Learning in the presence of partial observability and concept drifts.

Raihan Seraj

Master of Engineering
Electrical and Computer Engineering
McGill University
Montreal, Canada

April 2019

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering.

2019/04/12

To my parents

ACKNOWLEDGEMENTS

First of all I would sincerely like to thank my supervisor Aditya Mahajan, who helped me and guided me throughout my entire research work. I am grateful for his time in explaining me the core concepts and patiently reading my thesis and providing me useful insights and corrections that helped me to understand things in detail. His continuous guidance has helped me a lot in terms of research and writing. I would also like to thank my lab mate Jayakumar Subramanian for helping me to better understand python and explaining me some of the important concepts whenever I had trouble understanding them. Additionally I would like to thank my other lab mates Mohammad Afshari, Borna Syedana, Nima Akbarzadeh and Samin Yaser Arnob for keeping me in good spirit. I would also like to thank my friend Riashat, Tasnova as life in Montreal would be really difficult without them. I like to thank my friends Rashik, Mostofa, Mahir, Sadat, Imtiaz, and Martin back in Bangladesh for supporting me continuously.

I would like to express my gratitude to Negar Ghourchian, Michel Allegue and the wonderful R&D team at Aerial Technologies. They have helped me to understand how machine learning techniques can be applied to real world data and guided me throughout the entire process of data collection and research at Aerial, and it was a pleasure working with this wonderful team.

Finally I would like to thank my wonderful family, my parents, my sister Rezwana who has been supporting me continuously throughout my entire journey at graduate school. Their unconditional love and encouragement have always kept me motivated in doing things that I love doing.

ABSTRACT

This thesis is divided in two parts where we analyse machine learning algorithms in two different contexts. In the first part we study reinforcement learning algorithms, specially policy gradient methods for partially observable Markov decision processes (POMDPs). We use a special class of policy structure represented by a finite state controller. A comparison of different policy based methods are made and the performance of these methods are compared with the existing planning solutions for different POMDP environments used in the literature. The second part of this thesis outlines the problem of concept drifts for time series data where we analyze the temporal inconsistency of streaming wireless signals in the context of device-free passive indoor localization. We highlight that concept drifts play a major role in deteriorating the predictive accuracy of models trained for room level localization with WiFi signals and propose a phase and magnitude augmented feature space that is resistant to drifts. We study different learning algorithms with this new feature space and compare their performance in the presence of drifts.

ABRÉGÉ

Cette thse est divise en deux parties o nous analysons des algorithmes dapprentissage automatique dans deux contextes diffrents. Dans la premire partie, nous tudions les algorithmes d'apprentissage par renforcement, en particulier les mthodes de gradient de politique pour les processus de dcision de Markov partiellement observables (POMDP). Nous utilisons une classe spciale de structure de rgles reprsente par un contrleur d'tats finis. Une comparaison de diffrentes mthodes bases sur des rgles est effectue et les performances de ces mthodes sont compares aux solutions de planification existantes pour diffrents environnements POMDP utiliss dans la littrature. La deuxime partie de cette thse dcrit le problme des drives conceptuelles pour les donnes de sries chronologiques, o nous analysons lincohrence temporelle de la transmission en continu de signaux sans fil dans le contexte de la localisation intrieure passive sans appareil. Nous soulignons que les drives conceptuelles jouent un rle majeur dans la dtrioration de la prcision prdictive des modles forms pour la localisation au niveau de la pice avec des signaux WiFi et proposons un espace de fonctions augmentation de phase et de magnitude rsistant aux drives. Nous tudions diffrents algorithmes d'apprentissage avec ce nouvel espace de fonctions et comparons leurs performances en prsence de drives.

Contents

List of Figures

List of Tables

List of Acronyms

POMDP	Partially Observable Markov Decision Process
MDP	Markov Decision Process
FSC	Finite State Controllers
RL	Reinforcement Learning
CSI	Channel State Informationl

Chapter 1

Introduction

This thesis consists of two parts which we explain below.

1.1 Part 1– RL for POMDPs

The first part of this thesis focuses on Reinforcement learning (RL) algorithms for partially observable Markov decision processes (POMDPs). POMDPs are a class of mathematical models for sequential decision making where the decision maker (or agent) has imperfect information about its current state. Reinforcement learning is a class of algorithms used by agents to act optimally in an unknown environment by adapting their actions based on the observed rewards. Reinforcement learning theory and practice has achieved considerable progress in recent years [1, 2]. However, most of this progress is restricted to models where the agent perfectly observes the state of the environment. Learning in environments with partial state observation is not well understood. Understanding reinforcement learning for POMDPs is critical to understand learning in more complicated partially observed models such as partially observed semi-Markov models and multi-agent systems. In real world scenarios sequential decision problems are often formulated as a POMDP. Even if the dynamics of the agent's environment and observations are known, finding an optimal solution of POMDPs is NP hard [3]. It has been established that the computation complexity of solving a finite horizon POMDP is PSPACE-complete [4] and undecidable for infinite horizon POMDP [5]. For such problems reinforcement learning methods are an attractive option for finding a locally optimal solution.

In this thesis we perform an empirical evaluation of various RL algorithms for POMDPs

proposed in the literature. In chapter 2 we give a background literature on POMDPs and RL for POMDPs. In chapter 3 we describe the commonly used RL algorithms for POMDPs and and in chapter 4 we evaluate them empirically on some representative models and analyse their performance.

1.2 Part 2–Handling concept drifts for WiFi CSI

In the second part of this thesis, we study concept drifts for time series data. Machine learning algorithms for time series data are used with the underlying assumption that the statistical properties of the time series are stationary. Concept drift refers to the situation when such assumption is violated, i.e., when the underlying data distribution changes over time in an unforeseeable manner. Most of the techniques to tackle concept drifts in the literature are application specific, thus for this part of the thesis we focus on tackling concept drifts for the application of determining room level locations using WiFi signals. In the presence of drifts predictive models need frequent retraining by incorporating the latest observations. For supervised learning, depending on application domains, obtaining labelled examples for retraining can be quite expensive.

Indoor localization using WiFi signals have gained much attention over time since essential information characterizing signal propagation can obtained easily from off-the-shelf WiFi enabled devices. Passive indoor localization with WiFi signals is particularly attractive since it does not require the user to carry WiFi enabled target devices all the time in order to determine their location. One of the challenging tasks for localization with WiFi channel state information (CSI) is that they broadly suffer from concept drifts. This was first addressed in [6] where semi supervised learning is used to detect gradual shifts in the feature space and an adaptive learning strategy is proposed to regain the prediction accuracy. However, existing literatures on WiFi indoor localization largely focus on different methods for active localization rather than the passive one. In either case, surprisingly analysis of concept drifts for such applications or the proposition of adaptive learning strategies have not been made earlier, which motivated the second part of this thesis. We show how concept drifts cause a shift in the feature space for WiFi CSI and propose an augmented feature space that is less affected by drifts. We show the performance of different learning algorithms in this augmented feature space and propose a many to one sequence modelling for RNN with an LSTM module that captures the variation of CSI with time. In chapter 5

we provide technical background on WiFi channel state information and indoor localization and perform a magnitude and phase analysis for WiFi CSI. We discuss different types of concept drifts and features that are important for localization and propose the augmented feature space. In chapter 6 we present different learning algorithms that are trained with the proposed feature space and tested on drifted signals.

Chapter 2

Background on RL for POMDPs

In this section we discuss the model formulation of MDP and a POMDP and outline the difference between. We provide a background on reinforcement learning for MDPs and discuss how RL algorithms can be applied to POMDPs.

2.1 Markov decision process

An infinite horizon Markov decision process is represented by a tuple $\langle \mathcal{X}, \mathcal{U}, P, r, \gamma \rangle$ where

- $\mathcal{X}$ is the set of states. The state at time t is denoted by X_t.
- $\mathcal{U}$ is the set of actions. The action at time t is denoted by U_t.
- P is the conditional transition probability that governs the evolution of the system. In particular for any realization $x_{1:t}$ and $u_{1:t}$, we have

  $$\Pr(X_{t+1} = x_{t+1} | X_{1:t} = x_{1:t}, U_{1:t} = u_{1:t}) = \Pr(X_{t+1} = x_{t+1} | X_t = x_t, U_t = u_t) = P(x_{t+1} | x_t, u_t).$$

- $r(x, u)$ represents the per-step reward of taking action u at state x.
- γ is the discount factor where $\gamma \in (0, 1)$ used to geometrically discount future rewards.

Given a MDP, a policy $\mathbf{g} = (g_1, g_2, \dots)$ is a collection of functions used to generate actions i.e.,

$$U_t = g_t(X_{1:t}, U_{1:t-1}). \tag{2.1}$$

The performance of any policy is given by

$$J(x;\mathbf{g}) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} r(X_t, U_t) \;\Big|\; X_1 = x\Big]. \tag{2.2}$$

The objective is to find a policy $\mathbf{g}$ that maximizes $J(g)$

The main results for MDPs [7] are as follows:

1. There is no loss of optimality in restricting attention to time homogeneous Markov policies, i.e., policies of the form $\mathbf{g} = (g, g, \dots)$ where $g : \mathcal{X} \to \mathcal{U}$.

2. Let V be the unique bounded fixed point of the following system of equations:

$$V(x) = \max_{u \in \mathcal{U}} \Big[r(x,u) + \sum_{x' \in \mathcal{X}} P(x'|x,u) V(x') \Big], \quad x \in \mathcal{X}. \tag{2.3}$$

   Then for any time homogeneous Markov policy $\mathbf{g} = (g, g, \dots)$, $J(\mathbf{g}) \le V$ with equality if and only if

$$\mathbf{g}(x) \in \arg\max_{u \in \mathcal{U}} \Big[r(x,u) + \sum_{x' \in \mathcal{X}} P(x'|x,u) V(x) \Big], \quad x \in \mathcal{X}. \tag{2.4}$$

2.2 Partially observable Markov decision process (POMDP)

An infinite horizon partially observable Markov decision process is represented by a tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{O}, P, \Omega, r, \gamma \rangle$ where

- $\mathcal{X}$ is the set of states. The state at time t is denoted by X_t.
- $\mathcal{U}$ is the set of actions. The action at time t is denoted by U_t.
- $\mathcal{O}$ is the set of observations. The observation at time t is denoted by O_t.
- P is the conditional transition probability that governs the evolution of the system. In particular for any realization $x_{1:t}$ and $u_{1:t}$, we have

$$\begin{aligned} \Pr(X_{t+1} = x_{t+1} | X_{1:t} = x_{1:t}, U_{1:t} = u_{1:t}) &= \Pr(X_{t+1} = x_{t+1} | X_t = x_t, U_t = u_t) \\ &= P(x_{t+1} | x_t, u_t). \end{aligned}$$

- Ω is the conditional observation probability that governs the observation made by the agent when the system transitions to a new state following an action.

$$\begin{aligned}\Pr(O_{t+1} = o_{t+1}|X_{1:t+1} = x_{1:t+1}, U_{1:t} = u_{1:t}) &= \Pr(O_{t+1} = o_{t+1}|X_{t+1} = x_{t+1}, U_t = u_t) \\ &= \Omega(o_{t+1}|x_{t+1}, u_t).\end{aligned}$$

Given a POMDP, a policy $\mathbf{g} = (g_1, g_2, \dots)$ is a collection of functions used to generate actions, i.e.,

$$U_t = \mathbf{g}_t(O_{1:t}, U_{1:t-1}). \tag{2.5}$$

For ease of notation we define $I_t = \{O_{1:t}, U_{1:t-1}\}$ where I_t is the information available to the agent at time t, and write

$$U_t = g_t(I_t). \tag{2.6}$$

The performance of any policy is given by

$$J(i; \mathbf{g}) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} r(X_t, U_t)|I_1 = i\Big]. \tag{2.7}$$

The objective is to find a policy $\mathbf{g}$ that maximizes $J(g)$. The main results of a POMDP are as follows:

1. Policy is a function of history which grows with time, i.e., $\mathbf{g} = (g_1, g_2, \dots)$ where $g : I \to \mathcal{U}$. A belief state which is a function of information (also known as the information state or sufficient statistic for control) is a probability distribution over the states of the system. Hence the policy function can be mapped from the belief state to actions. $g : \pi \to \mathcal{U}$.

2. Dynamic programming decomposition for POMDPs, can be performed with the belief state as the information state:

$$V(\pi) = \max_{u \in \mathcal{U}} [r(x, u) + \gamma \sum_{o \in \mathcal{O}} P(o|\pi, u) V(\phi(\pi, u, o))]. \tag{2.8}$$

The optimal policy in terms of the belief state is written as

$$\mathbf{g}(\pi) = \arg\max_{u \in \mathcal{U}} [r(x,u) + \gamma \sum_{o \in \mathcal{O}} P(o|\pi,u) V(\phi(\pi,u,o))]. \tag{2.9}$$

Here, $\phi(\pi,u,o)$ is the next belief state that is recursively computed using Bayes rule when an agent receives an observation after taking an action. The belief state is updated as:

$$\begin{aligned}
&\pi_{t+1}(x_{t+1}) = P(X_{t+1} = x_{t+1} | O_{1:t+1} = o_{1:t+1}, U_{1:t} = u_{1:t}) \\
&\pi_{t+1} = P(x_{t+1} | o_t, u_t, \pi_t) \\
&\overset{(a)}{=} \frac{P(o_t | x_{t+1}, u_t, \pi_t) \cdot P(x_{t+1} | u_t, \pi_t)}{P(o_t | u_t, \pi_t)} \\
&\overset{(b)}{=} \frac{P(o_t | x_{t+1}, u_t) \cdot \sum_{x \in \mathcal{X}} [P(x_{t+1} | u_t, \pi_t, x_t) \cdot P(x_t | u_t, \pi_t)]}{P(o_t | u_t, \pi_t)} \\
&\overset{(c)}{=} \frac{F(x_{t+1}, o_t, u_t) \cdot \sum_{x \in \mathcal{X}} P(x_{t+1}, u_t, x_t) \cdot \pi_t(x_t)}{P(o_t | u_t, \pi_t)} \\
&=: \phi(\pi_t, u_t, o_t)(x_t),
\end{aligned} \tag{2.10}$$

where (a) is obtained from Bayes rule, (b) follows from (a) using the law of total probability. F in equation (c) is the observation function $F : \mathcal{X} \times \mathcal{U} \to \Pi(\mathcal{O})$ which is given by

$$F(x_t, u_t, o_t) := P(O_t = o_t \mid X_t = x_t, U_t = u_t)$$

.

For a POMDP with n number of states, the belief space is a subset of $\mathbb{R}^n$. Thus, performing value iteration for continuous states becomes intractable. A fundamental fact about the finite horizon value function is that it is piecewise linear and convex when we

are maximizing the reward (or concave when we are minimizing the cost) for every horizon length [8]. This means that value function over a finite horizon can be characterized by finite number of hyper planes when the belief space is multidimensional.

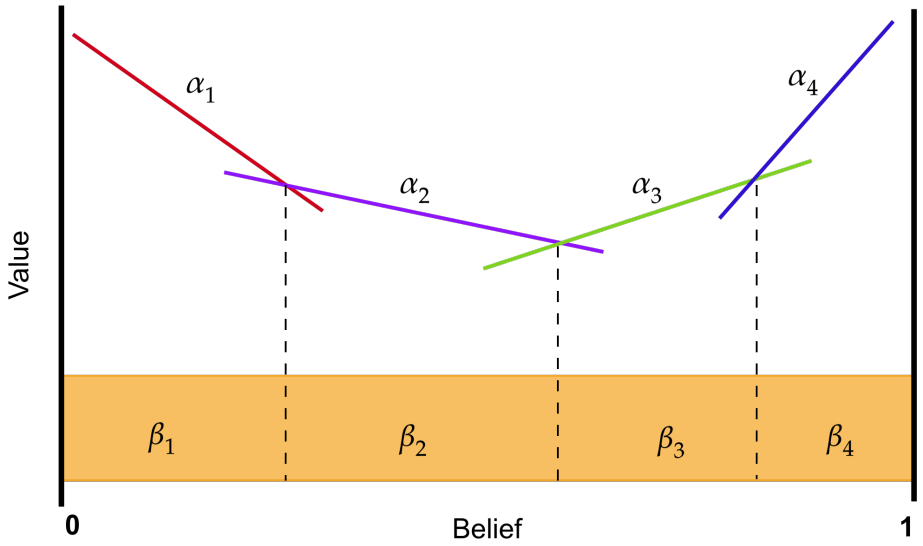


Fig. 2.1 Piecewise-linear characteristic of the value function over a finite horizon N

The linear segments can be represented by a set of $|\mathcal{X}|$ dimensional vectors known as the α vectors given by $\Gamma = \{\alpha_1, \alpha_2 \ldots \alpha_m\}$. The value function at any particular belief point is computed by taking maximum value of the dot product between the belief point and each of the alpha vectors given as

$$V(\pi) = \max_{\alpha \in \Gamma} \sum_{x \in \mathcal{X}} \pi(x)\alpha(x) \tag{2.11}$$

For infinite horizon POMDPs although the optimal value function maintains convexity or concavity [9] it loses the piecewise linear characteristics. The optimal infinite horizon value function V can be approximated by successive finite horizon value functions. $V_0, V_1, \ldots, V_N$ as $N \to \infty$ [10]. Even though value function over an infinite horizon can be approximated by a finite number of α vectors for POMDPs with belief states, value iteration gets expensive because with each step new α vectors may be added at an exponential rate. Many algorithms [8, 11, 12] directly try to manipulate the α vectors using combination of set projection and pruning. Others try to perform approximate value iteration by

selecting a small set of representative belief points and maintaining α vectors for each of these points [13]. All these methods fundamentally assume that the environment model (i.e., the observation probability, reward function) is known to the decision maker.

2.3 Reinforcement learning for MDPs

2.3.1 Model based learning

In model based reinforcement learning, the agent learns a model of the environment that predicts how the environment will evolve based on the actions taken. For instance given a state and an action, a model can predict the resultant next state and reward. Once the agent has adequately learned a model of the environment, it can then use planning algorithms with a learned model to find a policy.

2.3.2 Model free learning

In a model free learning, the agent learns a value function or a policy by interacting with the environment. Model free algorithms sample from experience and can be categorized into value based and a policy based learning. In a value based learning, the agent learns an action-value function which provides a value of taking an action given a state. Once the action value function is learned, the agent can then act greedily i.e., select actions with maximum values. Popular value based algorithms include Q learning, SARSA etc.

In policy based learning, the agent directly learns a parametrized policy. Policy gradient methods proposed in [14] are one of the most popular policy search algorithms, where the policy parameters θ are learned based on the gradient of some performance measure $J(\theta)$ with respect to the policy parameter. The gradient of the parametrized policy can be estimated using the "log-derivative trick" and stochastic gradient descent is used to find a locally optimal policy. Popular policy gradient algorithms fall under the broad umbrella of Actor only and Actor Critic methods. In Actor only methods, the policy performance is estimated using Monte Carlo which computes the discounted return from a single sample path. In Actor Critic methods, an action value function is guessed which is iteratively updated using temporal difference learning. A popular actor only algorithm is REINFORCE proposed in [15], the performance gradient is estimated using the following

equation

$$\nabla J(\theta) = \mathbb{E}_{\mathbf{g}}\Big[\nabla_\theta \log \mathbf{g}(\theta) J(\theta)\Big] \tag{2.12}$$

The performance $J(\theta)$ at each time t is estimated using Monte Carlo returns represented by G_t, the policy parameters are then updated as

$$\nabla J(\theta) = \mathbb{E}_{\mathbf{g}}\Big[\nabla_\theta \log \mathbf{g}(\theta) G_t\Big] \tag{2.13}$$

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \log \mathbf{g}(\theta) \tag{2.14}$$

Actor Critic methods provide a biased estimate of the performance gradient where a value function is computed using temporal difference learning (also known as bootstrapping). Compared to Actor only method like REINFORCE, bootstrapping in Actor Critic methods are beneficial because the bias and the reduction in variance introduced by bootstrapping accelerates learning. The performance gradient estimated by the Actor Critic method is analogous to that of REINFORCE except that the return G_t at time t is estimated by the following

$$\nabla J(\theta) = \mathbb{E}_g\Big[\Big(r_{t+1}(x,u) + \gamma Q(x',u') - Q(x,u)\Big)\nabla_\theta \log \mathbf{g}(\theta)\Big] \tag{2.15}$$

$$\theta_{t+1} = \theta_t + \alpha\Big(r_{t+1}(x,u) + \gamma Q(x',u') - Q(x,u)\Big)\nabla_\theta \log \mathbf{g}(\theta). \tag{2.16}$$

Here α is the learning rate and $Q(x,u)$ represent the action value function. When the policy $\mathbf{g}(\theta)$ is differentiable with respect to the parameters θ, the term $\nabla_\theta \log \mathbf{g}(\theta)$ for both the Actor only and Actor Critic methods are computed analytically.

2.4 Reinforcement Learning for POMDPs

Reinforcement learning under partial observability is difficult because complete observability is necessary for learning methods based on MDPs. In POMDPs, the agent only has observations of the state of the environment. This problem is also referred to as the problem of "incomplete information", "perceptual aliasing" or "hidden state". Hence under partial observability, reinforcement learning can be done on an information state that perfectly characterizes the state of the POMDP.

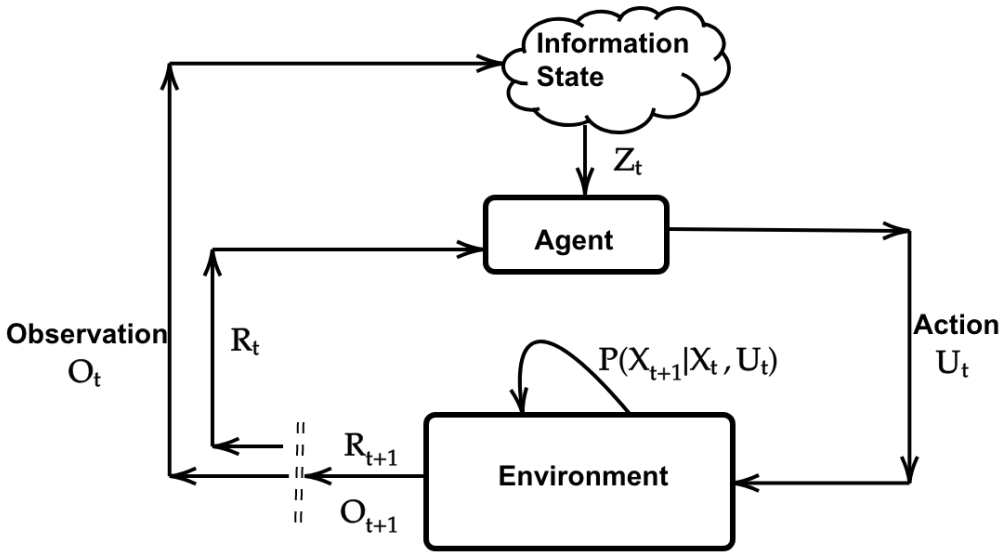


Fig. 2.2 Reinforcement learning in POMDPs

Unfortunately the only general information state for the POMDP is the belief state, the evolution of which requires access to the environment model as shown in equation (2.10). For this reason, many papers on RL for POMDPs [16, 17, 18] take the following view

- Given an initial start state, any policy in POMDPs can be approximated arbitrarily well by a finite state controller.
- Given a finite state controller with a specified size which acts as a memory of previous actions and observations to disambiguate the current state, we can use policy gradient methods to find the best policy with that class.

  Actor Critic methods are difficult for POMDPs since computing a critic using temporal difference learning involves access to the current state which is not available to the decision maker. Therefore for policies represented using a finite state controller, we only focus on the Actor only methods.

In the next chapter we discuss the policy gradient methods for POMDPs with finite state controllers.

Chapter 3

Policy gradient algorithms for POMDPs

In this chapter we outline policy gradient algorithms for POMDPs where the policies are represented using a finite state controller. We discuss the parametrization of the finite state controller and discuss variations of Actor only methods for POMDPs. Actor only methods are lucrative for POMDPs since they do not require computing a value function and lack intractability issues arising from imperfect state information. The algorithms studied in this section appeared in the literature before, however a detailed comparison of these algorithms with the planning solution for the benchmark problem domains did not appear in the literature. Additionally we also consider an alternate form of the policy gradient algorithm known as Renewal Monte Carlo which was demonstrated for MDPs and extend this in the POMDP framework.

3.1 Parametrization of finite state controllers

We consider a stochastic finite state controller with state space $\mathcal{S}$ which is a discrete and finite set. The transition probability function ψ of the internal states of the controller is parametrized with ϕ. This function outputs a distribution over the next states given the current observation, action and the internal state. Thus the next internal state at time $t+1$ is given by $S_{t+1} = \psi_t(S_t, O_t, U_t; \phi)$. Each internal state of the controller outputs a distribution over actions U_t which is parametrized with parameters θ. Thus the action from a particular node is given by the function $U_t = \mathbf{g}_t(S_t, O_t; \theta)$. These two functions can be

parametrized using a family of functions where the parameters belong to a compact and convex set. An example of such parametrization is the softmax function (Gibbs/Boltzman function) which has been adapted in this thesis. The softmax function is given in equation (3.1) .

$$\text{softmax}(z)_j = \frac{e_j^z}{\sum_{k=1}^{K} e_k^z} \tag{3.1}$$

Since the action selection depends on both the current state S_t and the observation O_t, our parameter formulation relates to the concept of a Mealy machine [19] where the output of the controller depends on both the current state and the input to the controller.

Another form of parametrization can be of the form of a Moore machine [20] where the output of the controller would only depend on the current state and not on the input. However this would result in a controller with more states and would often react slowly to faster changing inputs, hence in this thesis we only consider the Mealy machine framework of parametrization.

3.2 Actor only methods for POMDPs

Actor only methods are attractive for POMDPs since obtaining Monte Carlo samples as an estimate of the performance does not require the agent to have access to the states. However, one of the limitations of Monte Carlo methods is that they have a very high variance and are quite slow since they take the complete return from time t and consider all the future rewards until the end of the episode. As a result it is only suitable for episodic tasks. For infinite horizon problems Monte Carlo samples are obtained for large time steps so that the effect of long term rewards become negligible due to discounting which essentially truncates an infinite trajectory to a fixed length trajectory. Infinite horizon policy gradient estimation for POMDPs with policies represented as a finite state controller was proposed in [18, 17], which only provided theoretical guarantees for average cost POMDPs. We consider the infinite horizon discounted cost (or reward) case. The performance gradient estimation with finite state controllers is shown in equation (3.2).

$$\nabla J_{\theta,\phi} = \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \times \sum_{t=1}^{\infty} \big(\nabla_\theta \log \mathbf{g}_t(\theta) + \nabla_\phi \log \psi_t(\phi)\big)\Big]. \tag{3.2}$$

Equation (3.2) follows from the standard policy gradient theorem in [14]. Since the policy is a finite state controller, the gradient of both the action selection function and the internal transition function of the finite state controller is included in the computation of the performance gradient $\nabla J_{\theta,\phi}$.

3.2.1 Renewal Monte Carlo

We study an algorithm proposed in [21], which uses renewal theory [22] in order to estimate the performance gradient for MDPs. This is extended in the context of POMDPs, where the renewal state is visible to the decision maker. Renewal Monte Carlo has both the advantage of low variance as well as low bias. The performance gradient is estimated by the ratio of the expected discounted reward to the expected discounted time over a regenerative cycle. We study both the liklihood ratio based estimator as well as finite difference methods for computing the performance gradients with a finite state controller. Renewal Monte Carlo is fast and requires fewer samples since the algorithm does not have to wait to reach the end of the episode in order to compute the performance gradients. The gradients can be computed whenever a regenerative state is observed by the decision maker. The performance gradient estimation using renewal Monte Carlo is given by:

$$\hat{\nabla}_\theta R = \frac{1}{N}\sum_{n=1}^{N}\sum_{\sigma=\tau^{n-1}}^{\tau^n-1} R_\sigma^n \nabla_\theta \log[\mathbf{g}(\theta)]_\sigma \tag{3.3}$$

$$\hat{\nabla}_\theta T = \frac{1}{N}\sum_{n=1}^{N}\sum_{\sigma=\tau^{n-1}}^{\tau^n-1} T_\sigma^n \nabla_\theta \log[\mathbf{g}(\theta)]_\sigma \tag{3.4}$$

$$\hat{\nabla}_\phi R = \frac{1}{N}\sum_{n=1}^{N}\sum_{\sigma=\tau^{n-1}}^{\tau^n-1} R_\sigma^n \nabla_\phi \log[\psi(\phi)]_\sigma \tag{3.5}$$

$$\hat{\nabla}_\phi T = \frac{1}{N}\sum_{n=1}^{N}\sum_{\sigma=\tau^{n-1}}^{\tau^n-1} T_\sigma^n \nabla_\phi \log[\psi(\phi)]_\sigma \tag{3.6}$$

where N is the number of regenerative cycles T corresponds to the discounted time and $\sigma \in \{\tau^{n-1}, \ldots, \tau^{n-1}\}$ and τ corresponds the the stopping time when the system returns to the start state.

$$\nabla J_{\theta,\phi} = [\hat{T}_\theta \hat{\nabla} R_\theta - \hat{R}_\theta \hat{\nabla} T_\theta] + [\hat{T}_\phi \hat{\nabla} R_\phi - \hat{R}_\phi \hat{\nabla} T_\phi] \tag{3.7}$$

3.3 Deep reinforcement learning for POMDPs

In deep reinforcement learning, the policy and/or the action value function are approximated with a neural network. Deep reinforcement has been used for pixel based environments [1] which are environments with partial observability. The problems with partial observability is tackled by keeping track of event histories using RNN with LSTM layer post convolution layer for these pixel based environments. We analyse policy gradient methods for POMDPs with finite state controllers as well as deep learning models in order to compare their performance. We consider Recurrent Policy gradient algorithm where the policy is represented by a recurrent neural network (RNN) with long short term memory (LSTM) module [23]. RNN with LSTM layer provides limited memory for training policies that require storing sequences of observation over time. The method involves approximating policy gradient where the estimated return weighted eligibilities are backpropagated through time using RNN. This allows the policies to be mapping from event histories to actions. We used recurrent policy gradient algorithm with history dependent baselines for learning optimal policies in POMDPs. Recurrent policy gradient algorithm is an actor only algorithm where the returns are estimated using Monte Carlo samples. To tackle the issue with high variance [24] proposes a method for variance reduction by subtracting a constant baseline b from the gradient estimate. Typically baseline constant b is calculated by taking an expectation of the average return which is then subtracted from the original return. In RPG this baseline is calculated by incorporating a history dependent function approximator.

Chapter 4

Experimental analysis

In this chapter we make an analysis of the performance of different algorithms. We evaluate the algorithms on four classical POMDP environments with various state and observation spaces. In the sequel we briefly discuss each of the environments under consideration.

4.1 Environment Description

4.1.1 Tiger environment

This example first proposed in [25] consists of an agent standing behind two closed doors. Behind one of the door is a tiger and behind other is a large reward. The agent can open either of the doors or it can listen, in hopes of gaining some information about the location of the tiger. However, listening is neither free nor completely accurate. The agent's possible actions are *LEFT*,*RIGHT* and *LISTEN*. The reward for opening the correct door is +10 and the penalty for choosing the door with the tiger behind it is -100. The cost of listening is -1. There are only two possible observations: to hear the tiger on the left (TL) and or to here the tiger on the right (TR). The *LISTEN* action does not change the state of the world. The *LEFT* and *RIGHT* actions cause the problem to reset, with the tiger replaced randomly behind one of the doors. When the tiger is on the left, the listen action results in observation TL with a probability of 0.85 and the observation TR with a probability of 0.15; conversely when the tiger is on the right the listen action results in observation TR with a probability of 0.85 and TL with a probability of 0.15. Regardless of the state of the world *LEFT* and *RIGHT* actions result in either of the observation TR or TL with

probability 0.5. For this infinite horizon problem a discount factor of 0.95 is used. For the tiger problem the state resets whenever the the agent opens a door, hence renewal occurs if the state of the tiger is placed randomly to the initial state.

4.1.2 Voicemail environment

To illustrate the use of POMDP framework in the spoken dialogue system, an example using a simple voicemail application has been proposed in [26]. In this example, users listen to voicemail messages where at the end of each message the users are posed with a choice of either saving the message or deleting them. We refer to this as the user's goal and since the system does not know a priori which goal the user desires, they are considered as a hidden goal. For the duration of interaction relative to each message, the user's goal is fixed and the POMDP based dialogue manager is trying to guess which goal a user has. The machine therefore has three available actions. i) It can ask the user what they want to do in order to infer his or her current goal, or it can *doSave* or *doDelete* and move on with the next message. When the user responds to a question, it is decoded as either the observation save or delete. However, since the speech recognition error can corrupt the user's response, these responses cannot be used to detect the user's intent with certainty. If the user says *save* then an error may occur with probability 0.2 where as if the user says *delete*, then an error may occur with probability 0.3. Finally since the user wants *save* more than *delete* the initial belief is set to (0.65,0.35) and it is reset to this value after each *doSave* or *doDelete* via transition function.

The machine receives a large positive reward (+5) for getting user's goal correct a very large negative reward (-20) for taking the action *doDelete* when the user wanted *save* and a smaller but still significant negative reward (-10) for taking the action *doSave* when the user wanted *delete* since the user can always delete the message later. There is also a small negative reward (-1) for taking the action *ask*. The transition dynamics of the system is shown in the following table.

Table 4.1 Voicemail transition function

		x'	
a	**x**	**save**	**delete**
ask	*save*	1	0
	delete	0	1
doSave	*save*	0.65	0.35
	delete	0.65	0.35
doDelete	*save*	0.65	0.35
	delete	0.65	0.35

Table 4.2 Voicemail observation probability

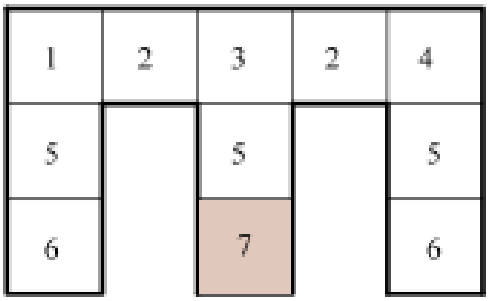
		o'	
a	**s'**	**save**	**delete**
ask	*save*	0.8	0.2
	delete	0.3	0.7
doSave	*save*	0.5	0.5
	delete	0.5	0.5
doDelete	*save*	0.5	0.5
	delete	0.5	0.5

Table 4.3 Voicemail reward structure

	s	
a	**save**	**delete**
ask	*-1*	-1
doSave	*+5*	-10
doDelete	*-20*	+5

For the voicemail environment the problem resets when ever the agent takes the *doSave* or *doDelete* action. When the environment returns do initial state from which it started, a renewal is said to have taken place. For this infinite horizon problem, a discount factor of 0.95 has been used.

4.1.3 McCallum's Cheese Maze



This is an 11 state POMDP presented in [27] where the objective of the agent is to go to the goal state. However, the agent is partially informed. The agent as total 7 observations where the observations corresponds to what would be seen in all four directions immediately to the adjacent location. Here states 5, 6 and 7 all have the same observations i,e 5. The agent is allowed 4 control actions which corresponds to moving *NORTH*, *EAST*, *WEST* and *SOUTH*. A reward of +1 is only received when the agent reaches the goal state 10. When the agent reaches the goal state the problem resets and the agent is randomly placed into any one of the states except the goal state. For this environment a discount factor $\gamma = 0.70$ was used. For this environment a renewal happens when the agent randomly reaches the random state to which the agent resets is identical to the start state.

4.1.4 4X4 Grid

This POMDP environment is a simple $4x4$ grid with 16 states which was proposed in [3]. The goal state is in state 16. The agent can either move *NORTH*, *SOUTH*, *EAST*, *WEST*. At the goal state the agent receives a reward of 1. The agent can only observe the goal state. When the agent reaches the goal state, the agent is randomly placed with uniform probability in any one of the 15 states except the goal state. For this environment a renewal happens when the agent randomly resets to the starting state after reaching the goal state.

4.2 Planning solutions

The environments considered in this thesis are the ones for which it is easier to compute the optimal planning solution. For each of the environments outlined above, the planning

solution is computed using incremental pruning algorithm [28], table 4.4, provides the optimal number of controllers for each of these environments as well as the optimal value function for a specified belief state.

Table 4.4 Optimal planning solution

Environments	**Discount**	**Optimal controller size**	**Initial beleif**	**Optimal value function for the initial belief**
Tiger	0.95	9	[0.5,0.5]	19.371
Voicemail	0.95	34	[0.65,0.45]	3.462
CheeseMaze	0.70	16	[0.1,...,0.1, 0.0]	0.3474
4x4 Maze	0.95	20	[0.667,...,0.667,0.0]	3.799

4.3 Experimental setup

For the Actor only methods with Monte Carlo policy evaluation, Monte Carlo policy evaluation are obtained over a horizon of 300 steps. The performance of the policy is obtained by averaging 10 independent Monte Carlo runs after every 20 policy improvement steps. Each experiments are run with 5000 iterations (policy improvement steps).

For Actor only method with Renewal Monte Carlo policy evaluation each policy improvement steps takes an average over 30 renewals. The performance is evaluated by averaging over 10 independent Monte Carlo runs each with a horizon of 300 steps. For Renewal Monte Carlo the policy performance is evaluated after every 20 policy improvement steps. Each experiment with RMC is performed with 50000 iterations (policy improvement steps).

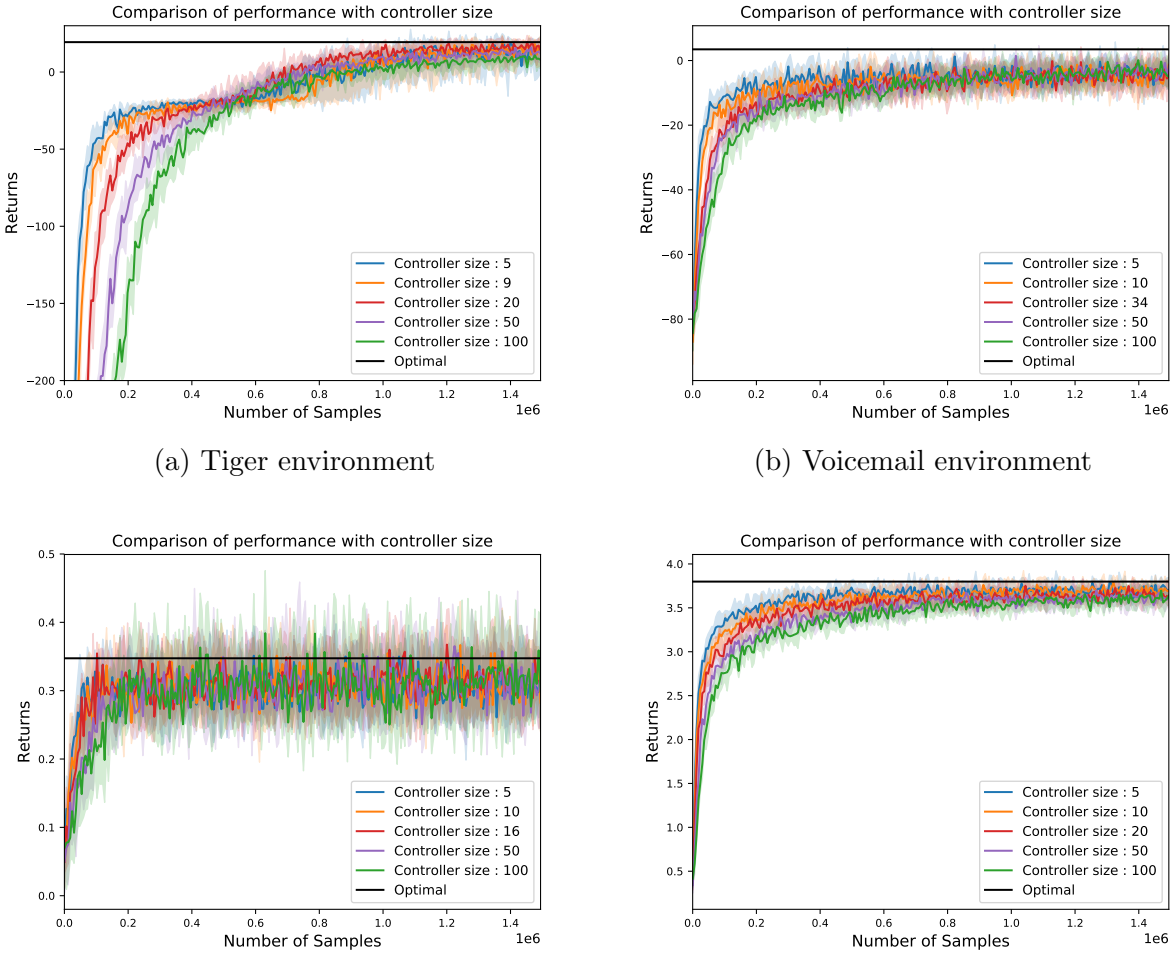
For recurrent policy gradients, Monte Carlo policy evaluation is obtained over a horizon of 300 steps. The performance of the policy is obtained by averaging 10 independent Monte Carlo runs after every 20 policy improvement steps. Each experiments are run with 5000 iterations (policy improvement steps).

We perform 25 independent runs for each of these experiments and report the mean, variance, median as well as the interquartile range of the performance for each problem domain.

4.4 Results

4.4.1 Actor only methods

Fig. 4.1 Comparison of performance with varying controller size



(a) Tiger environment

(b) Voicemail environment

(c) Cheese maze environment

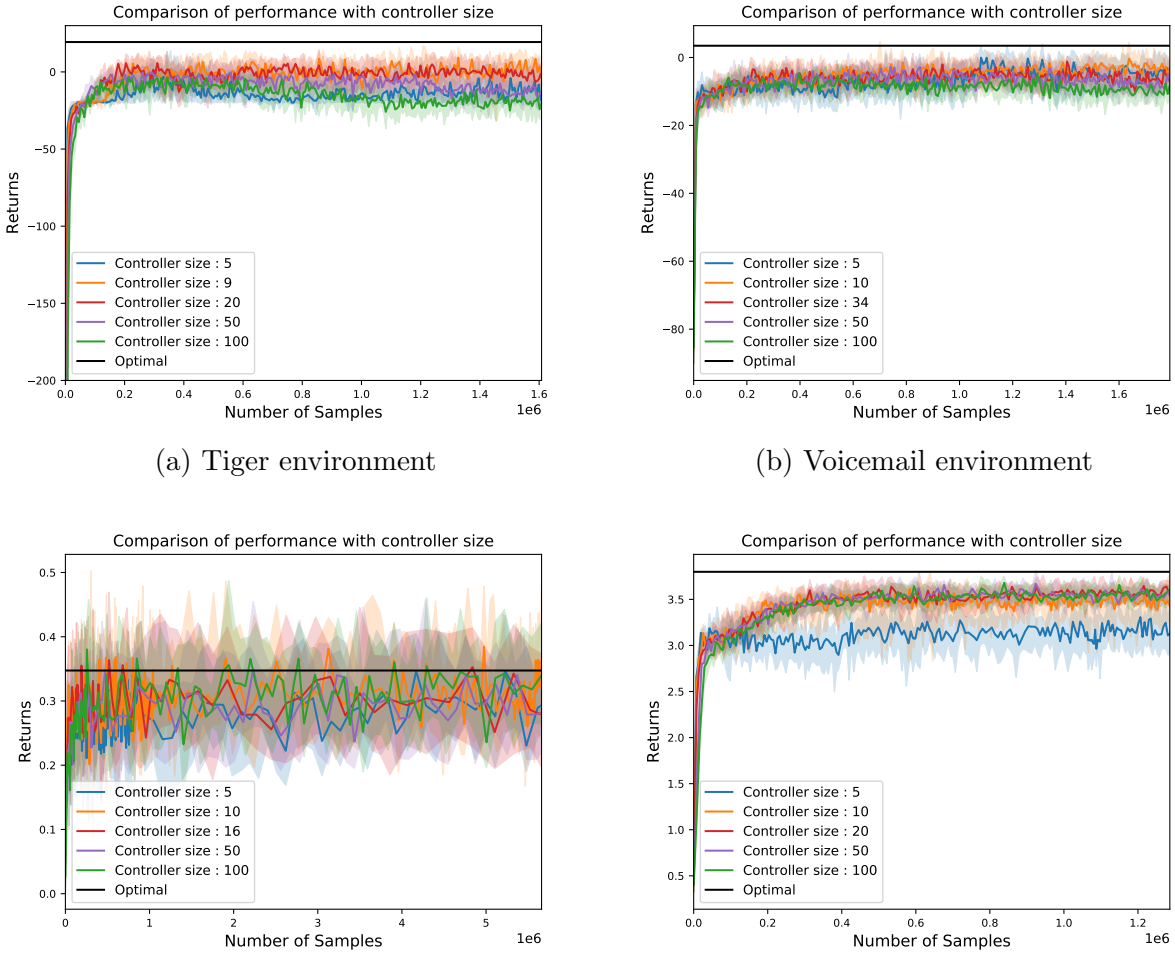
(d) 4x4 grid environment

Figure 4.1 shows that for all the environments the policies with finite state controllers where score function method is used to optimize the policy have converged to a local minima. The optimal planning solution is shown in black lines. The solid lines show the median and the shaded part shows the inter quartile range (i,e the region between the 75^{th} percentile and 25^{th} percentile. The median and the interquartile range was computed across

25 independent runs. We progressively varied the size of the controller where we observed that increasing the size of the controller leads to a slower convergence which is expected since increasing the number of controllers lead to a an increase in the number of parameters. For all the problems, policy gradient with finite state controllers converges to a policy that is close to optimal. However, given the nature of the tiger problem, the listen only policy (where the reward is -1) is a natural attractor, as a result in order to get out of this natural attractor, and it takes more number of samples to converge close to an optimal policy. For all these problems, increasing the size of the controller progressively does not lead to a significant improvement in the performance. Interestingly, these problems, a stochastic finite state controller, with a controller size less than that of the optimal (deterministic) controller yield similar performance. This is true since stochastic policies for POMDPs results in higher returns.

4.4.2 Renewal Monte Carlo

Fig. 4.2 Comparison of Renewal Monte Carlo with exact solution for different controllers



(a) Tiger environment

(b) Voicemail environment

(c) Cheese maze environment

(d) 4x4 grid environment

Figure 4.2 shows the performance of Renewal Monte Carlo algorithms with different controller sizes. For all the examples considered, the renewal state is visible to the decision maker. We observe that RMC with score function requires less number of samples compared to Monte Carlo. The lines in black shows the exact solution obtained using planning algorithm. Comparison with score function based method is only presented since estimation with finite differences does not perform well in these environments. We also emphasize

on the fact that the policy (which is a finite state controller) is differentiable with respect to the parameters, hence using the score function method is an ideal case for estimating the gradients. The lines show the median across 25 independent runs and the shaded region shows the interquartile range. In addition to it, a comparison with different number of controllers are performed. We observed that for the Tiger and the 4x4 grid environment, increasing the controller size increases the performance however, for the other two scenarios, the performance of the learning algorithm is invariant to the number of controllers. Since RMC has a low variance, it is unable to get rid of a local minima which are strong attractors. This is evident from the tiger problem where the solution converges to the *LISTEN* only action which is attainable with only one controller.

4.4.3 Recurrent policy gradient

The experiment with recurrent policy gradient was performed with a single LSTM layer with hidden size of 20, this is followed by a RNN layer and finally the fully connected softmax layer is used in order to obtain distributions over actions. Since we are using a history dependent baseline, the value network also contains a LSTM cell followed by an RNN layer. Figure 4.3 shows the over all architecture used for recurrent policy gradient. Figure 4.5 and 4.4 shows the value and the policy network architecture respectively.

The performance of recurrent policy gradient algorithm is outlined in figure 4.6 shows the performance of recurrent policy gradients compared to the planning solution. The lines in black show the exact solution. For the Tiger problem, recurrent policy gradient also converges to the *LISTEN* only action. On the other hand, for the Voicemail and the 4x4 grid, recurrent policy gradient obtains performance that is close to the optimal. However, it fails to learn for the Cheese Maze environment. Figure 4.6 shows the mean performance of the RPG algorithm. The shaded region represents 1 standard deviation.

We observe that recurrent policy gradient converge to a local solution and is affected by the strong attractor region for the Tiger problem. In case of Cheese Maze problem, we observed that the mean of the performance converges close to the optimal where the median of the performance does not. This illustrates the fact that for the Cheese Maze environment, some runs lead to convergence to a local optimal and some does not learn at all.

For all the performance plots reported, the median and the mean values are obtained

across 25 independent runs.

Fig. 4.3 RPG architecture

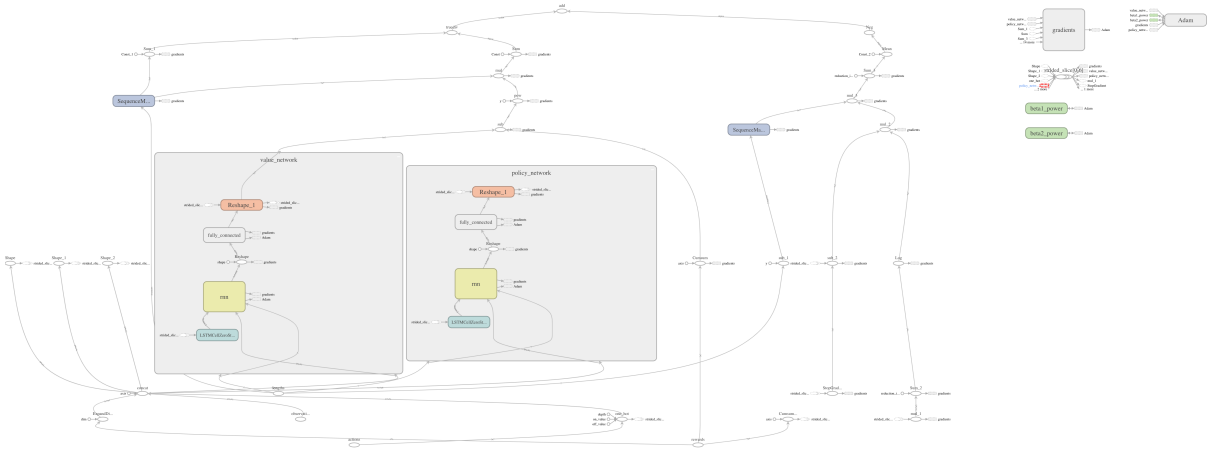


Fig. 4.4 Policy network

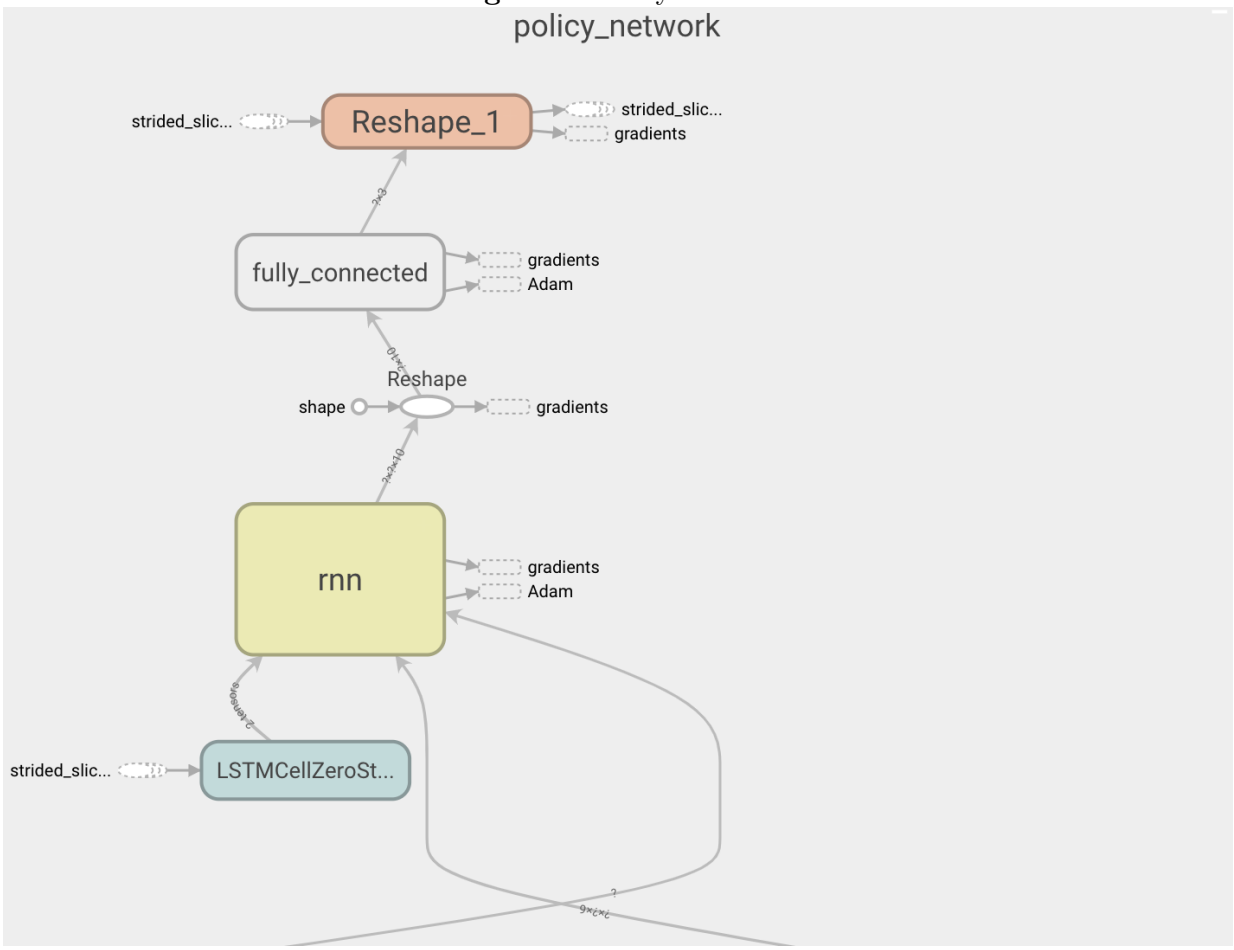


Fig. 4.5 Value network

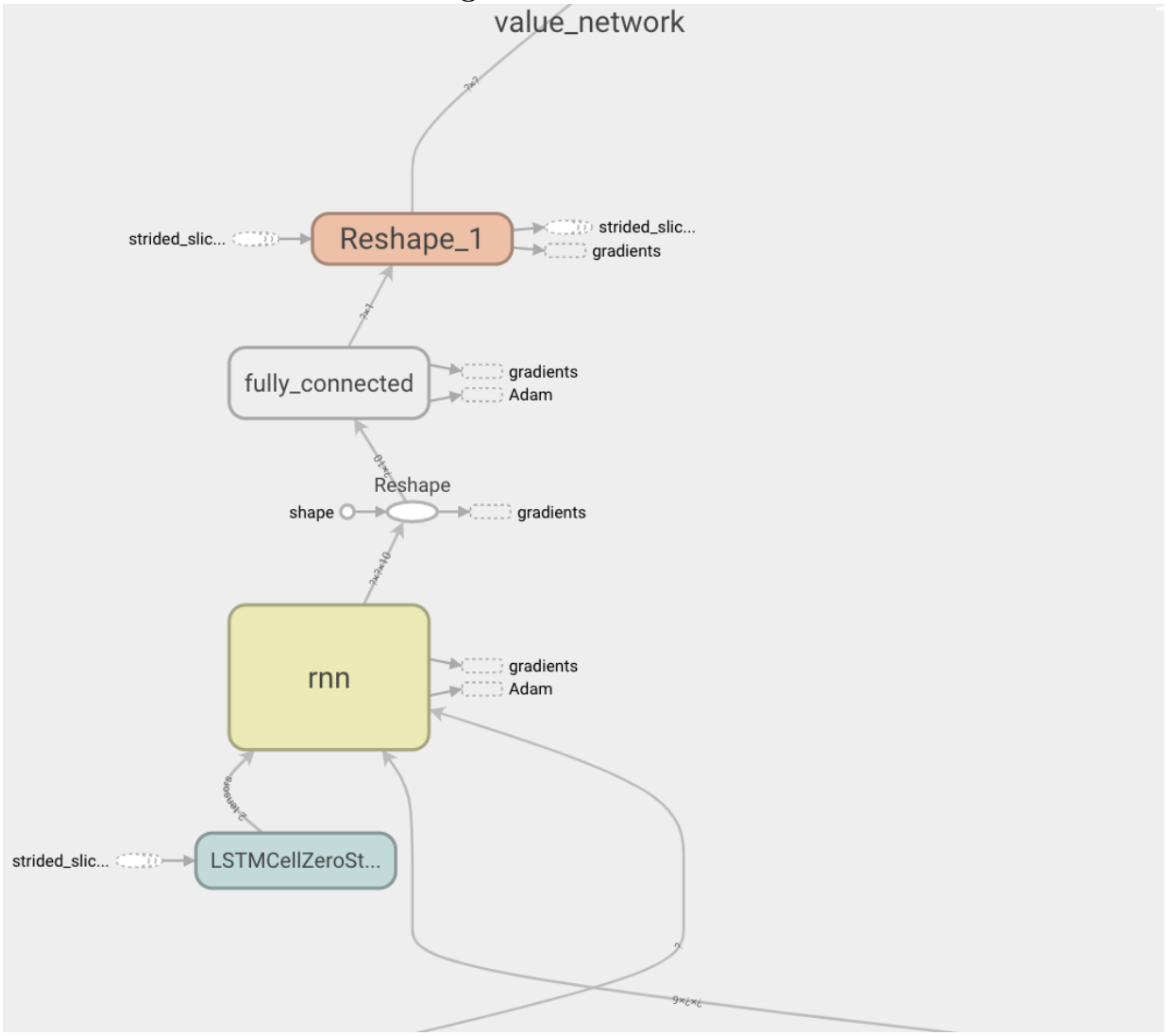
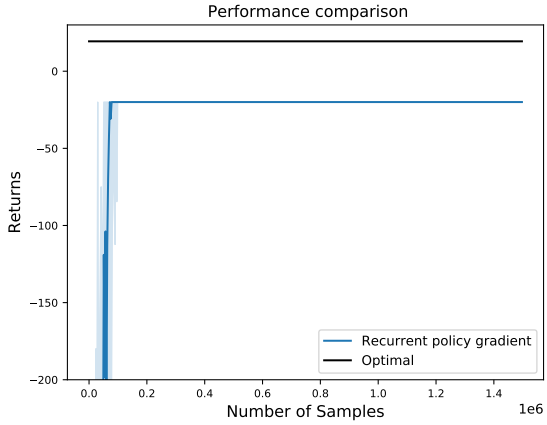
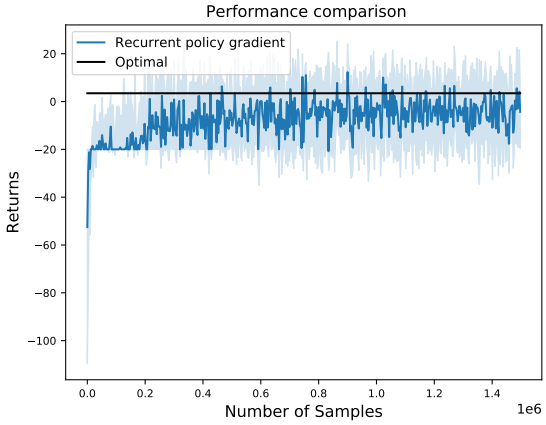


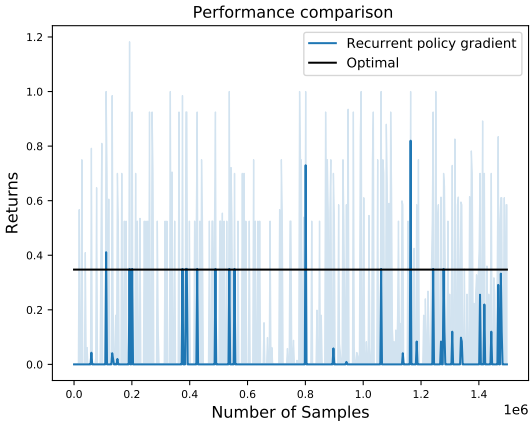
Fig. 4.6 Performance comparison (median) for recurrent policy gradients with exact solution.



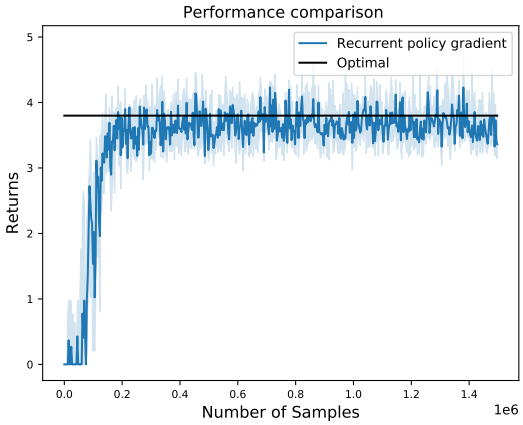
(a) Tiger environment



(b) Voicemail environment

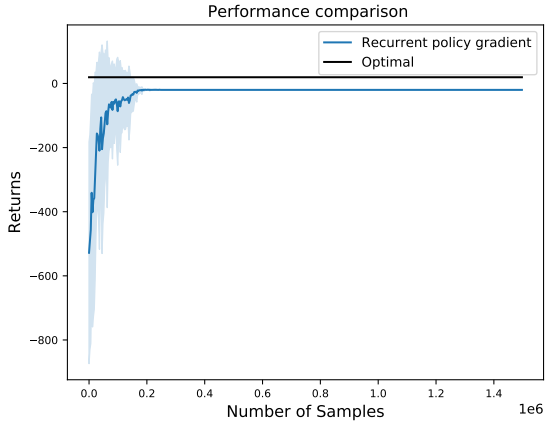


(c) Cheese maze environment

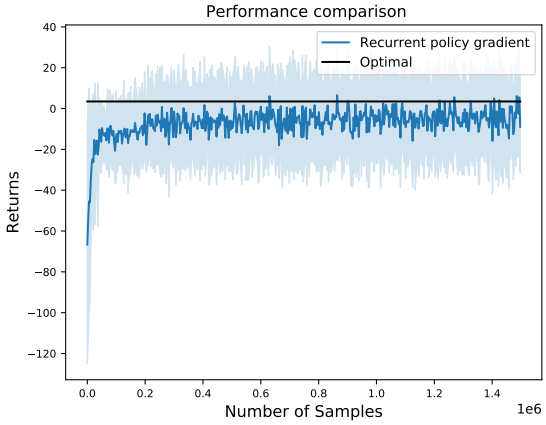


(d) 4x4 grid environment

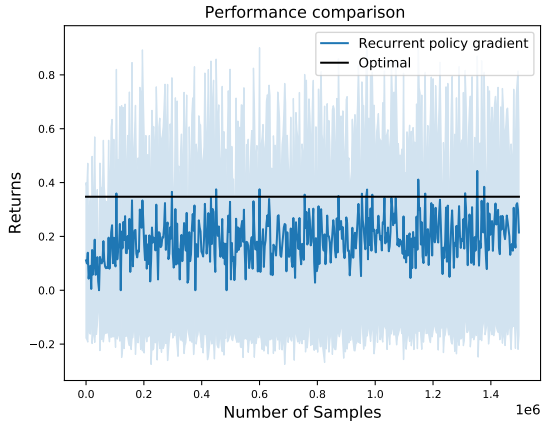
Fig. 4.7 Performance comparison (mean) for recurrent policy gradients with exact solution.



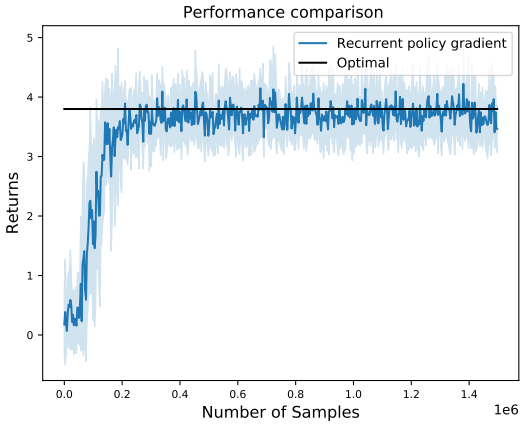
(a) Tiger environment



(b) Voicemail environment



(c) Cheese maze environment



(d) 4x4 grid environment

The following table summarizes the mean performances of different learning algorithms for POMDPs considered in this thesis.

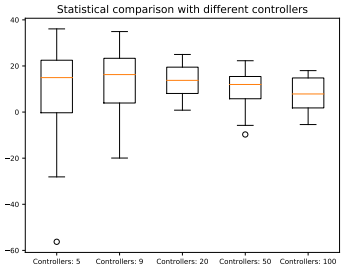
Table 4.5 Mean and Median performances of different learning algorithms

Environments	Tiger					Voicemail					Cheese Maze					4x4 Grid				
Controllers	5	9	20	50	100	5	10	34	50	100	5	10	16	50	100	5	10	20	50	100
Mean Performance Monte Carlo	5.44	5.51	11.83	9.14	4.49	-4.50	-5.14	-4.94	-4.38	-4.25	0.31	0.32	0.32	0.32	0.32	3.70	3.70	3.64	3.61	3.59
Mean performance Renewal Monte Carlo	-18.65	-14.22	-10.33	-10.12	-14.12	-10.67	-9.02	-8.43	-8.68	-9.32	0.28	0.29	0.31	0.30	0.27	3.11	3.32	3.47	3.54	3.52
Mean performance RPG	-19.99					-4.48					0.21					3.72				
Median Performance Monte Carlo	10.48	11.27	14.12	10.15	5.18	-4.30	-5.14	-4.75	-4.17	-4.11	0.31	0.31	0.32	0.31	0.31	3.69	3.69	3.64	3.60	3.58
Median performance Renewal Monte Carlo	-13.38	0.30	-0.97	-10.1	-19.5	-4.10	-3.61	-5.80	-6.94	-9.27	0.29	0.31	0.32	0.31	0.32	3.15	3.48	3.56	3.54	3.54
Median Performance RPG	-19.99					-4.37					0.003					3.70				

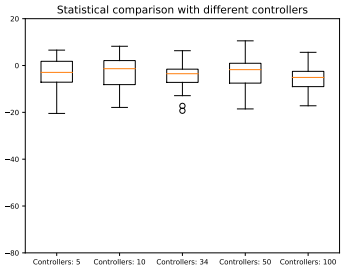
4.5 Box plot analysis

Fig. 4.8 Box plot analysis.

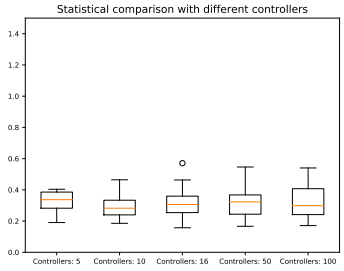
MC



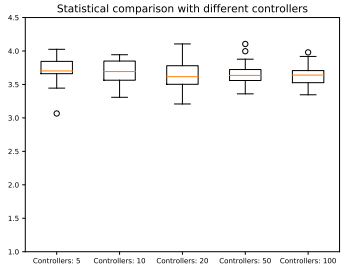
(a) Tiger



(b) Voicemail

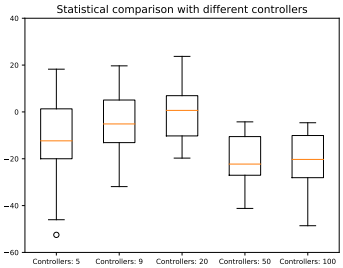


(c) Cheese Maze

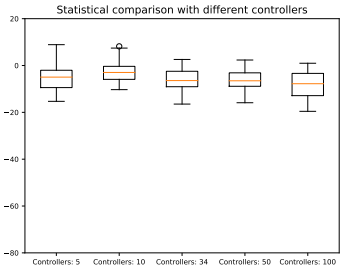


(d) 4x4 Grid

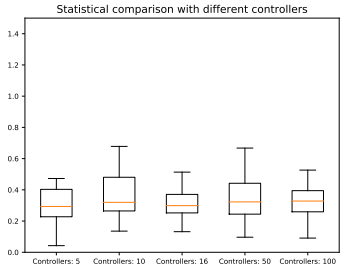
RMC



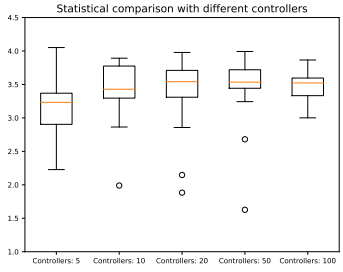
(e) Tiger



(f) Voicemail

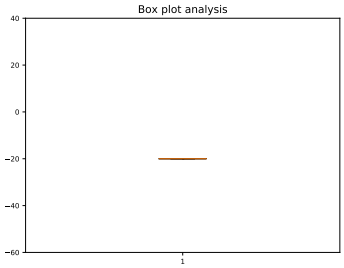


(g) Cheese Maze

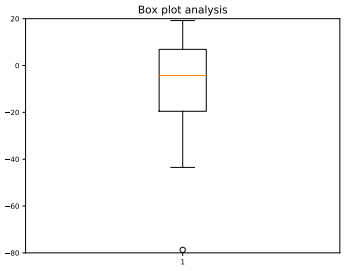


(h) 4x4 Grid

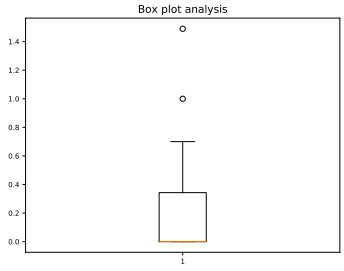
RPG



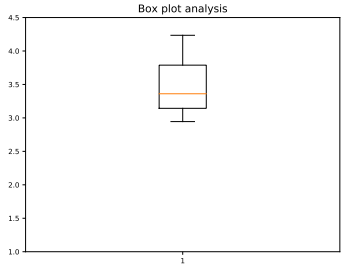
(i) Tiger



(j) Voicemail



(k) Cheese Maze



(l) 4x4 Grid

Figure 4.8 shows the statistical analysis of the performance for Policy gradient with Monte Carlo Policy evaluation (MC), Renewal Monte Carlo policy evaluation (RMC) and Recurrent policy gradients (RPG). The plots show the performance measure across 25 independent runs for different controller size. The line in orange shows the median performance value and the width of the box shows the interquartile range i.e., the difference between 75^{th} percentile and 25^{th} percentile of the performance measure. The box plots were plotted by considering the performance value of the last sample for each of the 25 independent runs.

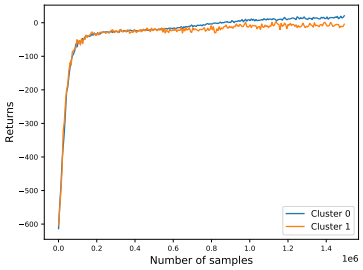
4.6 Clustering Analysis

In order to compare Monte Carlo and Renewal Monte Carlo and Recurrent policy gradient methods, we carry out a clustering analysis to determine different sample paths that the solution for these two methods converge to. To understand whether independent runs converge to different local minima, we perform x-means clustering on the returns for different environments. X-means clustering proposed in [29] extends the k-means clustering algorithm where the number of clusters are automatically determined using the Bayesian information criterion (BIC). The algorithm is initiated with minimum 1 cluster and then new clusters are formed based on BIC. Figure 4.9 shows the clustering results for gradient estimation using score function with Monte Carlo (MC) and Renewal Monte Carlo (RMC) policy evaluation and Recurrent policy gradient (RPG) algorithm. For Monte Carlo with score function we observe that for the Tiger environment, the converged sample path can be used to form two different clusters confirming that there exists a region of strong attractors where singnificant number of samples converge to. In case of RMC, we see overlapping clusters because none of the sample paths were able to get rid of the strong attractor points. For the 4x4 grid environment with Renewal Monte Carlo, also two separate clusters are formed, showing that a noticeable number of runs converge to another local minima. Interestingly for all the other examples, we observe overlapping clusters where the sample paths cannot be separable by higher number of clusters according to BIC. Hence the number of clusters remains to be 2 (i.e.,the initial number of clusters). For RPG algorithm, only one cluster is formed from since all the experiments converge to the LISTEN only policy. In case of Cheese Maze and Voicemail envrionment, 6 and 7 clusters are formed respectively showing the existance of 6, 7 local minima. For the 4x4 grid, only two clusters were formed.

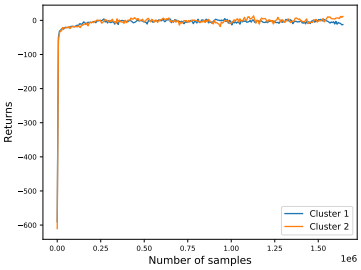
For all the environments we see that there are overlapping clusters, i.e., all the sample paths for 25 independent runs converge to the same local minima corresponding to each environment for both MC and RMC methods.

Fig. 4.9 Clustering Analysis.

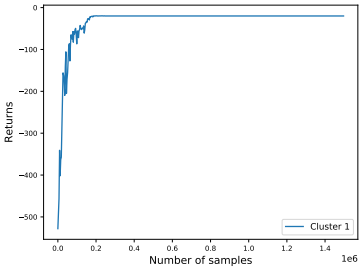
MC RMC RPG



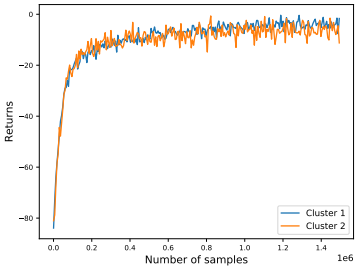
(a) Tiger



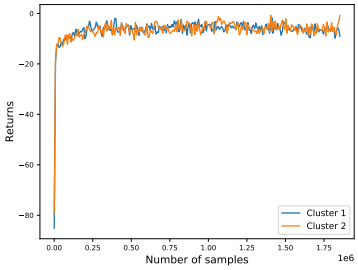
(e) Tiger



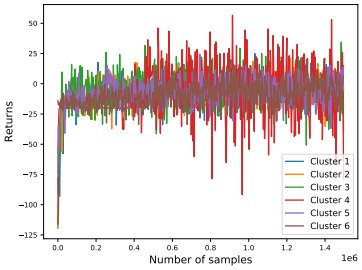
(i) Tiger



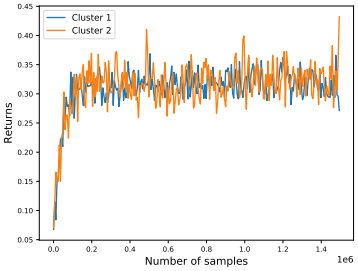
(b) Voicemail



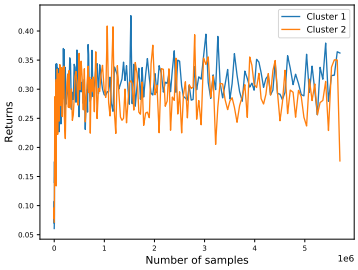
(f) Voicemail



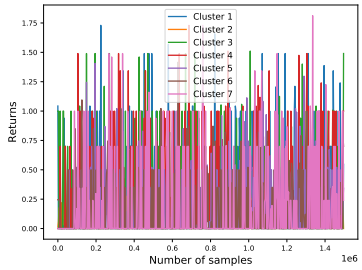
(j) Voicemail



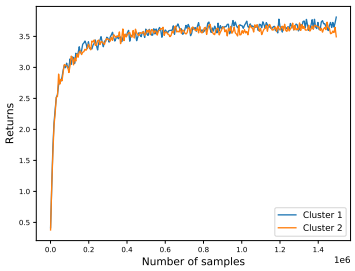
(c) Cheese Maze



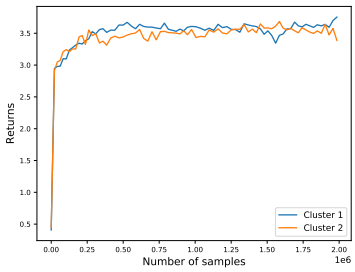
(g) Cheese Maze



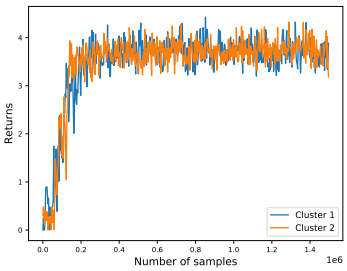
(k) Cheese Maze



(d) 4x4 Grid



(h) 4x4 Grid



(l) 4x4 Grid

4.7 Discussion

We perform an analysis of different policy gradient algorithms for POMDPs. From this empirical analysis it is shown that similar performance can be achieved with a stochastic finite sate controller with fewer states than that of a deterministic finite state controller. Our box plot analysis of the performance of different algorithms shows that Renewal Monte Carlo, quickly converges to a local minima, compared to policy evaluation done using Monte Carlo only. However, RMC often converges to a poor local minima for and Table 4.5 shows that the mean performance of RMC algorithms is less in all the environments for different number of controllers. RMC is a low bias and low variance algorithm compared to Monte Carlo method for policy evaluation. However, empirical results show RMC to perform poorly compared to Monte Carlo methods. A possible explanation for this would be that a low variance and bias in policy evaluation does not necessary lead to a low variance and bias in the gradient estimate. Our analysis with recurrent policy gradient algorithm shows that it has a high variance compared to finite state controllers despite of using a history dependent baselines for variance reduction. In particular for the Tiger problem we observed that recurrent policy gradient algorithm gets trapped in a region of strong attractor point which is the LISTEN policy only can can be achieved with single degenerate controller. We performed an extensive hyper parameter search for different network configuration as well as the hidden state and observed that for all these cases, all of the 25 independent runs converge to listen only policy hence the box plot analysis for RPG is represented by a single line. For the Cheese Maze environment, the median performance of RPG is quite poor while analysing the mean, we observe that the mean value tends to be converging. This is further confirmed by the box plot analysis showing that some runs perform really poorly while other performs really well. RNN with an LSTM module had been widely used for finding near optimal policies for POMDPs with large state spaces, however, when used on simple problem domain, they are really difficult to train. Hence for simpler POMDPs, it is suggested that one uses stochastic finite state controllers rather than RNN with an LSTM cell to store a history.

Chapter 5

Background on concept drifts and indoor localization

This chapter marks the beginning of the second part of this thesis, in this chapter we give a brief overview of concept drifts in different contexts. We then discuss the background literature on indoor localization with wireless signals and explain the nature of data that is widely used for wireless localization.

5.1 Concept drifts

The evidence of concept drift is reflected in the training examples. Observations about the underlying data distribution become irrelevant compared to the current phenomenon which the predictive models should forget. Consider a supervised learning set up with observation pair $\{\vec{xi}_i, y_i\}$, and m classes where $y_i \in \{C_1, C_2, \ldots, C_m\}$. At each time instant the learning algorithm outputs a prediction $\hat{y_i}$ for a given feature $\vec{x}_i$ with the assumption that the examples are generated independently and at random from an underlying stationary distribution $\mathcal{D}$. Given infinite number of such examples learning algorithms such as Naive Bayes can essentially model the distribution $\mathcal{D}$ with an accuracy bounded by Bayes error [30]. Consider the case when the observations $\{\vec{x}_i, y_i\}$ are obtained from a distribution $\hat{\mathcal{D}}$ which is non stationary. Furthermore, if a sequence of data $\langle S_1, S_2, S_3, \ldots \rangle$ where each element in the set has some stationary distribution $\mathcal{D}_i$ is observed, any learning algorithm fails to guarantee arbitrary precision. Each of the distributions pertaining to the sequence can be learned provided that the number of observation is large enough. However, the difficulty

lies in detecting the change in this distribution and identifying from which distribution the observed data is actually coming from. This change in data distribution is referred to as concept drifts. Although change point detection algorithms are useful for detecting abrupt drifts (also known as Concept Shift), they usually perform poorly to detect gradual drifts (also known as the Concept Drift). Although the techniques proposed in the literature for handling concept drifts are problem specific, they can be broadly classified into two categories:

- Active solutions: In an active solution, a change detection test [31] is performed in order to detect concept drifts, the model is then updated in order to maintain its predictive accuracy
- Passive solutions: In a passive solution, the model is continuously updated by retraining on most recent observation [32] or by taking an ensemble of different predictive models.

In this thesis we primarily focus on concept drifts for WiFi channel state information (CSI) data for the application of passive indoor localization.

5.2 Channel state information

In wireless communication a signal is transmitted from a transmitter (Tx) to the receiver (Rx) through multiple transmission channels using orthogonal frequency division multiplexing (OFDM). This allows the transmitter to simultaneously broadcast on narrowly separated subcarriers at different frequencies within each channel in order to increase the data rate. Channel state information (CSI) which is measured at the receiver characterizes the propagation of a signal and incorporates the effect of various forms of distortion experienced by the transmitted signal including fading, shadowing and multipath effects. The signal propagation behaviour can be quantitatively analysed in a WiFi covered area in order to extract different disturbance information pertaining to human motion; which we use to determine the location of a subject in this thesis.

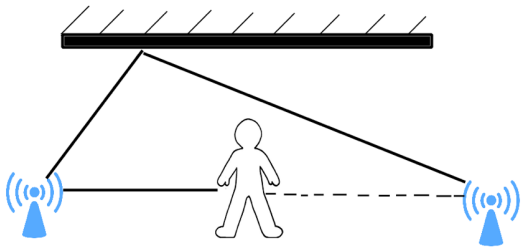


Fig. 5.1 Schematic diagram of multipath effect.

Consider the link between each pair of transmitter and receiver denoted by l where $l \in \{1, 2, \ldots, L\}$. Let i be the subcarrier index $i \in \{1, 2, \ldots, N\}$ where N is the total number of subcarriers. $CSI_{i,l}(t)$ is a complex number for the i subcarrier and l at time t given by

$$CSI_{il} = |CSI_{il}| \exp\{j \sin \angle CSI_{il}\} \tag{5.1}$$

5.3 Indoor localization with WiFi

Indoor localization using WiFi signals have gained much attention over time due to the development of wide range of services that leverages Internet of things (IoT). Existing localization strategies can be broadly classified into two forms.

- Active WiFi localisation: Active WiFi localization has the concept of a target device equipped with WiFi receiver whose location is to be determined relative.
- Passive WiFi localisation: Passive WiFi localization requires the position of the signal transmitting and the receiving devices to be fixed. The location of a moving subject is determined by analysing the distortion pattern of the received signals.

Many indoor localization systems leverage WiFi RSSI data for localization since obtaining RSSI data is easier and has low hardware requirements [33]. The Horus system proposed in [34] proposes a probabilistic model in order to perform passive indoor localization. The RADAR system proposed in [35, 36] uses WiFi RSSI for locating and tracking users inside buildings. One of the limitations of RADAR is that the system uses RSSI data and require multiple devices positioned to provide overlapping coverages. [37] is the the first work that

leverages WiFi CSI data for indoor localization. The proposed system named FILA is a range based localization system that uses CSI data along with a trilateration estimation from multiple nodes for active localization. [38, 33, 39], are some of the existing research where device free passive WiFi localization is used along with deep learning. In [40], the authors address drifts and the inconsistency of WiFi fingerprints for stationary subjects. However, most of this research and its experiments are performed in a very controlled environment and within a limited time frames. More rigorous analysis regarding other localization techniques can be found in [41]

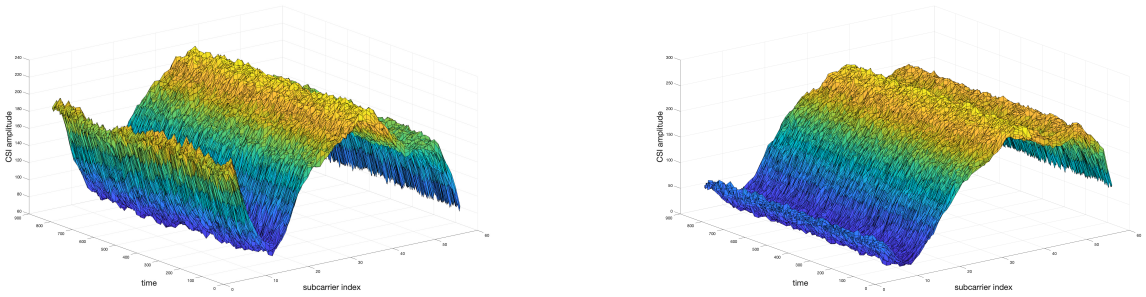
In recent years Wi-Fi based activity recognition and indoor localization using channel state information has gained much attention. This is primarily because CSI contains fine grained information rather than the Receiver signal strength indicator (RSSI) which was the mainstream available wireless signal measurement widely used in the literature. Also RSSI suffers in complex environments due to multipath fading and temporal dynamics [42].

5.4 Concept drifts in WiFi CSI for indoor localization

Concept drift is one of the most common problems that degrades the predictive performance of passive WiFi-based localization systems.The presence of concept drift means that the accuracy of the predictive models that are trained from historical data degrades over time due to evolving nature of the data. Hence, predictive models often need to be retrained frequently with a new set of labelled data, which might be expensive to obtain. These pattern changes can be categorized based on their transition speed from one state to another into abrupt, or gradual drifts [43]. In either case, the deployed solution is expected to diagnose unintended changes automatically and adapt accordingly.

The problem of concept drift in WiFi-based localization systems, was first mentioned in [6], which presents a technology that utilizes only off-the-shelf WiFi-enabled devices such as access points, laptops, smart TV for passive sensing in the environment of interest. The authors applied an online semi-supervised approach to automatically detect gradual shifts in the feature space and propose an adaptive learning strategy to regain the prediction accuracy. This thesis aims to address the same problem without making any assumption about the drift type. We illustrate that from time to time, both sudden and gradual drifts, can occur to the streaming WiFi data, which often hinder the performance of the trained models when tested on the measurements.

The majority of the existing WiFi-based indoor localization systems are device-based, where the user's location is determined by a WiFi-enabled target device that needs to be carried by the subject all the time [44]. Practical challenges of using device-based approaches, impose some restrictions and therefore, a device-free and passive solution is a promising line of research both for academia and industry. The effect of concept drift mostly appears over time due to real-world conditions such as natural WiFi channel or bandwidth switches, or when certain exogenous factor such as temperature and humidity changes. Therefore, the existing methods do not address them explicitly and the experimental results do not reflect the performance of the model in measurements taken few days apart. Figure 5.2 shows the different shapes of the capture which are 2 hours apart showing the presence of drifts. Surprisingly this change in data distribution is not fixed, i,e such changes are often random and does not follow a fixed pattern.



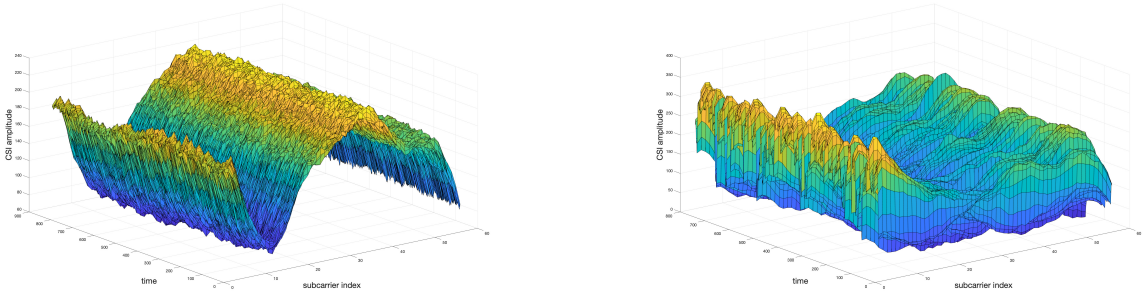
(a) Empty capture (b) Drifted Empty Capture

Fig. 5.2 Variation of CSI mesh in the presence of drift

5.5 Analysis of phase and magnitude data

We perform separate analysis of the phase and the magnitude of the CSI and outline the important features to be used for classification purposes. The magnitude information from the CSI is a 3 dimensional matrix with cardinality $\mathcal{I} \times \mathcal{M} \times \mathcal{S}$ as mentioned in section 5.2, for the application of localization we used $\mathcal{I} = 56$, $\mathcal{M} = 800$ and $\mathcal{S} = 16$, which corresponds to one minute data for each location. Since the CSI provides the channel frequency response of each individual subcarrier over all the spatial streams, the magnitude information has been primarily considered as the main feature to be used for the localisation purposes. Figure 5.3 shows the variation of the CSI magnitude with an empty one minute capture

versus a one minute capture with walk(motion) in the same room.



(a) No motion (b) Walk

Fig. 5.3 Variation of CSI mesh with motion and no motion in a room

The raw phase information from the CSI data, specifies the phase of the signal path between a single transmit-receive antenna pair. Since we have 4 receiving and 4 transmitting antennas, it results in 16 combination, that corresponds to 16 streams of the CSI data. Table 5.1 outlines the receiving and the transmitting antennas along with the stream number to which each pair corresponds to.

Table 5.1 Stream configuration of Tx and Rx antennas

Rx \Tx	1	2	3	4
1	1	5	9	13
2	2	5	10	14
3	3	7	11	15
4	4	8	12	16

The raw phase information alone for CSI data, does not provide sufficient information for localization because the phase is a result of the unsynchronized timing between the receiver and the transmitter. In order to make phase information useful for localization, the raw phase information needs to be sanitized and calibrated to remove any phase offset. The calibration is done by applying a linear phase transformation to extract essential phase information. Such transformations have been successfully applied in [45] for obtaining a

calibrated phase to identify line of sight (LOS) for passive human movement detection using WiFi signals. The measured raw phase contains the following information

$$\widehat{\angle CSI_i} = \angle CSI_i + 2\pi \frac{m_i}{N} \Delta t + \phi + Z. \tag{5.2}$$

Here $\angle CSI_i$ represents the true phase of the data and the $\widehat{\angle CSI_i}$ represents the measured phase for subcarrier i. Z is a random noise that arises from the measurement of CSI. Δt represents the time lag, ϕ is the unknown phase off set and N is the size of the fast Fourier transform and m is the index of the subcarrier. Since the knowledge about Δt and β is unknown this makes it impossible to obtain the true phase information of the underlying CSI data. The linear transformation of the phase data is obtained based by calculating a gradient of the phase and the offset, using the following equation

$$\texttt{gradient phase} = \frac{\widehat{\angle CSI_{56}} - \widehat{\angle CSI_1}}{m_{56} - m_1}. \tag{5.3}$$

The gradient of the offset value is computed using the following equation.

$$\texttt{gradient offset} = \frac{1}{30} \sum_{i=1}^{56} \widehat{\angle CSI_i}. \tag{5.4}$$

The calibrated phase is therefore obtained by subtracting the `gradient phase` and the value of `gradient offset` from the measured phase value in order to obtain the calibrated phase value.

$$\widehat{\widetilde{\angle CSI_i}} = \widehat{\angle CSI_i} - \texttt{gradient phase - gradient offset}. \tag{5.5}$$

We used a system with a 5GHz band channel where it has been established in [46] that the phase difference between consecutive receiving antennas is stable. Although [46] mainly uses this information in order to obtain a periodic signal with the same frequency as the breathing signal, the fact that the phase difference between consecutive antennas lead to stable phase spread is useful for localization application as well. Hence after performing linear phase transformation phase differencing between consecutive antennas are performed. As outlined in Table 5.1, we have 4 receiving antennas and 4 transmitting antennas, hence the link between each pair of transmit and received antennas forms each of the index of the

stream. Therefore in order to compute the phase difference we obtain a difference between stream (1,2), stream (2,3) and stream (3,4) which correspond to the links between single receiving antennas and all 4 transmitting antennas. Figure 5.4 shows the relative stability of the phase difference from consecutive antennas.

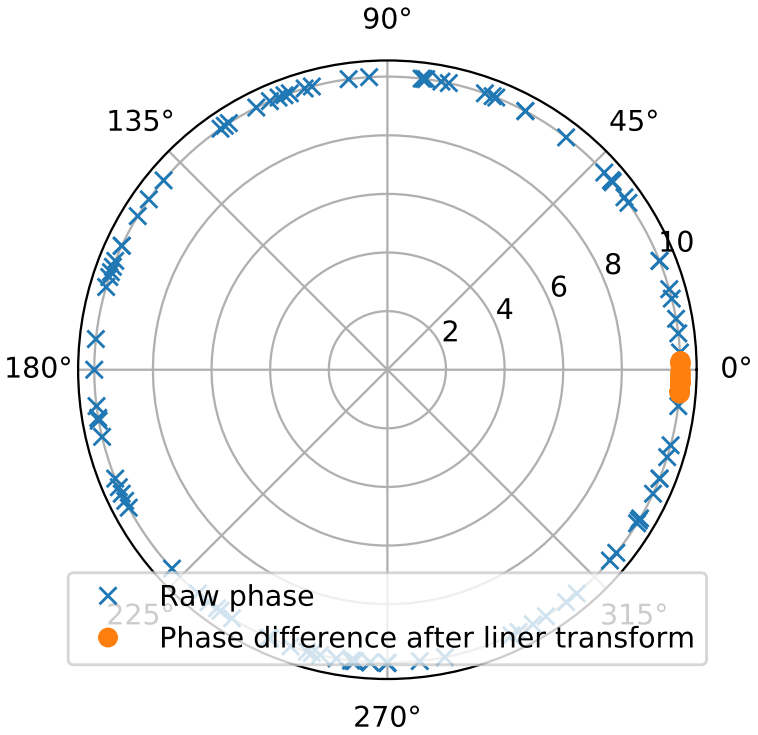


Fig. 5.4 Comparison between CSI phase data from a single antenna and the phase difference between two consecutive antennas of the 2nd subcarrier for 100 consecutive received packets

Chapter 6

Experimental Analysis

In this section we analyse the performance of different learning algorithms. For this study, the datasets are obtained from three different apartments (Apt 1, Apt 2 and Apt 3) with different numbers of rooms. The data is collected by performing a one minute walk in each of the rooms. Each room is assigned one label and the problem becomes a standard classification problem with class labels defined by the rooms where a subject walked.

6.1 Data preprocessing

The raw CSI data containing one minute recordings of a single person walking in each of the rooms with in an apartment is collected. The CSI data are captured with a sampling rate of 20 `packets/second`. We apply the following filtration process in order to sanitize the raw CSI magnitude data.

- **RSSI drop filter:** The RSSI drop filter scans through successive received packets and measures the RSSI values. It then discards those packets where the corresponding RSSI values exceed a threshold δ. We used a value of $\delta = 50$ for all the data. This sudden peak in RSSI can be the result of constructive interference, multipath fading and temporal dynamics.

- **Normalization:** The data consists of multiple subcarriers and streams which correspond to different links between the transmitter and the receiver, at each point in time, they can take values which are scaled to a wide range. Hence after discarding

the packets based on RSSI corrections, we perform normalization of the CSI amplitudes to a predefined range. The L_2 norm of the CSI vector is then calculated for each of the CSI vectors in order to re-scale their values to the predefined range.

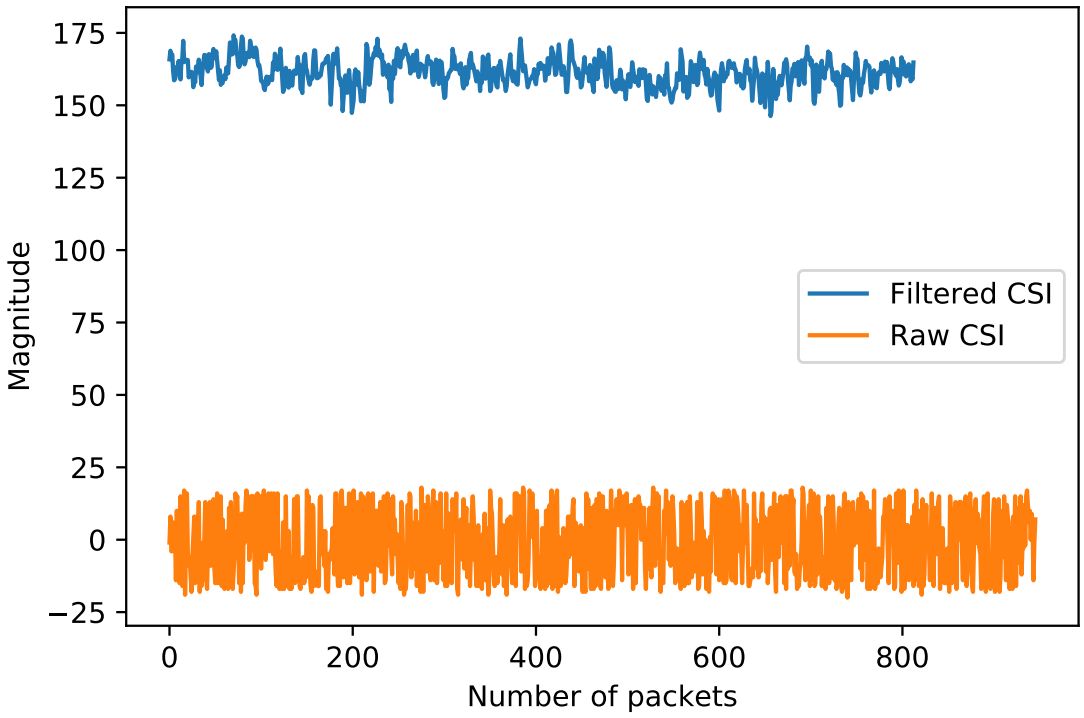


Fig. 6.1 Comparison of the raw and rescaled csi magnitudes after filtering

Figure 6.1 shows the difference between the raw and the filtered CSI magnitude data for a single subcarrier number 2 and the stream number 2.

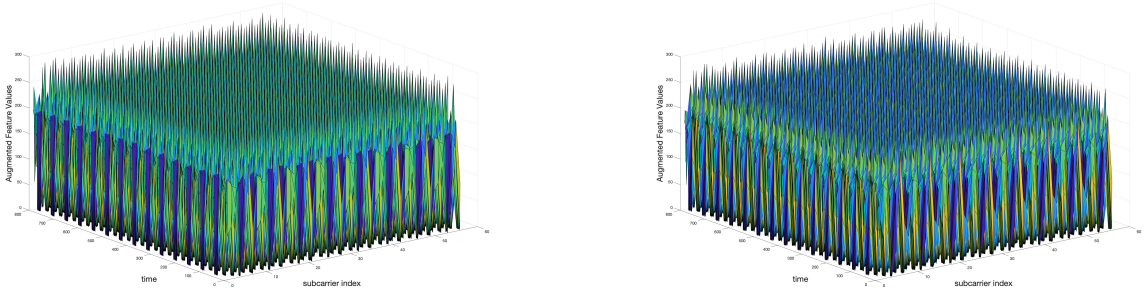
6.2 Feature extraction

For the localization task, CSI magnitude data used as training features yield a high predictive accuracy. However, when drifts occur the predictive accuracy of the model falls drastically and the distribution of the features change in a random manner. In order to ameliorate this reduction of predictive performance, we consider both an alternative feature

space that is less affected by drifts and in parallel we study the performance of different learning algorithms. The phase difference information is less affected by drifts since it provides an information of the angle of arrival of the signal. However, phase data used as a feature alone performs poorly for localization. We therefore propose an augmented feature space, that consists of both the elements of magnitude (which better characterises variation of WiFi mesh due to movements in different locations) and phase difference (which is less affected by drifts), this form of feature augmentation increases the dimensionality of the dataset, where the classifier will tend to suffer from the curse of dimensionality. It happens when beyond a certain point adding more features to the dataset hinders the accuracy of the predictive models. Therefore we select the first 4 streams for the magnitude (that corresponds to the transmit and the receiving link combination of all the receiving antennas with 1 transmitting antenna only.

The feature obtained from magnitude information is a 3-dimensional matrix with size $56 \times 800 \times 4$. The phase difference data consists of the phase difference information from the first 4 streams having a size of $56 \times 800 \times 3$. We consider the phase difference between stream 1-2,stream 2-3 and stream 3-4 as they correspond to the links from a single transmitter to all 4 of the receiving antennas. A phase correction is then performed for the phases such that their values lie with in the range $(-\pi, \pi)$. We then use a Hampel filter in order to remove the DC component of the phase information and to detrend the phase data. For our Hampel filter we use a large sliding window of 300 samples and with a small threshold of 0.01 in order to get the general trend of the data. Once the trend has been computed, it is then removed from the the phase difference information. Then, we further leverage Hampel filter with a smaller sliding window of 15 samples and a threshold of 0.01 in order to remove the high frequency noise from the streaming phase data. The augmented feature space therefore consists of $(224 + 168) \times 800 = 392 \times 800$.

Even though after combining only 4 streams from the magnitude and 3 streams from the phase, the combined feature space is still high dimensional, we found through repetitive experiments that using Principal Component Analysis (PCA) for dimensionality reduction yields very poor classification accuracy even when appropriate components are chosen based on explained cumulative variance analysis. Hence, we do not perform any dimensionality reduction in our dataset. Figure 6.2 shows that with the combined feature space, the data distributions which are 9 hours apart do not change drastically.

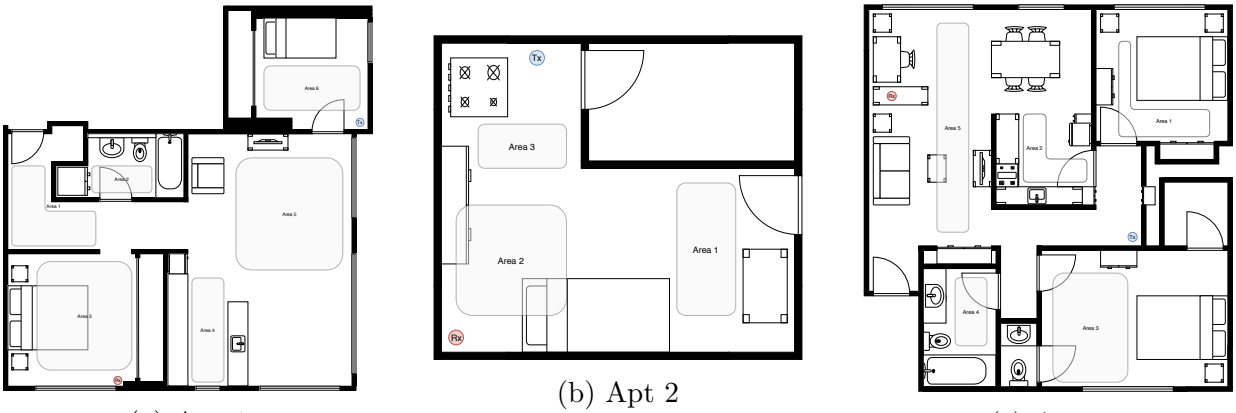


(a) Empty capture (b) Empty capture (9 hours apart)

Fig. 6.2 Augmented feature space getting less affected by drifts

6.3 Experimental setup

We use two different devices for the experiments the Tx and the Rx which corresponds to the routers for transmission and the reception respectively. Both of the devices are placed further apart in the apartment and the experiment begins by taking an empty capture at the first instance. This empty capture corresponds to no motion at any of the rooms in the apartment. We next proceed towards capturing one minute data by walking in each of the rooms of the apartment, respectively, in order to obtain annotated data. The data is then collected and processed and converted to the augmented feature space as described in Section 6.2, respectively. For Apt 1 we captured 5 rounds of data each of which is roughly 30 minutes apart. Although drift is more apparent for measurements taken over longer intervals, for measurements associated with Apt1, we force a channel switch (abrupt drift) before collecting the 5th round by switching the devices off. This ensures that a drift has occurred since drifts are expected during a channel change. For Apt 2 we perform a more rigorous measurement and hence take 6 rounds of data, where we captured 3 rounds which are 12 hours apart and the last 2 rounds which are 2 days apart. For Apt 2 we take the measurements in such a diverse manner so that the effect of drift can be thoroughly studied. Finally for Apt 3 , we capture 3 rounds of data where round 1 and round 2 are data which are 6 hours apart and round 3 is captured at an interval of 12 hours from round 2. Through all of the experiments we ensured that the position of Tx and Rx remains fixed. Figure 6.3 shows the layout of the three apartments in which the experiments were conducted. The areas marked (Area 1, Area 2, etc) are the corresponding labels in the



(a) Apt 1 (b) Apt 2 (c) Apt 3

Fig. 6.3 Layout of the apartments where experiments are conducted

dataset that included a walk.

In order to validate whether walking in different rooms of an indoor space actually correspond to distinguishable clusters from WiFi propagation perspective, we do an unsupervised clustering analysis over the dataset to evaluate our location partitioning. From the elbow analysis of the unsupervised clustering we found that WiFi mesh distortions can also be categorized in an unsupervised manner, where the number of clusters correspond to walking or physical activities in number of areas of the apartment.

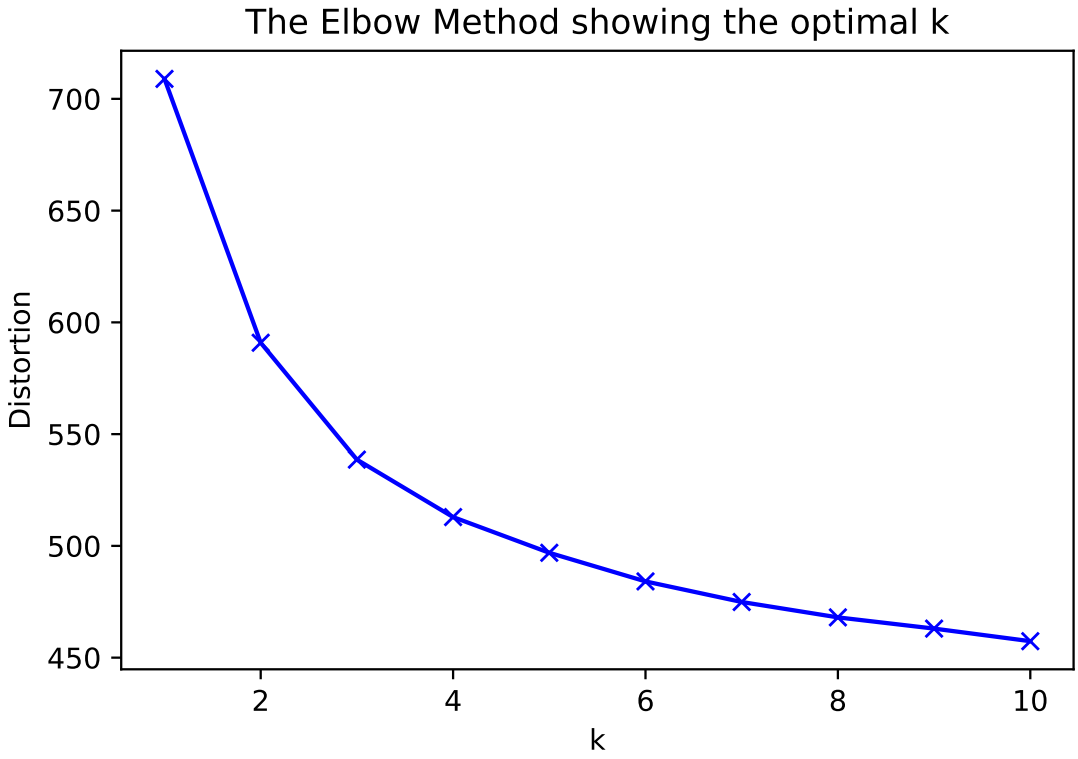


Fig. 6.4 Elbow analysis if clusters formed in Apt 1

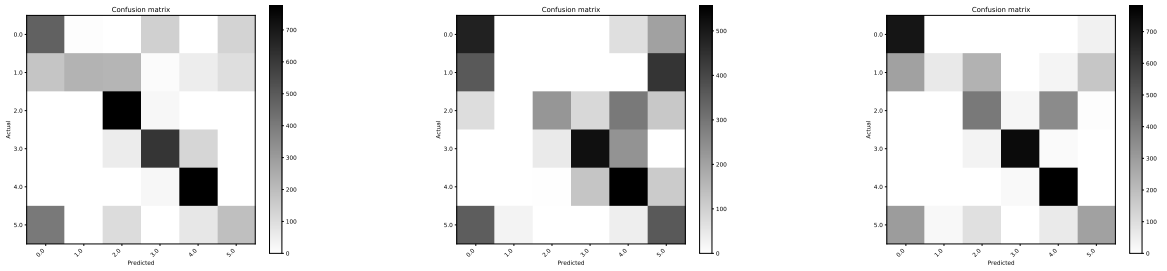
Figure 6.4 presents the elbow analysis done in Apt 1 which consists of labelled data for 6 locations. For the elbow analysis described in Figure 6.4 we can see that there is no significant reduction in distortion when the number of clusters are increased from 6 to 8. This provides an indication that movements in different locations indeed leads to different data distribution that can be classified.

For our experiments, the rounds containing no drifts are used to train a standard classifier model. During training the rounds containing both abrupt and gradual drifts are held out as test sets and we analyse the performance of the predictive models. We successively train learning algorithms with different different features (magnitude only, phase only and the combined feature space) and make a relative comparison among them. Before feeding the data in the classifier model, we take a rolling variance across the packets with a window size of 40. Through experiments we found that computing rolling variance over a window size of 40 significantly improves the performance of all the base classifiers. In the sequel we

outline the algorithms use along with the methodologies using which we train each model.

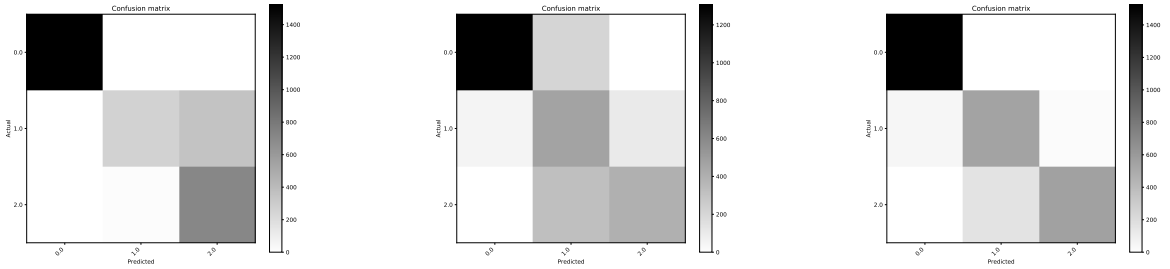
6.4 Random forest classifier

The random forest classifier proposed in [47] is a meta estimator, that fits multiple decision trees to the training dataset. Since it is a multiclass classification problem, the mode of the decision stubs are taken in order to obtain the predicted labels.The random forest algorithm is one of the most effective ensemble classifiers which takes an ensemble of highly decorrelated trees. We obtain an Accuracy Weighted Ensemble (AWE) approach [48] where a new classifier is trained on new incoming data and is used to evaluate the performance of the existing classifier in the ensemble. Such methods are quite popular for handling concept drifts in streaming data. Our random forest classifier consists of 100 decision trees, the minimum number of samples in order to split the decision tree is chosen to be 40 packets, as this corresponds to roughly 2 seconds of data. Once the model is trained and evaluated on test sets, a measure of confidence score on the predicted labels is calculated based on the proportion of the trees that agree with the predicted label. This confidence of the model acts as a sudo indicator for drifts, i,e when labels are predicted with a low confidence score, it gives an indication of drifts.



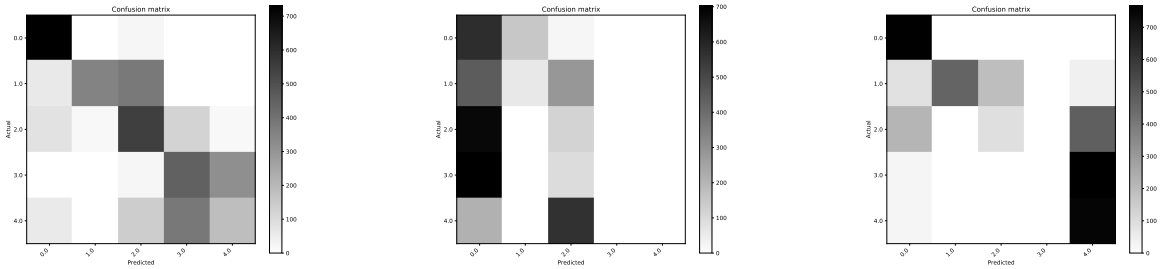
(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.5 Confusion matrix for different training features for Apt 1



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.6 Confusion matrix for different training features for Apt 2

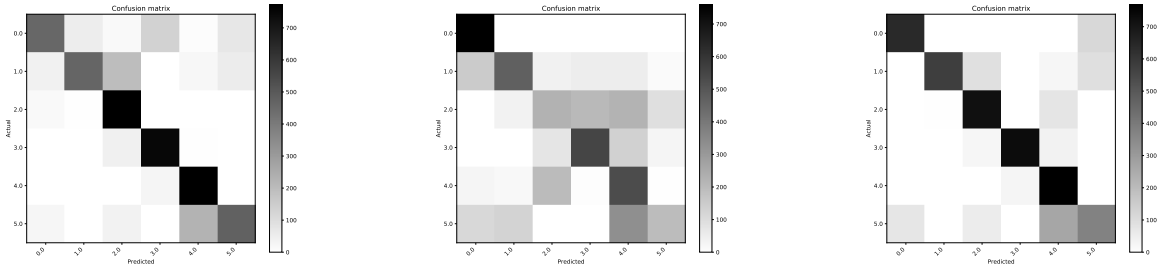


(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.7 Confusion matrix for different training features for Apt 3

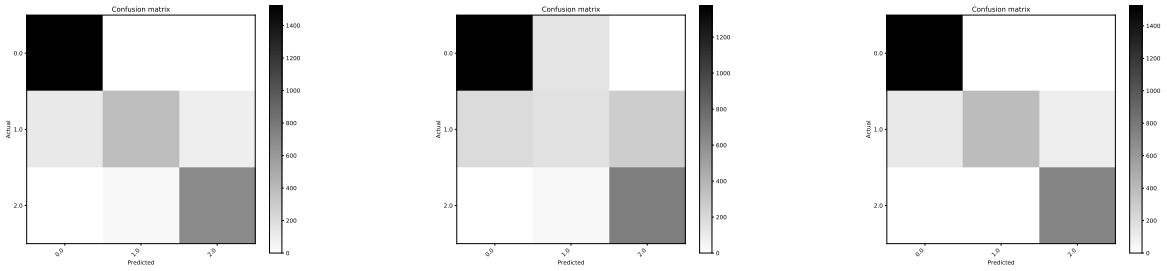
6.5 Support vector machines

A support vector machine (SVM) [49] is a non probabilistic binary classifier that finds a linear decision boundary while maximizing the the margin (also known as support vectors) along the decision boundary as far as possible. Since we have a multiclass classification problem the svm converts this multi-class classification task into multiple classification task. SVM is effective for high dimensional data and is a memory efficient algorithm since it only requires a subset of the data in order to calculate the support vectors. We trained our SVM to find a non linear decision boundary in order with an order 3 polynomial kernel. We observed that training SVM require more time compared to Random Forest. Unlike random forest SVM is particularly sensitive to the feature scales, thus we standardize the data with zero mean and unit variance.



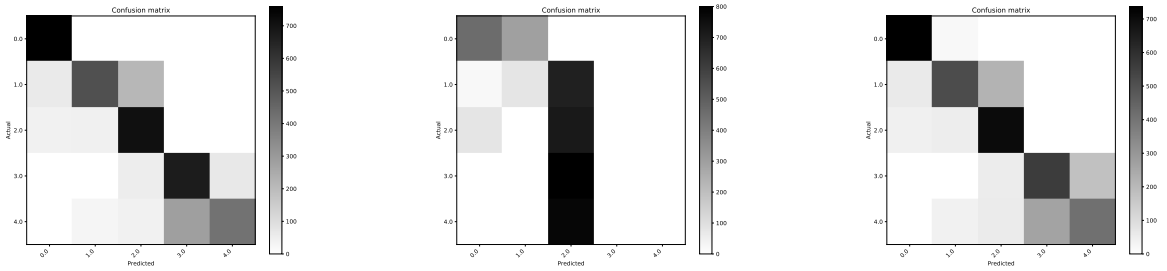
(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.8 Confusion matrix for different training features for Apt 1



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.9 Confusion matrix for different training features for Apt 2



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.10 Confusion matrix for different training features for Apt 3

6.6 Incremental SVM

Incremental learning framework allows the input data to continuously extend the existing knowledge of the model. Incremental SVM have been proposed in [50] where the model is trained incrementally by discarding all the previous information of the data except for their support vectors. Thus the model can adapt to new data without forgetting its existing knowledge, in such a way so that it will adapt quickly to a very slow change in distribution of the data. For our learner, we use a SGD classifier with hinge loss and L2 regularizer, this results in an SVM that can be updated incrementally. Although such framework has been widely used for training large amounts of streaming data, we focus on the fact that streaming data contains gradual drifts where such a learning framework can adapt to the underlying change in data distribution. Figure 6.11 shows the training procedure in an incremental framework.

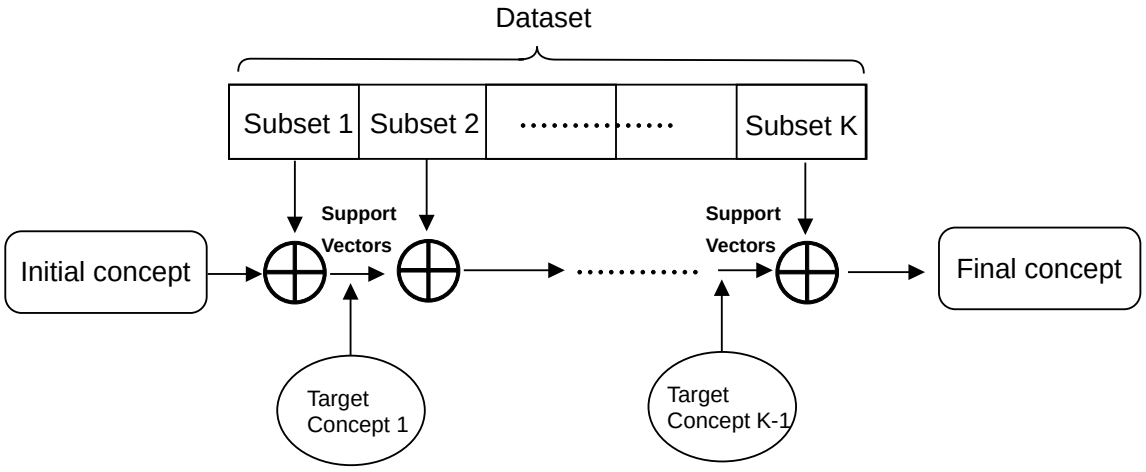
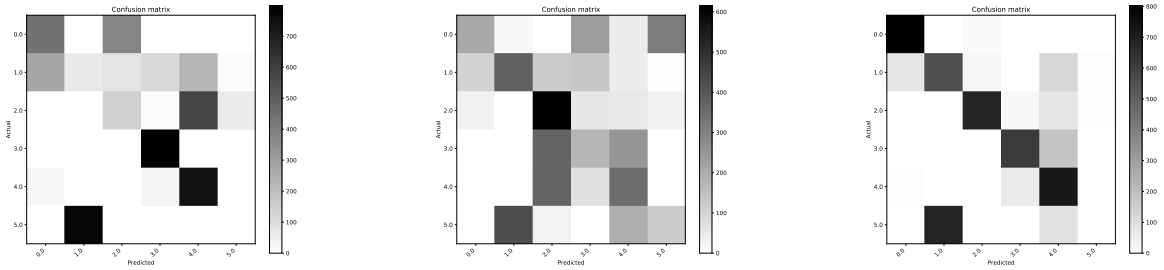
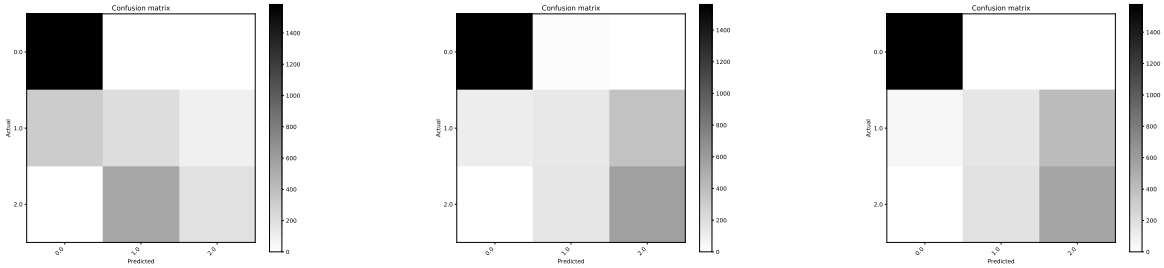


Fig. 6.11 Training stages of incremental learning



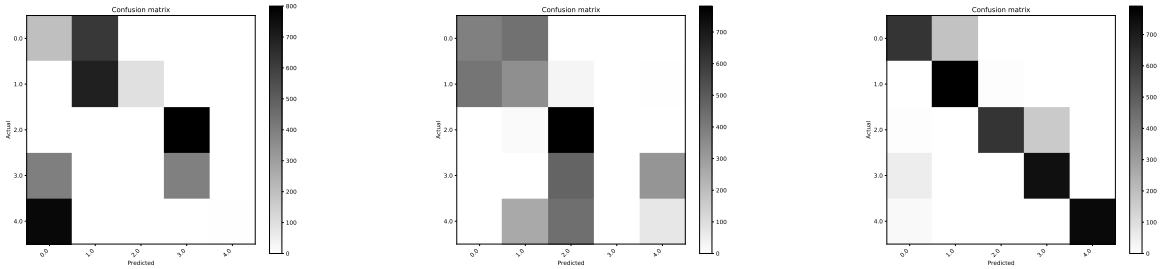
(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.12 Confusion matrix for different training features for Apt 1



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.13 Confusion matrix for different training features for Apt 2



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.14 Confusion matrix for different training features for Apt 3

Table 6.1 shows the performance of different learning algorithms for CSI features incorporating magnitude only, phase only and the proposed augmented feature space. The table

Table 6.1 Accuracy in % of the learning algorithms for different feature space

Datasets	Features (Magnitude only)				Features (Phase only)				Augmented Features			
	SVM	RF	Inc.SVM	LSTM	SVM	RF	Inc.SVM	LSTM	SVM	RF	Inc. SVM	LSTM
Apt1	77.94	59.84	54.85	62.36	55.09	46.04	38.92	38.42	**80.52**	60.31	70.67	69.89
Apt2	90.2	87.8	67.26	53.36	78.8	88.67	74.33	64.34	**92.0**	89.8	75.12	71.85
Apt3	78.24	60.5	35.6	34.16	32.95	36.12	38.8	35.8	74.53	63.95	**83.35**	78.49

shows that for all the learning algorithms the augmented feature space performs better and is more resistant to drifts. Also we note that in case of streaming data, the performance of incremental SVM described is consistently better for the augmented feature space. Thus we show that in case of large incoming data, the augmented feature space presents more robust features in terms of representing the WiFi CSI for localization.

6.7 Analysis of deep learning algorithms

Deep learning for indoor localization with WiFi CSI data had been successfully demonstrated in [38, 33, 44]. Existing techniques perform active localization by training an Restricted Boltzman Machine (RBM) layer by layer or use Convolution neural network (CNN) in order to perform the offline training phase. To the best of our knowledge device free passive indoor localization using WiFi CSI has not been studied well. We mainly analyze two algorithms: Deep convolution neural network [51] and recurrent neural network with long short term memory [52]. We train these models with the magnitude of the CSI only since deep neural networks extract useful features automatically.

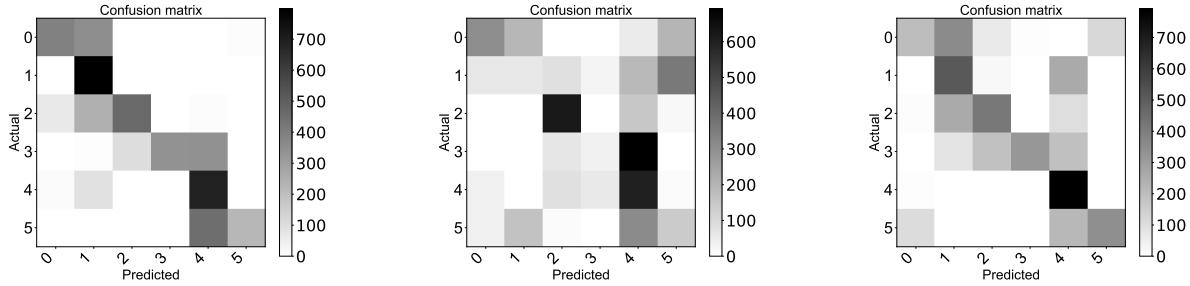
6.7.1 Deep convolution neural network

Deep convolution neural network applies convolution operations to an input image in order to extract essential features that are embedded in the training set. A non linear activation function is applied followed by the convolution operation that is performed at each layer of network. We show that the WiFi CSI matrix can be projected as a $30 \times 30 \times 3$ images where each dimension corresponds to the number of subcarriers, number of packets and the number of streams respectively. For the magnitude data in order to construct an RGB image, we only consider 3 streams. Instead of using stacked convolution layers, followed by max pooling and dropout to reduce over fitting we use inception module proposed in

[53]. In a typical CNN structure, each layer is chosen either to perform convolution with defined filter size, max pooling or dropout operation to the output of the previous layer. An inception module suggests to use all of them. Thus instead of adding particular convolution filter size layer by layer, inception module suggests to include all 1×1, 3×3 and 5×5 filters and perform convolution on the output of the previous layer. Through experimentation we found that although the CSI data can be projected as an RGB image, vanilla CNN or CNN with the inception module fails to learn and capture the temporal variation of the data with time. Therefore such models although appeared in the literature for active localization for phase fails to capture the spatio temporal differences and learn for passive localization.

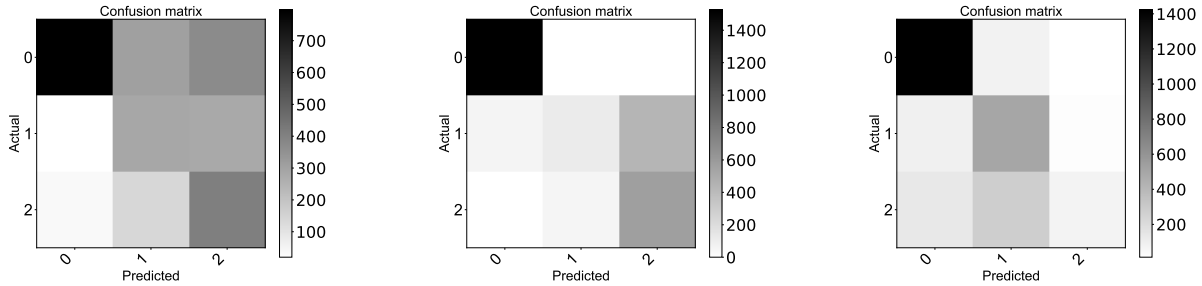
6.7.2 Classification with LSTM

A long short term memory (LSTM) is a class of recurrent neural network (RNN) of specific type. RNN along with LSTM module were used to model temporal sequences, they are specifically used for sequences that have their long range time dependencies. The CSI data are obtained with a sampling rate of 20 `packets/second`. Since localization data corresponds to a 1 minute walk in each room, and classification on the data is usually done on a packet by packet basis. We use RNN with an LSTM module for many to one sequence modelling. The RNN + LSTM module performs the classification after it sees every 80^th input which roughly corresponds to 4 seconds of data. So far LSTM with WiFi CSI for indoor localization has not been considered in the literature. Figure 6.15, 6.16 and 6.17 shows the confusion matrix with LSTM. The results are consistent with our previous hypothesis where inclusion of both the phase and the magnitude of the CSI yield better results during drifts. LSTM on the other hand still under performs during abrupt drifts (Apt 1) compared to SVM or the incremental SVM. Feeding data with a sequence length of 80 packets essentially reduces the number of training examples being fed to the LSTM module since only minute capture is obtained in each of the location. We hope that with better hyper parameter tuning and with more examples the advantage of deep learning models would be noticeable.



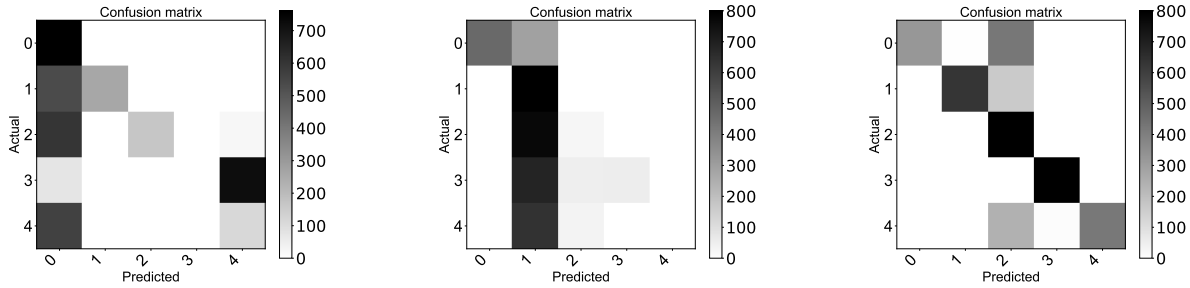
(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.15 Confusion matrix with LSTM for different training features for Apt 1



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.16 Confusion matrix with LSTM for different training features for Apt 2



(a) Magnitude only (b) Phase only (c) Magnitude+Phase

Fig. 6.17 Confusion matrix with LSTM for different training features for Apt 3

6.8 Discussion

The experimental results show that all the algorithms considered in this section performs significantly better in the augmented feature space. In the presence of gradual drifts, use of incremental SVM with augmented feature space is suitable for passive localization. In order to treat the streaming WiFi CSI as sequential data, we presented a LSTM classifier, to the best of our knowledge, this is the first time LSTM has been used for wifi indoor localization. RNN with an LSTM cell also performs better with the augmented feature space compared to phase only or the magnitude only feature space. Although drift is still an issue with streaming wireless signals, the proposed feature space is very robust to both the gradual and the abrupt concept drift. We show that deep convolution neural network fails to capture the temporal differences when the WiFi CSI is projected as an image for passive WiFi localization. The WiFi CSI data although high dimensional can be easily tackled by simple learning algorithms like SVM or Random Forest, however these algorithms are not designed to deal with concept drifts well. Hence research in this domain should focus more on finding appropriate feature space that is resistant to drifts.

Chapter 7

Conclusion

We have structured this thesis into two parts. In the first part we study reinforcement learning algorithms for partially observable markov decision process and in the second part we highlight the problem of concept drifts in WiFi data and propose an augmented feature space in order to handle drifts. In the first part we have shown that policies represented with finite state controllers converges to a local minima. We analysed the performance of different methods for performance gradient estimation. We studied an alternative algorithm known as Renewal Monte Carlo and extended them for policy evaluation in POMDPs with finite state controllers. We showed that when the decision maker has perfect observation of the renewal state the solution converges with less number of samples. We make a clustering analysis and showed that for different runs of our experiments, the sample paths converge to the same local optima. A comparison of Recurrent Policy gradients and policies with finite state controllers are made, and we observed that both the methods learn and converge to a local solution.

In the second part we highlighted that concept drift widely degrades the performance of learning algorithms for WiFi indoor localization. A detail study of the phase and magnitude of WiFi CSI is presented and a feature space is constructed incorporating both the phase and the magnitude. We also analysed incremental learning framework which and showed that such a framework is useful for streaming data and can tackle gradual drifts.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[2] 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, p. 484, 2016.

[3] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *AAAI*, vol. 94, pp. 1023–1028, 1994.

[4] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[5] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems," in *AAAI/IAAI*, pp. 541–548, 1999.

[6] N. Ghourchian, M. Allegue-Martinez, and D. Precup, "Real-time indoor localization in smart homes using semi-supervised learning.," in *AAAI*, pp. 4670–4677, 2017.

[7] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[8] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[9] C. White and D. P. Harrington, "Application of jensen's inequality to adaptive suboptimal design," *Journal of Optimization Theory and Applications*, vol. 32, no. 1, pp. 89–99, 1980.

[10] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations research*, vol. 26, no. 2, pp. 282–304, 1978.

[11] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable markov decision processes," in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pp. 54–61, Morgan Kaufmann Publishers Inc., 1997.

[12] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable markov decision processes," *Journal of Artificial Intelligence Research*, vol. 14, pp. 29–51, 2001.

[13] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for pomdps," in *IJCAI*, vol. 3, pp. 1025–1032, 2003.

[14] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[16] H. Yu, "A function approximation approach to estimation of policy gradient for POMDP with structured policies," in *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, 2005.

[17] D. Aberdeen and J. Baxter, "Scaling internal-state policy-gradient methods for pomdps," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pp. 3–10, 2002.

[18] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[19] G. H. Mealy, "A method for synthesizing sequential circuits," *The Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, 1955.

[20] E. F. Moore, "Gedanken-experiments on sequential machines," *Automata studies*, vol. 34, pp. 129–153, 1956.

[21] J. Subramanian and A. Mahajan, "Renewal monte carlo: Renewal theory based reinforcement learning," *arXiv preprint arXiv:1804.01116*, 2018.

[22] W. Feller, *An introduction to probability theory and its applications*, vol. 2. John Wiley & Sons, 2008.

[23] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Logic Journal of the IGPL*, vol. 18, no. 5, pp. 620–634, 2010.

[24] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[25] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[26] J. D. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.

[27] R. A. McCallum, "Overcoming incomplete perception with utile distinction memory," in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 190–196, 1993.

[28] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable markov decision processes," in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pp. 54–61, Morgan Kaufmann Publishers Inc., 1997.

[29] D. Pelleg, A. W. Moore, *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters.," in *Icml*, vol. 1, pp. 727–734, 2000.

[30] J. Gama and G. Castillo, "Learning with local drift detection," in *International Conference on Advanced Data Mining and Applications*, pp. 42–55, Springer, 2006.

[31] M. Basseville, I. V. Nikiforov, *et al.*, *Detection of abrupt changes: theory and application*, vol. 104. Prentice Hall Englewood Cliffs, 1993.

[32] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine learning*, vol. 23, no. 1, pp. 69–101, 1996.

[33] X. Wang, L. Gao, S. Mao, and S. Pandey, "Deepfi: Deep learning for indoor fingerprinting using channel state information," in *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, pp. 1666–1671, IEEE, 2015.

[34] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pp. 205–218, ACM, 2005.

[35] P. Bahl, V. N. Padmanabhan, and A. Balachandran, "Enhancements to the radar user location and tracking system," *Microsoft Research*, vol. 2, no. MSR-TR-2000-12, pp. 775–784, 2000.

[36] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 775–784, Ieee, 2000.

[37] K. Wu, J. Xiao, Y. Yi, M. Gao, and L. M. Ni, "Fila: Fine-grained indoor localization," in *INFOCOM, 2012 Proceedings IEEE*, pp. 2210–2218, IEEE, 2012.

[38] X. Wang, L. Gao, and S. Mao, "Phasefi: Phase fingerprinting for indoor localization with a deep learning approach," in *Global Communications Conference (GLOBECOM), 2015 IEEE*, pp. 1–6, IEEE, 2015.

[39] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You are facing the mona lisa: spot localization using phy layer information," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 183–196, ACM, 2012.

[40] X. Chen, C. Ma, M. Allegue, and X. Liu, "Taming the inconsistency of wi-fi fingerprints for device-free passive indoor localization," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pp. 1–9, IEEE, 2017.

[41] F. Zafari, A. Gkelias, and K. Leung, "A survey of indoor localization systems and technologies," *arXiv preprint arXiv:1709.01015*, 2017.

[42] Z. Yang, Z. Zhou, and Y. Liu, "From rssi to csi: Indoor localization via channel response," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, p. 25, 2013.

[43] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[44] X. Wang, X. Wang, and S. Mao, "Cifi: Deep convolutional neural networks for indoor localization with 5 ghz wi-fi," in *Communications (ICC), 2017 IEEE International Conference on*, pp. 1–6, IEEE, 2017.

[45] C. Wu, Z. Yang, Z. Zhou, K. Qian, Y. Liu, and M. Liu, "Phaseu: Real-time los identification with wifi," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pp. 2038–2046, IEEE, 2015.

[46] X. Wang, C. Yang, and S. Mao, "Phasebeat: Exploiting csi phase data for vital sign monitoring with commodity wifi devices," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pp. 1230–1239, IEEE, 2017.

[47] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[48] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 226–235, AcM, 2003.

[49] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[50] N. A. Syed, S. Huan, L. Kah, and K. Sung, "Incremental learning with support vector machines," 1999.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[52] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.