplest test case the ideal interventions would perfectly align with the user's interventions. In this situation any additional exploration might cause harm. Since the user's goals are being achieved, the agent should avoid further exploration in an attempt to optimise its own objective. Satisfying the user's goals should be the primary objective. This can be seen in figure 6.1.

In the next two test cases, the user given intervention does not match with the ideal correct intervention. The difference between those is greater in the second, requiring greater exploration. These can be seen in figure 6.2.

In the final test case, we evaluate the performance of the agent when it satisfies the user's goals but can explore to learn alternative good interventions. In this case, the agent does not need to explore, but exploration leads to possibly better results. It would be interesting to observe how the agent strikes the balance between satisfying the user's goals and exploring for better results. This can be seen in figure 6.3.

Furthermore, these test cases can also be represented in terms of activity based interventions.

S0	S1	S1	S3	S4	S5
S0	S1	S1	S1	S4	S5

FIGURE 6.3: Overlapping ideal and user interventions

S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5

FIGURE 6.4: Change in intervention

S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S4	S3	S2	S5
S0	S1	S4	S3	S2	S5

FIGURE 6.5: Change in routine

6.2 Personalizing

Habit formation interventions are most useful when they appear in personalised contexts. To improve the user’s experience, our solution must adapt to the user’s personal preferences and behaviour rapidly. Since behaviour patterns are highly dynamic, the agent must be able to keep up with changing behaviour patterns and goals. Our next set of test cases test the agent’s abilities to do this at varying levels of dynamism.

The first few cases evaluate the performance of the agent due to a change in a particular aspect of the user’s behaviour. This can be-

1. A shift in the preferred intervention, for time or activity-based interventions. For example, the user prefers an intervention at 8:00AM on weekdays but 10:00AM on weekends. Refer to figure 6.4.
2. A shift in the user’s routine, while the intervention preferences stay the same. For example, the user likes an intervention at 10:00AM everyday, but on the weekends the user doesn’t have to attend school. Refer to figure 6.5.
3. A combination of these two.

The next few cases evaluate the performance of the agent on a routine which has been seen in the past but not recently. This can be looked as a test of catastrophic forgetting. These changes

S0	S1	S0	S3	S4	S5
S0	S1	S0	S3	S4	S5
S0	S1	S0	S3	S4	S5
S0	S1	S0	S3	S4	S5
S0	S1	S0	S3	S4	S5
S0	S1	S0	S3	S4	S5
S0	S1	S0	S3	S4	S5

FIGURE 6.6: Past routine

S0	S1	S2	S3	S4	S5
S0	S4	S2	S0	S4	S5
S5	S1	S2	S3	S1	S0
S0	S1	S1	S3	S4	S5
S3	S3	S2	S1	S4	S5
S1	S1	S2	S2	S2	S5
S0	S4	S2	S3	S4	S5

FIGURE 6.7: Erratic routine

in the behaviour pattern can also be divided in the above three ways, as well as in terms of time and activity-based interventions. This can be seen in figure 6.6.

Finally, we also evaluate the performance of the agent on a highly dynamic, erratic behaviour pattern. This represents a routine with no fixed pattern or commonality between episodes and the interventions. This can be seen in figure 6.7.

6.3 Multiple Objectives

The user may set multiple simultaneous behaviour change objectives, which might even conflict with each other. Our next set of test cases evaluates the agent performance in such a situation of multiple conflicting objectives. As an example, the user may prefer an intervention at 10:00AM everyday, however on Sundays the user meditates at 10:00AM and prefers not to be disturbed while meditating. In these cases the agent has to prioritise between the different objectives. These cases might also be supported with additional information about the user's prioritisation for these different objectives.

We have two test cases, similar to the example described above. In the first case, the user sets an intervention for a particular state everyday. However, the user gets annoyed by these interventions in some of the episodes. The agent must learn to help the user build the habit while avoiding user annoyance. Refer to figure 6.8.

In the second test case, the user does not get annoyed by specific interventions. Instead, the user sets a limit on the number of interventions that can be given by the agent. Although giving an intervention at a particular time everyday would be the correct action to take for the agent,

S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	X S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	S5
S0	S1	S2	S3	S4	X S5

FIGURE 6.8: Avoiding user annoyance. X indicates user annoyance.

S4	S2	S0*	S0	S1	S5
S4	S2	S0*	S0	S1	S5
S4	S2	S0*	S0	S1	S5
S4	S2	S0*	S0	S1	S5
S4	S2	S0*	S0	S1	S5
S4	S2	S0*	S0	S1	S5
S4	S2	S0*	S0	S1	S5

FIGURE 6.9: Limited number of actions. * indicates a time at which the user has limited the number of interventions.

the user limits the agent's actions to not be reminded everyday. The agent must learn to give the interventions which are most effective while not exceeding the limit. Figure 6.9 shows this test case.

6.4 Interpreting User Guidance

The user can guide the agent in two distinct ways(refer to the input described in chapter 2). These are-

1. By setting rules at the start of usage.
2. By giving guidance at different instances while using.

The performance of the agents while receiving these different forms is evaluated by these test cases. These test cases require the agent to learn a simple intervention rule. The guidance given to the agent can be as a rule for all episodes, a rule for some of the episodes, or at the appropriate time points when the intervention should be received.

These tests are effective if the agent is interpretable. Specifically, if we can determine the agent's confidence for a particular intervention, we can observe how it varies as the episodes progress or as the instance-based guidance is provided.

These test cases can be extended to include different situations. The interventions can be time based or activity based interventions. Additionally, the user guidance can be provided as a combination of rules and instance-based guidance.

6.5 Sensitivity Over Reward Scale

Our last set of test cases evaluates the agent's performance with different scales for the rewards given. In the actual system the user's response indicates the degree to which they liked or disliked a particular intervention. This is a relative scale, which can be assigned absolute values on different scales. The different test cases allow us to compare the agent's performance with smaller reward values versus larger reward values. We use the reward scales used in literature, i.e. reward values of 0.1, 0.3, 1, 3, and 10.

Chapter 7

Results

7.1 Reinforcement Learning

We evaluated the performance of the TensorFlow agents on our test cases described in chapter 6. We did not evaluate the agent performance on the test cases with multiple objectives since the environment does not support alternative objectives, like limited number of actions, yet.

7.1.1 Guided Exploration

The neural UCB bandit performs best on the guided exploration test cases. This conclusion is based on two observations-

1. it performs lesser random exploration, with a lesser number of ineffective actions.
2. it converges to the highest returns fast.

The UCB bandit also performs relatively well, converging to high returns with less random exploration. However, the neural UCB bandit is faster and more efficient. The Thompson sampling bandit performs worse as compared to the UCB bandit. It does not converge to high returns as fast and takes a lot more ineffective actions, continuing to explore even after discovering the ideal actions. The DQN agent's performance is comparable to the TS bandit.

7.1.2 Personalizing

For the personalizing test cases, none of these agents perform well. This can be attributed to the less amount of data. Since there is variation in the routine, the agent gets less data about each pattern.

The Thompson sampling agent performs relatively better than the other agents. However, it continues to explore other actions even after discovering a good action. This behaviour might be harmful. The agents with neural networks, i.e. Neural UCB, DQN, and C51 perform comparably to each other. All of these take too many actions. While this leads to good returns in some cases, it is not desirable. The UCB bandit performance is between that of the TS bandit and the neural network models in terms of taking effective actions. It too takes too many actions, however the loss curve is slightly less random.

7.1.3 Interpreting User Guidance

While we ran different models on these test cases, they don't serve their actual purpose as these agents are not interpretable. We can only evaluate their performance, without evaluating how the agent actually interprets the guidance.

The neural UCB bandit performs best on these test cases. In these test cases, we primarily want to use the user guidance to restrict random exploration. The neural UCB bandit shows the least amount of exploration, while converging to trajectories with good returns. The C51 agent performs almost comparably with training that converges to good trajectories although it appears to be unstable from the loss curves. The Thompson sampling bandit performs too much exploration. The UCB bandit also performs ineffective exploration, but to a lesser extent. The DQN performance is similar to the UCB bandit.

7.1.4 Sensitivity Over Reward Scale

The tests over different scales highlight the fact that exploration is lesser with higher reward values. The TS bandit particularly has markedly lower exploration with higher rewards. The performance of the TS bandit and the UCB bandit is comparable with higher rewards. The neural UCB bandit explores lesser but also overfits to take too many actions. The same is observed with the C51 agent. The DQN agent does not overfit as much as the C51 agent.

7.2 Sequence Modelling

We evaluated the performance of the different sequence models on simple data. The simplicity of the data might be a confounding factor, since we observed that the simpler, smaller models performed better than the more complex models.

7.2.1 High Frequency Dataset

The high frequency data is relatively simple data for the model. All the models perform well on this data. The RNN outperforms the other CNN, LSTM AE, and the SeriesNet marginally. The Transformer and VQ VAE perform relatively worse.

7.2.2 Different Frequencies

The data with different frequencies combined together is also a relatively simple dataset. The CNN, LSTM AE, and SeriesNet perform well on this data, while the RNN is marginally worse. Again, the Transformer and the VQ VAE have the lowest performance.

7.2.3 Higher Number of States

This data results in a marked degradation in performance for the RNN and the SeriesNet as compared to the previous two datasets. The Transformer performance improves slightly, and is comparable to the RNN. The CNN and LSTM AE continue to perform well, while the VQ VAE performance remains poor.

7.2.4 Discontinuity as a Patch

Since the discontinuity is added as a repeating patch of meaningless data, this dataset isn't very different from the simple data in terms of patterns in the data. The disadvantage is that the even if the models perform well, they cannot predict the data which was not observed due to the discontinuity. The CNN continues to perform the best, followed by the LSTM AE and the SeriesNet. The VQ VAE performance is slightly better, making it comparable to the RNN and the Transformer.

7.2.5 Added Noise

With added noise, the CNN performance drops and almost all the models are comparable with this data. The CNN and LSTM AE are only marginally better than the RNN and the Transformer. The SeriesNet performance drops to match the VQ VAE.

7.2.6 Discontinuity as Missing Data

Since the discontinuity is spliced out at regular intervals, this data also isn't very different from the regular or high frequency data. The performance of the different models is the same as is seen in the high frequency dataset.

7.2.7 Linear Trend

An important finding was that none of the models performed well on data with trends. This can be attributed to the training procedure. To mirror an online learning setting we trained the models on the time series in a sequential manner. This meant that for data with trends, it would continually encounter previously unseen states which have never been trained on.

Only the CNN and LSTM AE have a reasonable performance, which is still poor. All the other models have a very poor performance.

Chapter 8

Conclusion

The findings detailed in this thesis highlight that RL agents can be used to augment behaviour change interventions successfully. Drawing on previous literature, contextual bandits are the best algorithms to tackle the problem since they don't overfit to limited amount of data. They learn reasonably well in limited data settings, thus providing a way to incorporate sample efficiency. Human guidance can be used to direct the exploration process by combining it with Bayesian estimation to learn in an interpretable manner. Bayesian prior-posterior methods provide a way to estimate the confidence of the actions which make the model interpretable. Lastly, the bandits can be combined with CNN models to learn a model of the environment. The work detailed in this thesis identifies the best methods to solve different challenges. Future work includes combining these different parts to build a complete system which is sample-efficient and interpretable.

Appendix A

RL Environment Input Files

An example of the input files passed to the RL environment are described here. This test cases represents a simple routine with a time based intervention for the agent to learn. More specifically, this shows the test case in guided exploration when the user-given intervention differs from the correct intervention. There is no instance based user guidance in this example.

S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5
S0	S1	S1	S3	S4	S5

FIGURE A.1: The observations file, which shows 7 episodes with identical routines for 6 time steps. The states are labelled as S1-S5

episode	objective	ranking	day	start time	end time	desired state	before_after_during	neighboring state	desired action	max occurrences	min occurrences	description
0	1	1	all			S6			1			

FIGURE A.2: The objectives file, which shows that the user has an objective to reach state S6 everyday. Since the start time and end time are missing, the state will be attained right after the corresponding action(intervention) is received.

id	add/delete/update	episode	start_time	end_time	state	preference	value	action	goal
0	add	0	2	3		1	1	1	1

FIGURE A.3: The correct interventions which direct the environment. This file states that the correct action corresponding to objective 1 is between time points 2 and 3.

id	add/delete/update	episode	start_time	end_time	state	preference	action	goal
0	add	0	0	1		1	1	1

FIGURE A.4: The user given preferred intervention file, which directs the agents exploration. The user given intervention does not match the correct one, saying that the intervention should be given between time points 0 and 1.

id	add/delete/update	episode	timestep	state	value	action	goal
0	add	0	3	S6	1		
1	add	1	3	S6	1		
2	add	2	3	S6	1		
3	add	3	3	S6	1		
4	add	4	3	S6	1		
5	add	5	3	S6	1		
6	add	6	3	S6	1		

FIGURE A.5: The user rewards file. This indicates that the agent will get a reward of value 1 if the user is in state S6 at time step 3. The reward for each episode has to be listed separately.

Bibliography

- [1] Joshua Achiam and Dario Amodei. “Benchmarking Safe Exploration in Deep Reinforcement Learning”. In: 2019.
- [2] Erik Bjäreholt and Johan Bjäreholt. *ActivityWatch - Open-source time tracker*. <https://activitywatch.net>. [Online; accessed 30-November-2020]. 2017. URL: <https://activitywatch.net>.
- [3] Greg Brockman et al. “OpenAI Gym”. In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: <http://arxiv.org/abs/1606.01540>.
- [4] Hilarie Cash et al. “Internet Addiction: A Brief Summary of Research and Practice”. In: *Current Psychiatry Reviews* 8.4 (Oct. 2012), pp. 292–298. DOI: 10.2174/157340012803520513. URL: <https://doi.org/10.2174/157340012803520513>.
- [5] Eun Kyoung Choe et al. “SleepTight: Low-Burden, Self-Monitoring Technology for Capturing and Reflecting on Sleep Behaviors”. In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp ’15. Osaka, Japan: Association for Computing Machinery, 2015, 121–132. ISBN: 9781450335744. DOI: 10.1145/2750858.2804266. URL: <https://doi.org/10.1145/2750858.2804266>.
- [6] Eun Kyoung Choe et al. “Understanding Quantified-Selfers’ Practices in Collecting and Exploring Personal Data”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, 1143–1152. ISBN: 9781450324731. DOI: 10.1145/2556288.2557372. URL: <https://doi.org/10.1145/2556288.2557372>.
- [7] Deepmind. *deepmind/sonnet: TensorFlow-based neural network library*. <https://github.com/deepmind/sonnet>. [Online; accessed 16-October-2020]. 2018. URL: <https://github.com/deepmind/sonnet>.
- [8] Javier García, Fern, and o Fernández. “A Comprehensive Survey on Safe Reinforcement Learning”. In: *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480. URL: <http://jmlr.org/papers/v16/garcia15a.html>.

- [9] A. Graves et al. “A Novel Connectionist System for Unconstrained Handwriting Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009), pp. 855–868. DOI: 10.1109/TPAMI.2008.137.
- [10] Alex Graves. “Generating Sequences With Recurrent Neural Networks”. In: *CoRR* abs/1308.0850 (2013). arXiv: 1308.0850. URL: <http://arxiv.org/abs/1308.0850>.
- [11] Karol Gregor and Frederic Besse. “Temporal Difference Variational Auto-Encoder”. In: *CoRR* abs/1806.03107 (2018). arXiv: 1806.03107. URL: <http://arxiv.org/abs/1806.03107>.
- [12] Sergio Guadarrama et al. *TF-Agents: A library for Reinforcement Learning in TensorFlow*. <https://github.com/tensorflow/agents>. [Online; accessed 5-November-2020]. 2018. URL: <https://github.com/tensorflow/agents>.
- [13] Rebecca Gulotta et al. “Fostering Engagement with Personal Informatics Systems”. In: *Proceedings of the 2016 ACM Conference on Designing Interactive Systems - DIS '16*. ACM Press, 2016. DOI: 10.1145/2901790.2901803. URL: <https://doi.org/10.1145/2901790.2901803>.
- [14] Geza Kovacs, Zhengxuan Wu, and Michael S. Bernstein. “Rotating Online Behavior Change Interventions Increases Effectiveness But Also Increases Attrition”. In: 2.CSCW (Nov. 2018). DOI: 10.1145/3274364. URL: <https://doi.org/10.1145/3274364>.
- [15] Paul Krebs, James O. Prochaska, and Joseph S. Rossi. “A meta-analysis of computer-tailored interventions for health behavior change”. In: *Preventive Medicine* 51.3-4 (Sept. 2010), pp. 214–221. DOI: 10.1016/j.ypmed.2010.06.004. URL: <https://doi.org/10.1016/j.ypmed.2010.06.004>.
- [16] Dominika Kwasnicka et al. “Theoretical explanations for maintenance of behaviour change: a systematic review of behaviour theories”. en. In: *Health Psychol. Rev.* 10.3 (Sept. 2016), pp. 277–296.
- [17] Huitian Lei, Ambuj Tewari, and Susan A. Murphy. *An Actor-Critic Contextual Bandit Algorithm for Personalized Mobile Health Interventions*. 2017. arXiv: 1706.09090 [stat.ML].
- [18] Jan Leike et al. “AI Safety Gridworlds”. In: *CoRR* abs/1711.09883 (2017). arXiv: 1711.09883. URL: <http://arxiv.org/abs/1711.09883>.
- [19] Bryan Lim et al. *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*. 2020. arXiv: 1912.09363 [stat.ML].
- [20] Richard Maclin et al. “Giving Advice about Preferred Actions to Reinforcement Learners Via Knowledge-Based Kernel Regression.” In: Jan. 2005, pp. 819–824.

- [21] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning”. In: *CoRR* abs/1711.00937 (2017). arXiv: 1711.00937. URL: <http://arxiv.org/abs/1711.00937>.
- [22] Aäron van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: *CoRR* abs/1609.03499 (2016). arXiv: 1609.03499. URL: <http://arxiv.org/abs/1609.03499>.
- [23] Pablo Paredes et al. “PopTherapy: Coping with Stress through Pop-Culture”. In: *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*. PervasiveHealth ’14. Oldenburg, Germany: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, 109–117. ISBN: 9781631900112. DOI: 10.4108/icst.pervasivehealth.2014.255070. URL: <https://doi.org/10.4108/icst.pervasivehealth.2014.255070>.
- [24] Zhipeng Shen et al. “SeriesNet: A Generative Time Series Forecasting Model”. In: *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. IEEE, 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489522. URL: <https://doi.org/10.1109/IJCNN.2018.8489522>.
- [25] Ayush Singh. *Reinforcement Learning : Markov-Decision Process (Part 1)*. <https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da>. [Online; accessed 15-December-2020]. 2019. URL: <https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da>.
- [26] Joshua M Smyth et al. “Online Positive Affect Journaling in the Improvement of Mental Distress and Well-Being in General Medical Patients With Elevated Anxiety Symptoms: A Preliminary Randomized Controlled Trial”. In: *JMIR Mental Health* 5.4 (Dec. 2018), e11290. DOI: 10.2196/11290. URL: <https://doi.org/10.2196/11290>.
- [27] Ilya Sutskever, James Martens, and Geoffrey Hinton. “Generating Text with Recurrent Neural Networks”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML’11. Bellevue, Washington, USA: Omnipress, 2011, 1017–1024. ISBN: 9781450306195.
- [28] Andrea L. Thomaz and Cynthia Breazeal. “Teachable robots: Understanding human teaching behavior to build more effective robot learners”. In: *Artificial Intelligence* 172.6-7 (Apr. 2008), pp. 716–737. DOI: 10.1016/j.artint.2007.09.009. URL: <https://doi.org/10.1016/j.artint.2007.09.009>.
- [29] Lisa Torrey et al. “Using Advice to Transfer Knowledge Acquired in One Reinforcement Learning Task to Another”. In: Oct. 2005, pp. 412–424. ISBN: 978-3-540-29243-2. DOI: 10.1007/11564096_40.
- [30] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: <http://arxiv.org/abs/1706.03762>.

-
- [31] Wendy Wood and David T Neal. “Healthy through habit: Interventions for initiating & maintaining health behavior change”. In: *Behavioral Science & Policy* 2.1 (2016), pp. 71–83.
- [32] Wendy Wood and Dennis Runger. “Psychology of Habit”. en. In: *Annu. Rev. Psychol.* 67 (2016), pp. 289–314.
- [33] Jiayu Yao et al. *Power-Constrained Bandits*. 2020. arXiv: 2004.06230 [cs.LG].
- [34] Elad Yom-Tov et al. “Encouraging Physical Activity in Patients With Diabetes: Intervention Using a Reinforcement Learning System”. In: *Journal of Medical Internet Research* 19 (Oct. 2017), e338. DOI: 10.2196/jmir.7994.
- [35] Chao Yu, Jiming Liu, and Shamim Nemati. “Reinforcement Learning in Healthcare: A Survey”. In: *CoRR* abs/1908.08796 (2019). arXiv: 1908.08796. URL: <http://arxiv.org/abs/1908.08796>.
- [36] Mo Zhou et al. “Personalizing Mobile Fitness Apps using Reinforcement Learning”. In: *Joint Proceedings of the ACM IUI 2018 Workshops co-located with the 23rd ACM Conference on Intelligent User Interfaces (ACM IUI 2018), Tokyo, Japan, March 11, 2018*. Ed. by Alan Said and Takanori Komatsu. Vol. 2068. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: <http://ceur-ws.org/Vol-2068/humanize7.pdf>.
- [37] Feiyun Zhu and Peng Liao. “Effective Warm Start for the Online Actor-Critic Reinforcement Learning based mHealth Intervention”. In: *CoRR* abs/1704.04866 (2017). arXiv: 1704.04866. URL: <http://arxiv.org/abs/1704.04866>.
- [38] Feiyun Zhu et al. “Group-driven Reinforcement Learning for Personalized mHealth Intervention”. In: *CoRR* abs/1708.04001 (2017). arXiv: 1708.04001. URL: <http://arxiv.org/abs/1708.04001>.