North East University Bangladesh

Department of Computer Science and Engineering

Reinforcement learning for Flight Ticket Pricing

By

Md. Abu Shahan
Reg. No. 170103020017
BSc. (Engg.) in CSE
4th Year 3rd Semester

Nabil Islam
Reg. No. 170103020060
BSc. (Engg.) in CSE
4th Year 3rd Semester

Supervised By

Al Mehdi Saadat Chowdhury

Assistant Professor

Department of Computer Science and Engineering

17th February 2021

Reinforcement learning for Flight Ticket Pricing

A Project submitted to the Department of Computer Science and Engineering,
North East University Bangladesh, in partial fulfillment of the requirements,
for the degree of Bachelor of Science in Computer Science and Engineering

By

Md. Abu Shahan
Reg. No. 170103020017
BSc. (Engg.) in CSE
4th Year 3rd Semester

Nabil Islam
Reg. No. 170103020060
BSc. (Engg.) in CSE
4th Year 3rd Semester

Supervised By

Al Mehdi Saadat Chowdhury

Assistant Professor

Department of Computer Science and Engineering

17th February 2021

Recommendation Letter from Project Supervisor

These Students, *Md. Abu Shahan, Nabil Islam,* whose project entitled *"Reinforcement learning for Flight Ticket Pricing",* is under my supervision and agrees to submit for examination.

Signature of the Supervisor :

Al Mehdi Saadat Chowdhury
Assistant Professor
Department of Computer Science and Engineering
North East University Bangladesh

Qualification Form of BSc (Engg) Degree

Name: Md. Abu Shahan, Nabil Islam

Project Title: Reinforcement learning for Flight Ticket Pricing.

This is to certify that the project is submitted by the student named above in February, 2021. It is qualified and approved by the following person and committee.

Head of the Dept.	**Supervisor**
Tasnim Zahan	Al Mehdi Saadat Chowdhury
Assistant Professor & Head	Assistant Professor
Department of CSE	Department of CSE
North East University Bangladesh	North East University Bangladesh

Abstract

From the beginning of the 20th century Airplane is the world's most expensive transportation system. The price of the air tickets often fluctuate before the departure date. This paper reports a study on the behavior of the air ticket of a particular period of time to predict the optimal time to buy the tickets. We prepared a method to mining the air fare data using Q-Learning algorithm. We also prepared another method using Deep Q-Network. Both of models can predict the optimal time to buy the air tickets. Before implementing our models we studied a lot of things about Reinforcement Learning, Markov Decision Process (MDP), Q-Learning and Deep Q-Network.

Keyword: Air ticket, Reinforcement Learning, Markov Decision Process (MDP), Q-Learning, Deep Q-Network, Air ticket price.

Table of Content

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Air traveling is the most expensive transportation system in the world. The price of air tickets always fluctuate. From our recent study, the price of air tickets changes 5-6 times a day in the same category. Airline companies often use complex and hidden policies to change ticket prices over time. It frequently changes the prices per seat, based on some factors including seasonality, availability of seats and more other price changing factors. On holidays the price of the air tickets went very high. When so many seats are available then tickets are sold at a very cheap price. Competitive manner of the airlines is also a major factor for often price changing. The airlines use some software's to compute ticket prices on any given day, but the algorithms they use are very complex that guard trade secrets. So it is very hard to predict the optimal price for the general passengers.

However, it is difficult to estimate when the ticket price will be the lowest. The passengers always want to buy tickets at a low price. The passengers can save their money if they choose to buy a ticket when its price is the lowest.

It is a big problem to determine when is the best time to buy the target ticket for a customer. But analyzing historical data by a suitable model can predict the optimal time to purchase the target ticket.

This paper reports on to give an approach to predict air ticket price. We will try it with 3 different datasets where we will predict all flights of 2 domestic routes and an international route by using the reinforcement learning approach. It is a time consuming task to collect and preprocess a huge number of data. And use of reinforcement learning methods on this project is also challenging for us. Those things motivate us to go for this task.

1.1 Why Optimization of Predicting Air Ticket Price is Important?

Airplanes are the most popular and time consuming transport for long distance traveling. According to the Federal Aviation Administration [ffa.gov], every day more than 44,000domestic and international flights carries 2.7 million passengers across the world. There are over 5,000 commercial aircraft companies in the world which make this field more competitive.

Every company often uses complex policies to vary product prices over time to maximize their revenue. The airline industry is one of the most sophisticated where its use of a complex pricing manner in an approach to maximize its profit. Those airlines have many fare classes for seats on the same flight. They use different sales channels such as travel agents which make it frequently vary the price per seat over time. Based on some factors including seasonality, availability of seats, and competitive moves by other airlines, and more the price of those tickets goes up and down. Ticket prices can fluctuate by hundreds of dollars from day to day, and predicting the change can seem virtually impossible. Leaving many customers to simply buy whatever ticket they need at the time. There is a logic to these changes, however, although so many factors go into determining the shifts that it can be difficult to predict when the lowest fares will be available.

Airlines look at two main types of customers: early purchasers and last-minute purchasers. An early purchaser generally can wait some time to find the best deal on a flight, but often will simply buy a relatively affordable ticket, since predicting when the lowest price point is can be too difficult. Last-minute purchasers often pay full price for a ticket and do not have the flexibility of waiting for cheaper deals. As a result, prices will tend to spike radically within a few days of a flight, since airlines know some consumers have no other option. But there is a chance for the early purchasers to go for a good deal.

Availability of seats can be a major factor for price fluctuation. We often see when most of the seats are available than the price of the ticket

Airlines always are trying to maximize their profit based on the forecast demand for a destination. As long as their predictions do not change, prices typically will not change until immediately before the travel date. Occasionally, however, a destination's popularity will increase for some reason, a big event like FIFA World Cup, Olympic Games or from some other cause.

For example, flight ticket prices to cities that are hosting the 2018 World Cup have grown an average 230 percent according to a research by a global business website Statista.com. Prices to Volgograd, which will host group stage matches, increased the most. Instead of the 6,300 rubles ($109) for flights from Moscow at the beginning of summer, football fans will have to pay 30,000 to 40,000 rubles for flights near to match days. Similarly, tickets to Yekaterinburg from Moscow, Sochi and Kaliningrad, which usually cost 7,500 rubles in the summer, spiked to 20,000 to 40,000 rubles near match days. When this type event happens, airlines might predict increased demand and raise their prices accordingly. This kind of Global event can sometimes cause aircraft to be removed from service. When this happens, overall capacity for a route is reduced, leaving fewer seats to be filled and that flights will be fuller and will increase ticket prices.

It is very difficult to predict when tickets for a particular flight will be at their lowest price, but analyzing historical data by a suitable model can predict the optimal time. A number of services, such as the online site Expedia, show historical price data for a specific flight date, displaying what the ticket prices were on each day leading up to the flight. This allows customers to make more informed decisions as to when they should purchase their tickets.

1.2 Why Reinforcement Learning?

Reinforcement learning is the branch of machine learning which is used to make a sequence of decisions. In Reinforcement Learning, agents learn to achieve a goal in an uncertain, possibly complex environment. The AI faces some very complex situations in reinforcement learning. The device employs tests and errors to find a way to solve the problem. To get the machine to do what the user wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward.

However, the programmer sets the reward policy, and that is, the rules of the system or something like a game. He does not give the model any tips or direction for how to solve the problem. It is the responsibility of the model to find out how to perform the task to maximize the reward. Beginning from typically some random trials which finish with sophisticated strategy and extraordinary proficiency. For those things reinforcement learning is currently the most effective way to generate machine creativity. Unlike humans, artificial intelligence can gather experience from thousands of parallel gameplay if a reinforcement learning algorithm is run on a powerful computer infrastructure.

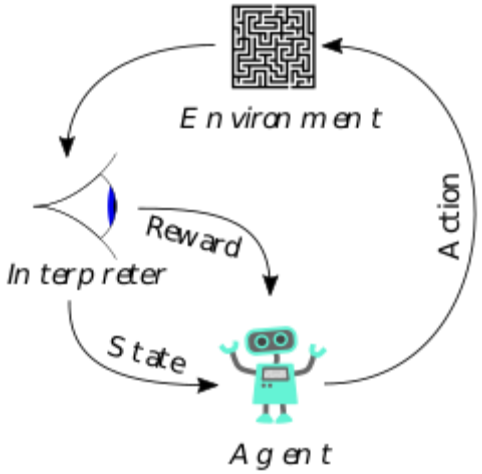


Fig 1.1: Typical Reinforcement Learning Scenario (from Wikipedia)

In Reinforcement learning the agent interacts with the environment, receives a reward and adapts its policy. But you do not need to tell which one is the right action in a given state, or which one is the correct answer. You have to provide partial information to the model, but not have to provide a correct answer or action. And no labeling is needed like supervised learning. So the traditional classification algorithms do not work properly so that there is a big chance to lose a big amount of money.

For this scenario, we decide to use reinforcement learning to optimize the decision to purchase the air ticket.

CHAPTER 2

2.1 A Comparative study on Mining Airfare Data

In Paper [1], authors have compared different methods based on their accuracy. They compared four methods which are Rule Learning, Time Series, Hand-Crafted Rule and Q-learning. They also generated a model named Hamlet by combining the results of Ripper, Q-Learning, and Time series. They use a crawler machine (data collection agent) to collect data from a website which is selling the air ticket online. Firstly they apply those data mining methods to 12,000 data which was collected in a 3 hours interval for 41 days.

They combine the results by a special technique named stacking generalizer for a better result. Below is the table for the performance of algorithms on multiple flights over a three hour interval.

Algorithm	% of Optimal	Net Savings
By Hand	55.5%	$163,523
Ripper	53.5%	$173,234
Time Series	-81.1%	-$262,749
Q-Learning	46.2%	$149,587
Hamlet	59.2%	$191,647

Table 2.1: Performance of algorithms

Finally they declare that Hamlet their custom made algorithm has better performance over Rule Learning, Time Series, Ripper and Q-Learning.

2.2 Comparative Study of RL Algorithms on Airfare Data

This paper [2], they compare 3 methods to predict airline price of a domestic route. Those methods are baseline, Q-Learning and Deep Q-Network. In the Baseline model they try to find out the earliest possible time point to buy a ticket for a good price which is to minimize risk aversion. In the Q-Learning model they modified the main equation of the Q-Learning algorithm to find the best reward. It helps to predict whether to buy a ticket or not to buy a ticket or wait some time to buy. They also implement the DQN to incorporate the nuanced features. They used the standard Q-Learning Formula to implement the neural network. They use ReLU as an activation function to allow negative q-values for a better output layer.

Finally they train those models to and find the following test results.

Algorithm	% of correct wait decision	% of savings
Baseline	0.00%	00.00%
Q-Learning	26.82%	-0.62%
DQN	31.00%	-1.60%

Table 2.2: Performance of Baseline, Q-Learning and DQN

They showed Q-Learning has a better performance over baseline. DQN is better then Q-Learning in terms of waiting decisions and Q-Learning is better than DQN for saving more money.

2.3 Another Competitive Study on Flight Ticket Pricing

In this paper [3], the authors tried to predict air ticket prices though different machine learning methods. They choose 4 classification algorithms namely Logistic Regression, AdaBoost-

DecisionTree, KNN, and Uniform Blending as well as Q-Learning which is a Reinforcement Learning Algorithm to predict buy or wait for a particular ticket. They declared that AdaBoost-DecisionTree is the best algorithm for this task and Q-Learning is also a better option for predicting air ticket price.

2.4 Regression Models for Predict Optimal Air Ticket Price

This paper [4] is also a competitive paper. The authors train 2 domestic route flights, constructing Partial Least Squares (PLS) regression model. First they adjust the model complexity by selecting the number of PLS factors to generate when training. They compute the Lagged feature using only the most recent values. They serially varied the number of factors in the optimal lag scheme search. They showed the structure of a trained model can be tested for knowledge about the domain. For these reasons, the performance of the PLS algorithm increased.

Finally they compared their model with a website named Bing Travel and showed that Bing Travel predictions do not save significant money over the immediate purchase for the airline specific target flight. Finally they declared their proposed model has a better performance than the website Bing Travel.

2.5 Classification Algorithms for Predicting Air Ticket

In paper [5], they implemented some classification algorithms to train a huge number of data. First they clean up the data and prepare the training data. They train the dataset by Support Vector Machine (SVM), Linear Regression, Decision Tree, Multilayer Perceptron, Gradient Boosting and Random Forest Algorithm to find the following test results. They use python library scikit to learn models. R-square, MAE and MSE are used to find the accuracy of these models.

Algorithm	R-Square	MAE	MSE
Decision tree	0.67	0.13	0.21
Random forest	0.68	0.13	0.21
K-NN	0.65	0.13	0.22
Linear Regression	0.40	0.19	0.29

Table 2.3: Accuracy of the ML Models

They showed that Random Forest has a better accuracy than any other model they implemented.

2.6 ML Models for Predicting Air Ticket Price

This paper [6] reported on a preliminary study with 9 Machine Learning models to predict air ticket price. They work for a dataset with 8 different features. They trained the data several times excluding some features to find the best prediction. The training with all features gave the best accuracy over all models. But execution time of training with all features was very high. And the Bagging Regression Tree gave the best accuracy overall, but time consuming. Below is the comparison table with all features.

ML Models	Accuracy (%)	Execution Time (sec)
Multilayer Perceptron	80.28	20.88
GR Neural Network	66.83	0.13
Extreme Learning Machine	68.68	0.05

Random Forest Regression Tree	85.91	5.50
Regression Tree	84.13	0.04
Bagging Regression Tree	87.42	17.05
Regression SVM (Polynomial)	77.00	1.23
Regression SVM (Linear)	49.40	0.34
Linear Regression	57.25	0.10

Table 2.4: Comparison table with all features

2.7 Predicting Airfare Prices

In this paper [7], authors focus on features that mostly affect price fluctuation and ignore the features that impact very little. He used the Weka Machine Learning Suite to experiment with different Machine Learning algorithms. He tested effectiveness using 70% Hold-out Cross Validation. The performance of Ripple Down Rule Learner, Logistic Regression and Linear SVM are as follows.

Models	Accuracy
Ripple Down Rule Learner	74.5%
Logistic Regression	69.9%
Linear SVM	69.4%

Table 2.5: Performance of different algorithm

Ripple Down Rule Learner has a better accuracy than Logistic Regression and Linear SVM.

2.8 Reinforcement Learning on Stock Market Prediction

In this paper [8], they tried to implement a reinforcement learning model to predict the future stock market. They used Deep Q-Network with a Convolutional Neural Network function approximator. They used stock chart images as input to predict the global stock market. The model they proposed predicts profit in the stock market of a country where it also predicts profit in international stock markets. They trained their model in the US market and tested it in some other countries for a long period of time. They used Q-Learning to find some patterns on stock chart images and predict future stock price scenarios for global stock markets. They showed that the future price of the stock can be predicted for all kinds of stock markets in the world. They tried to show that their model has a better performance over other models for future stock market prediction and Q-Learning is a better algorithm for solving this kind problem.

2.9 A study on Computational Complexity of Air Travel Planning

In this paper [9], The author tried to give some basic information to understand why air travel planning is an interesting and difficult problem for a computer science background student. He tried to give some detailed information about the flights, behavior of flight tickets, complexity of air travel planning, effect of seat availability on ticket pricing, effect of global events on pricing and some other important price changing factors. He explained why the availability of tickets is the most important price changing factor. Whatever, he actually tried to understand why it is very difficult to predict air ticket prices.

2.10 A study on Predicting Flight Prices in India

This paper [10], reports on a huge study on flight ticket prices in India. They trained a huge dataset from 18 routes across India with some different machine learning models. They studied the behavior of the flight ticket price of almost every domestic route within the country India. They

implement the validations or contradictions towards myths regarding the airline industry. It is a comparison study among various models in predicting the optimal time to buy the flight ticket and the amount that can be saved of a passenger. Finally they proposed a customized model which included a combined result of the all models and also statistical models that have been implemented. They showed that their custom model has a best accuracy of above 90% for a few routes.

2.11 Summary of the Literature

To gather a good background knowledge to do this project we studied a lot of tasks similar to this project. We also studied the behavior of the flight ticket price from some trusted website. From all of the above study, we can say air ticket price prediction is a very complex and time consuming task. There are so many tasks already completed by using supervised learning and unsupervised learning models but there are very few tasks where reinforcement learning models are used.

CHAPTER 3

Reinforcement Learning and Q-Learning

Reinforcement learning is one of the three major branches of machine learning. Reinforcement learning is learning what to do, how to map situations to actions and how to maximize the reward signal. The learner is not told which actions to take, but instead it must find a set of actions and determine which actions have the most reward. In the most interesting and challenging cases, actions may act not only the immediate reward but also the next situation and, through that, all subsequent rewards. Two most important features of reinforcement learning is trial and error search and delayed reward. Those two characteristics are distinguishing Reinforcement Learning from Supervised Learning and Unsupervised Learning. The distinction between problems and solution methods is very important in reinforcement learning in some particular terms. Failing to make this distinction may create many confusions. It can formalize the problem of reinforcement learning using ideas from dynamical programming theory and it is particularly, as the optimal control of incompletely known Markov decision processes. The basic idea of formalization is simply to capture the most important aspects of the real problem facing a learning agent interacting over time with its environment to achieve a goal. A learning agent must be able to sense the state of its environment to some extent and must be able to take actions that act the state. The agent also must have a goal or a set of goals relating to the state of the environment. Markov decision processes are intended to include just these three aspects, sensation, action, and goal. Any method that is well suited to solving such problems can be considered to be a reinforcement learning method. Some well known reinforcement learning methods are Monte Carlo, Q-Learning, Deep Q-Network, SARSA, Q-Learning Lambda and SARSA Lambda.

3.1 Supervised Learning Vs Reinforcement Learning

Reinforcement learning is different from supervised learning, the kind of learning studied in most current research in the field of machine learning. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. Each example is a description of a situation together with a specification and the label of the correct action. The system should take to that situation which is often to identify a category to which the situation belongs. The object of this kind of learning is for the system to generalize its responses so that it acts correctly in situations which is not present in the training set. This is an important kind of learning, but alone it is not sufficient for learning from interaction. In interactive problems it is often ineffective to obtain examples of expecting behavior that are both correct and representative of all the situations in which the agent has to act. In the modern world we expect learning to be most beneficial and an agent must be able to learn from its own experience.

3.2 Unsupervised Learning Vs Reinforcement Learning

Reinforcement learning is also different from Unsupervised Learning which is another major branch of machine learning. Unsupervised learning is typically about finding structure hidden in collections of unlabeled data. It is not true that the terms supervised learning and unsupervised learning only classify machine learning paradigms. From some point of view you can think of reinforcement learning as a kind of unsupervised learning because it does not rely on examples of correct behavior, reinforcement learning is trying to maximize a reward signal but does not find hidden structure. The hidden structure can certainly be useful in reinforcement learning, but it does not help the reinforcement learning agent to maximize a reward signal.

3.3 Special feature of reinforcement learning

Delayed reward: Which of the actions in its sequence are to be credited with producing the eventual rewards are called delay reward.

Exploration: In reinforcement learning, suppose agent has a state and this state may have a set of actions. If the agent choose the action that maximize the reward then it can be called exploitation and if the agent choose a random action that not maximize the reward then it can be called exploration.

Partially observable states: for understanding partially observable state suppose a robot with a forward-pointing camera cannot see what is behind it. In such cases, it may be necessary for the agent to consider its previous observations together with its current sensor data when choosing actions.

Life long learning: suppose a mobile robot may need to learn how to dock on its battery charger, how to navigate through narrow corridors, and how to pick up output from laser printers. This setting raises the possibility of using previously obtained experience or knowledge to reduce sample complexity when learning new tasks

3.4 Markov Decision Process

The Markov decision process is a model that attempts to predict an optimal result. It is a discrete-time stochastic control process that gives a mathematical structure for modeling decision making where result in a way random and under the control of a decision-maker. Markov Decision Process used to solve optimization problems using dynamic programming. For a better understanding of MDP, we have to study Markov Chain, Markov Property, Markov Process, and Markov Reward Process.

3.4.1 Markov Property

Evaluation of the Markov Process in the future depends only on the present state and does not depends on past state property is called Markov Property. That's mean the future is independent of the past given the present. So that, the Markov process is a memoryless random process.

A state S_t is Markov Property if and only if,

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

- The state pick up all relevant information from the history.
- Once the state is known, the history may be thrown away.
- The state is a sufficient statistic of the future

3.4.2 Markov Process

Markov Process is a memoryless random process which is a sequence of random states with the Markov property. It is a stochastic extension of finite-state automation. The system is only in the current state at each time step and state transition is not dependent on the previous history. State transitions are probabilistic in Markov Process.A Markov Process is a tuple (S, P)

- S is a (finite) set of states.
- P is a state transition probability matrix,

$$\boldsymbol{P_{ss'}} = \mathbb{P}[\boldsymbol{S_{t+1}} = \boldsymbol{s'} \mid \boldsymbol{S_t} = \boldsymbol{s}]$$

3.4.3 Markov Chain

Markov chain is a stochastic model describing a sequence of state actions where the present state fully observes all the information to evaluate the future actions. That is, the probability of each action depends only on the previous event. Markov chain is a type of Markov process. The Markov decision process is an extension of the Markov Chain.

3.4.4 Markov Reward Process

The Markov reward process is a stochastic process which is an extension of the Markov chain. That is adding an additional variable record a reward to each state. The system receives an expected reward at a given time.

A Markov Reward Process is a tuple (S, P, R, γ)

- S is a finite set of states
- P is a state transition probability matrix,

$$\boldsymbol{P_{ss'}} = \mathbb{P}\left[\boldsymbol{S_{t+1}} = \boldsymbol{s'} \mid \boldsymbol{S_t} = \boldsymbol{s}\right]$$

- R is a reward function,

$$\boldsymbol{R_s} = \boldsymbol{E}[\, \boldsymbol{R_{t+1}} \mid \boldsymbol{S_t} = \boldsymbol{s}]$$

- γ is a discount factor, γ ∈ [0, 1]

3.4.5 Formal Definition of MDP

Markov decision process (MDP) is Markov reward process with decisions. It is an environment in which all states have Markov property. Markov Decision Process is a tuple (S, A,P, R, γ) where,

- S is a finite set of states
- A is a finite set of actions
- P is a state transition probability matrix,

$$\boldsymbol{P^a_{ss'}} = \mathbb{P}[\boldsymbol{S_{t+1}} = \boldsymbol{s'} \mid \boldsymbol{S_t} = \boldsymbol{s}, \boldsymbol{A_t} = \text{a}]$$

- R is a reward function,

$$R^a_s = \boldsymbol{E}[\, \boldsymbol{R_{t+1}} \mid \boldsymbol{S_t} = \boldsymbol{s}, \boldsymbol{A_t} = \text{a}]$$

- γ is a discount factor γ ∈ [0, 1].

3.5 Elements of Reinforcement Learning

There are four main sub elements of a reinforcement learning system beyond the agent and the

environment. They are a policy, a reward signal, a value function, and, optionally, a model of the environment.

3.5.1 Policy

A policy defines the learning agent's way of behaving at a given time. More specifically, policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic, specifying probabilities for each action.

$$\pi: \mathrm{A} * \mathrm{S}[0,1] \rightarrow [0,1]$$

$$\pi(\mathrm{a}, \mathrm{s}) = \Pr(\mathrm{at} = \mathrm{a} \mid \mathrm{st} = \mathrm{s})$$

This the policy map equation given above, where it gives the probability of taking action ***A*** when in state ***S***.

3.5.2 Reward

A reward signal defines the goal of a reinforcement learning problem. On each time step, the environment sends to the reinforcement learning agent a single number called the reward. The agent's only objective is to maximize the total reward it receives over the long run. The reward signal thus defines what are the good and bad events for the agent. In a biological system, the rewards are analogous to the experiences of pleasure or pain. They are the immediate and defining features of the problem faced by the agent. The reward signal is the primary basis for altering the policy; if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future.

$$R = \sum_{t=0}^{\infty} \gamma^t \, r_t$$

This is the reward equation, where r_t is the reward at step t,$\gamma \in [0,1]$ is the discount rate.

In general, reward signals may be stochastic functions of the state of the environment and the actions taken. Whereas the reward signal indicates what is good in an immediate sense, a value function specifies what is good in the long run.. However, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Where rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long term desirability of states after taking into account the states that are likely to follow and the rewards available in those states. For example, a state might always produce a low immediate reward but still have a high value because it is regularly followed by other states that produce high rewards or vise-versa. To make a human analogy, rewards are somewhat like pleasure (if high) and pain (if low), whereas values correspond to a more refined and farsighted judgment of how pleased or displeased we are that our environment is in a particular state. Rewards are in a sense primary when values and predictions of rewards are secondary. Without rewards there could be no values, and the only purpose of estimating values is to achieve more reward.

3.5.3 Value

The reward function gives some values that are the most important for making and evaluating decisions. Action choices are made based on value judgments. We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run. Unfortunately, it is much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime. In fact, the most important component of almost all reinforcement learning algorithms we consider is a method for efficiently estimating values. The central role of value estimation is fairly the most important thing that has been learned about reinforcement learning over the last six decades.

$$V_\pi(s) = E[R] = E[\sum_{t=0}^{\infty} \gamma^t r_t A = \pi r^2 \mid_{s_0 = s}]$$

This is the value function where the random variable R denotes the return, and is defined as the sum of future discounted rewards (gamma is less than 1, as a particular state becomes older, its effect on the later states becomes less and less. Thus, we discount its effect).

3.5.4 Environment

The fourth and final element of some reinforcement learning systems is a model of the environment. This is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave. For example, given a state and action, the model might predict the resultant next state and next reward. Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. Methods for solving reinforcement learning problems that use models and planning are called model-based methods. As opposed to simpler model-free methods that are explicitly trial-and-error learners which are viewed as almost the opposite of planning. Modern reinforcement learning spans the spectrum from low-level, trial-and-error learning to high-level, deliberative planning.

3.6 Q-learning

One of the most important in reinforcement learning algorithms is known as Q-learning. It can also be viewed as a method of dynamic programming (DP). In Q-learning, future states depend only on the current state. The agent experiences a sequences of actions, without building any map. The agent choose an action at an individual state, then calculates its consequences to getting the immediate reward or penalty. The agent receives the reward value and count it with the value of the state to which it is taken. The agent continually trying all actions in all states to learn which actions are best overall. And making those best actions are possible by judging long-term best reward. This is the main Q-Learning equation below.

$$Q(S_t, A_t) \leftarrow \mathrm{Q}(S'_t, A'_t) + \alpha * \mathrm{R} + \gamma \max Q(S_{t+1}, A_t) - Q(S'_t, A'_t) \ldots \quad (3.1)$$

In the Q-Learning equation α is learning late. Learning rate is challenging as a value which is too small may result in a long training process that could get hold up, where a value which is too large may result in learning a sub-optimal set of weights too fast or an unstable training process. **R** is a reward which is after completing an action at a given state then received a value. Every action gives a reward. This is the most important thing to set up a good reward, because performance of the Q-learning method mostly depended on this reward.

In Q-Learning equation γ is a discount factor it's used to balance immediate and future reward. **max Q (S_{t+1}, A_t)** mean that find the future best reward in possible actions in its current state.

In Q-learning algorithm, an agent moves around some discrete, finite state on the environment and chooses one from a finite collection of actions at every step. Every action has a specific reward. Q-learning has a specific learning late and a discount factor. First of all an agent chooses a random state. Every state moves some specific action. Before the action agent checks the previous maximum reword state. Then select this action of the state and immediately update the q table value according to the equation. Where γ(discount factor) is a number between 0 and 1 ($0 \leq \gamma \leq 1$) and also α(learning rate) is a number between 0 and 1 ($0 \leq \alpha \leq 1$). In a certain number of time, the agent can learn the sophisticated environment.

For understanding how Q-learning work algorithm works, we have an example of maze solving robot. Where the agent (robot) interact with a deterministic environment. In this environment every possible space where the robot can go will be the states. The agent (robot) have some finite number of action which are move right, move left, move up, move down.

3.6.1 Q-table initialization

In the Q-Learning equation (3.1), **Q** is a table where all of the values are initially arbitrarily fixed and which is chosen by the programmer. **S** is a state of the Q-table and **A** is an action of this state. Q-table is a table or matrix that follows the shape of [state, action]. For maze solving robot there is a Q-table where all of values initialized by 0. So that initially the agent (robot) doesn't have any knowledge about the environment. Then Q-table may update and store q-values after an episode. This q-table becomes a reference table for agent (robot) to select the best action based on the q-

value.

In the next step the agent have to interact with the environment and make updates to the state action in the q-table Q [state, action]. In Q-Learning, the agent can interacts with the environment in 2 ways. One is exploitation and another is exploration. The first way we can use the q-table as a reference table and tries all possible actions for a given state. The agent choose the action on the max value of those actions. In second way, the agent take action randomly. This is called exploring. In Table 3.2 the values of Table 3.1 are updating based on max value of those actions.

Q-Table		Action			
		Right	Left	Up	Down
State	0	0	0	0	0
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	328	0	0	0	0
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	499	0	0	0	0

Table 3.1: Q-Table Initialization

3.6.2 Updating Q-table

The updates on the q-table occur when the agent chose an action and ends when the agent reaching some terminal state. A terminal state is a state which can be anything like reaching the goal on the

maze with completing some desired objective. After a single episode, the agent does not learn much. But a good number of exploring can make the agent to learn the optimal q-values that is Q-table. We set some values for our maze. For every valid action we set the state reward as -1, for invalid move we set reward as -100. And the reward of reaching the final state is +100.according to the reward Table 3.1 q-table continuously updating using q-learning equation until a certain period of time. After completing those episode the q-table get maximum values as you see in the Table 3.2.

After training the Q-learning algorithm Q-table looks like the table below

Q-Table		Action			
		Right	**Left**	**Up**	**Down**
State	**0**	**-520.0**	**-101.24**	**-121.5**	**-121.05**
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	328	**-271.58**	**-570.96**	**-222.30**	**-109.85**
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	499	**-690.22**	**-333.02**	**-150.58**	**-120.78**

Table 3.2: Training of Q-Table

3.7 Deep Learning

Deep Learning is an advanced machine learning method broadly used based on Artificial Neural Networks. Typical Deep learning can be supervised, unsupervised or semi-supervised and also used in Reinforcement Learning which is another major branch of machine learning as Deep Reinforcement Learning. Deep Neural Networks, Deep Belief Networks, Recurrent Neural Networks, Convolutional Neural Networks (CNN) are some common Deep Learning methods broadly uses in recent years to increase the machine's performance.

Deep learning algorithms use multiple layers on the network to extract raw inputs that have incredible high-level features. Each of those layers learns to transform raw input data (that may have complex representation) to abstract and compound representation. The Architectures of Deep Network can be constructed with a layer-by-layer greedy method which can help to distinguish and pick some features that improve the performance of the system.

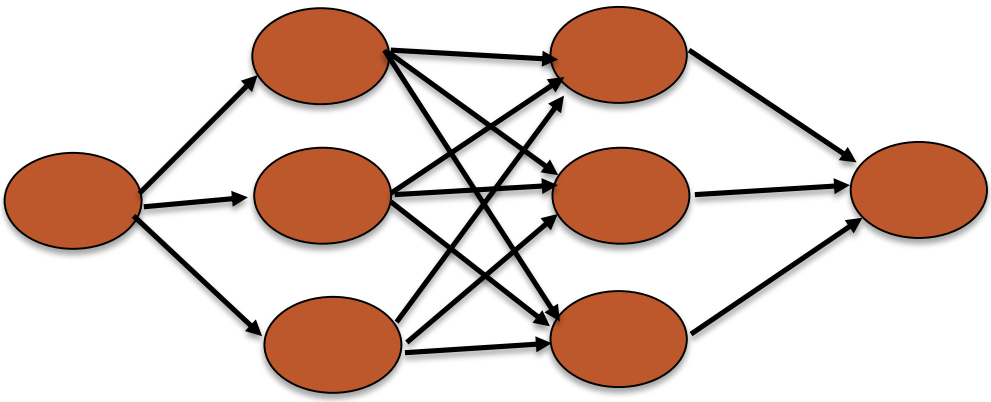


Fig 3.1: An artificial neural network diagram

3.8 Deep Reinforcement Learning

Deep Reinforcement Learning (Deep RL) is a subfield of Reinforcement Learning. It is an advanced Machine Learning field that combines Reinforcement Learning (RL) and Deep

Learning. In this field, RL creates an agent to make a decision and Deep Learning helps the agent to make an optimal decision from basically a huge amount of unstructured input data.

3.9 Deep Q-Network

Deep Q-Network (DQN) is an advanced version of Q-learning algorithm that a deep artificial neural network [basically deep convolutional neural network (CNN)] used to approximate the Q-value function. In DQN, the state is given as the input and taking values of all possible actions to generate the Q-value as output.

DQN mostly used in big state spaces because in that case, Q-learning may be ineffective. Imagine a state space that has 10000 states. And for every action there 10000 actions. There needed a table that has 10000*10000 cells. This is a very big state space so that Q-learning may yield poor output. For obscure states, Q-learning can't educe the q-value of future states from previous states so it couldn't be used for that kind of state space.

The difference between Q-Learning and DQN is illustrated below:

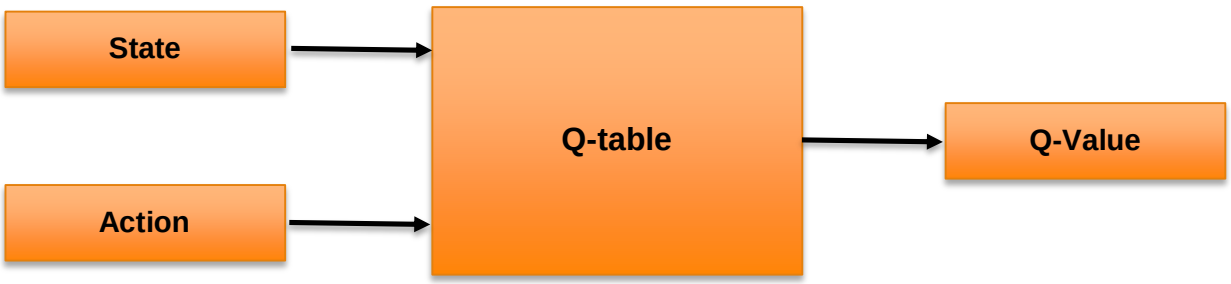


Fig 3.2: Q-learning

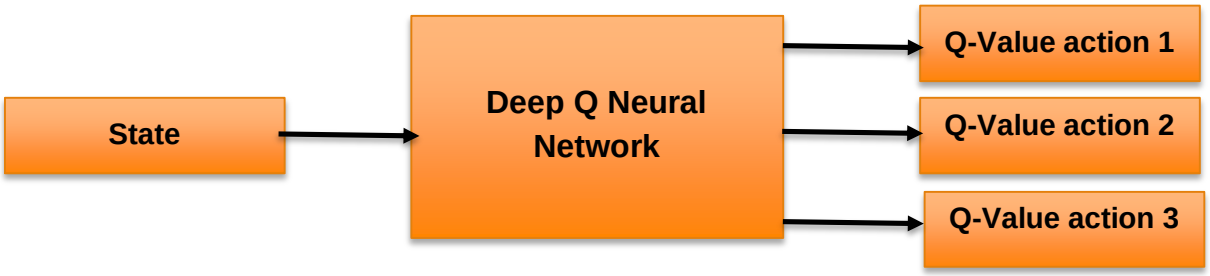


Fig 3.3: Deep Q-Network

In deep Q learning, there is a neural network for approximate the q-value function. The state will be an input and the neural network generate the q-values for all possible action of the states. The target is to produce the next action. This is not bounded to a Fully Connected Neural Network. The Convolutional Neural Network or whatever type of network which suits the model can be used for implementing DQN.

3.9.1 Reply Memory

To implement a DQN, first we have to understand what replay memory is and why it is used in Deep Q-Network.

In deep Q-network, a technique called experience replay often used during training the model. Using this technique we can store the experiences gain by the agent at each time step in a memory called replay memory. Suppose the agent's experience at time t as e_t. It is defined is as a tuple:

$$e_t = (s_t, a_t, r_{t+1}, s_{t+1})$$

Where,

- s_t is the state of the environment
- a_t is the action taken from state s_t
- $r_{t+1}$is the reward taken at time t+1 as the result of the previous state-action transition
- s_{t+1} is the next state

Each time the agent taking action, the experiences over all episodes are stored in the replay memory. The replay memory is bounded by some finite size limit so that it can only store some final experiences.

3.9.2 Network Layers

The agent is going to learn in a way that is at least similar to how a biological (human or animals) brain able to. The network layers act in a way that the biological neuron does. The layers take some raw input and transform this input data by calculating weights. Then calculate an intermediate state by applying a non-linear function to this transformation. By repeating those steps the neural network learns multiple layers of non-linear data. Then combine those data in the final layer for output.

3.9.3 Target Network

In Deep Q-Network a second network is used during the training procedure. This network is used to generate the target values of the Q-table that will be used to compute the loss for each action of the agent during the training period. In DQN, the target network is a separate network used to avoid some unexpected error issues. That is when the Q-table values are shifting and if we use a constantly shifting set of values to adjust those network values, then the value estimation process can go out of control. For this issue, the target network is separated and the weight of the target network is fixed. This network makes the procedure a bit slow but using this target network the training can proceed in a more stable process.

3.9.4 Updating the Q-table using Bellman Equation

The bellman Equation is the key to update the q-table after each step. The agent updates the current value with the calculated optimal future reward. This assumes that the agent takes the best action for the current state. The agent will search for all possible actions for a particular state and then

choose the state-action pair with the highest Q-value. This equation and the updating procedure are already described broadly in 3.6. Following this procedure, the agent updates the q-table in each step.

CHAPTER 4

Methodology & Result Analysis

4.1 Dataset

We use two datasets to implement our model. The first dataset we used from (https://usc-isi-i2.github.io/knoblock/publications.html) this link And the second one is from Kaggle (https://www.kaggle.com/nikhilmittal/flight-fare-prediction-mh?).

Our first dataset contains over 12,000 fare observations for a 41 day period for six different airlines for the route Los Angeles (LAX) to Boston (BOS) and Seattle (SEA) to Washington, DC (IAD). For each departure date, the data was collected 21 days in advance at three-hour intervals.

Our second dataset contains over 10,000 observations for 12 different airlines for 6 different domestic routes in India. Those routes are as follows.

- Bangalore - New Delhi
- Bangalore - Delhi
- Kolkata - Bangalore
- Delhi - Cochin
- Chennai - Kolkata
- Mumbai - Hyderabad

4.2 Setting up the Environment and Reward

The environment plays the major rule in reinforcement learning approaches. In section 3.3.4 we had a brief overview of the environment. The environment is divided by some particular states. The properties of the states are as follows.

- Airline
- Departure Time
- Source
- Destination
- Date of Journey

Setting up reward is one of the important thing of the method Q-Learning. In section 3.3.2, we had a brief overview of reward. For every state of the environment there must be a reward. We initially set the reward values as the price of the particular states. The reward set against the state as follows.

State	Reward
Airline Departure Time Source Destination Date of Journey	Price

Table 4.1: State and Reward setting

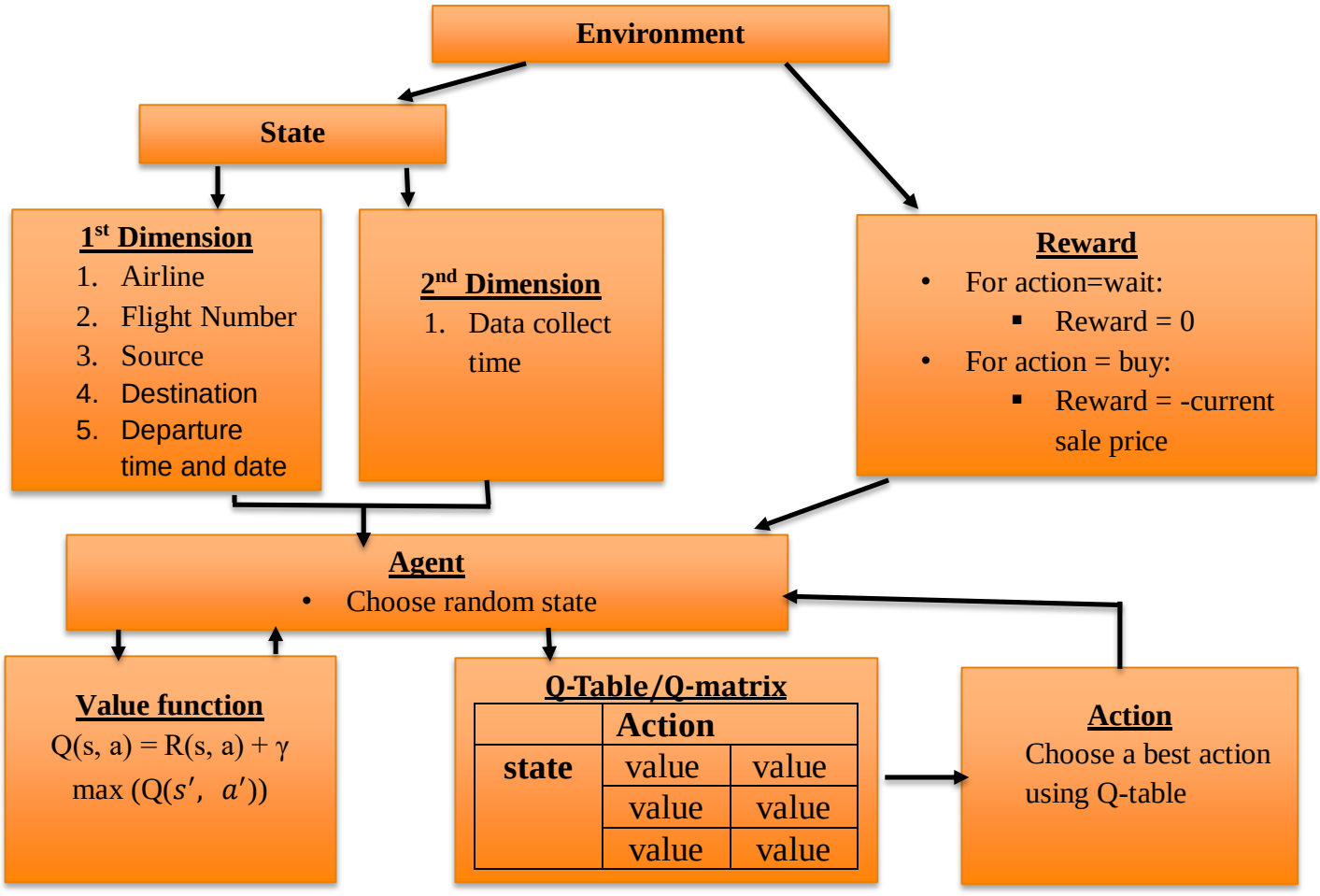
4.3 Dimension Setting for Q-Table

We set three dimensions for our Q-Table. In the first dimension we keep state properties. State properties are Airline, Departure Time, Source, and Destination. In the second dimension we keep the date and time of the collected data. The second dimension is for keeping the track to check the random time. And the third dimension is for action. The action will be buy or wait.

4.4 Q-Learning

First we initialize all values of Q-Table by zero. We choose a random row (state) and a random column (date and time) from our q-table. Then we check terminal state or available state. If the action of the state is 'buy', it gets the reward as the negative value of the price. And if the action is 'wait', the state reward remains unchanged. In section 3.5 we had a brief overview of Q-Table. We set our episode as 100000, epsilon is 0.9 that means we have 0.9 times exploitation and 0.1 times exploration. And discount factor is 1.0. This is for reduce the noise values. Then for every episode we choose a random state (row of the q-table) and random time (column of the q-table). The agent choose an action from the state get some reward and go to the next state. This process will be continue until the agent get the action 'Buy'. For every action it's updating the q-table by the q-equation. After completing all of the episode the q-table get maximum values.

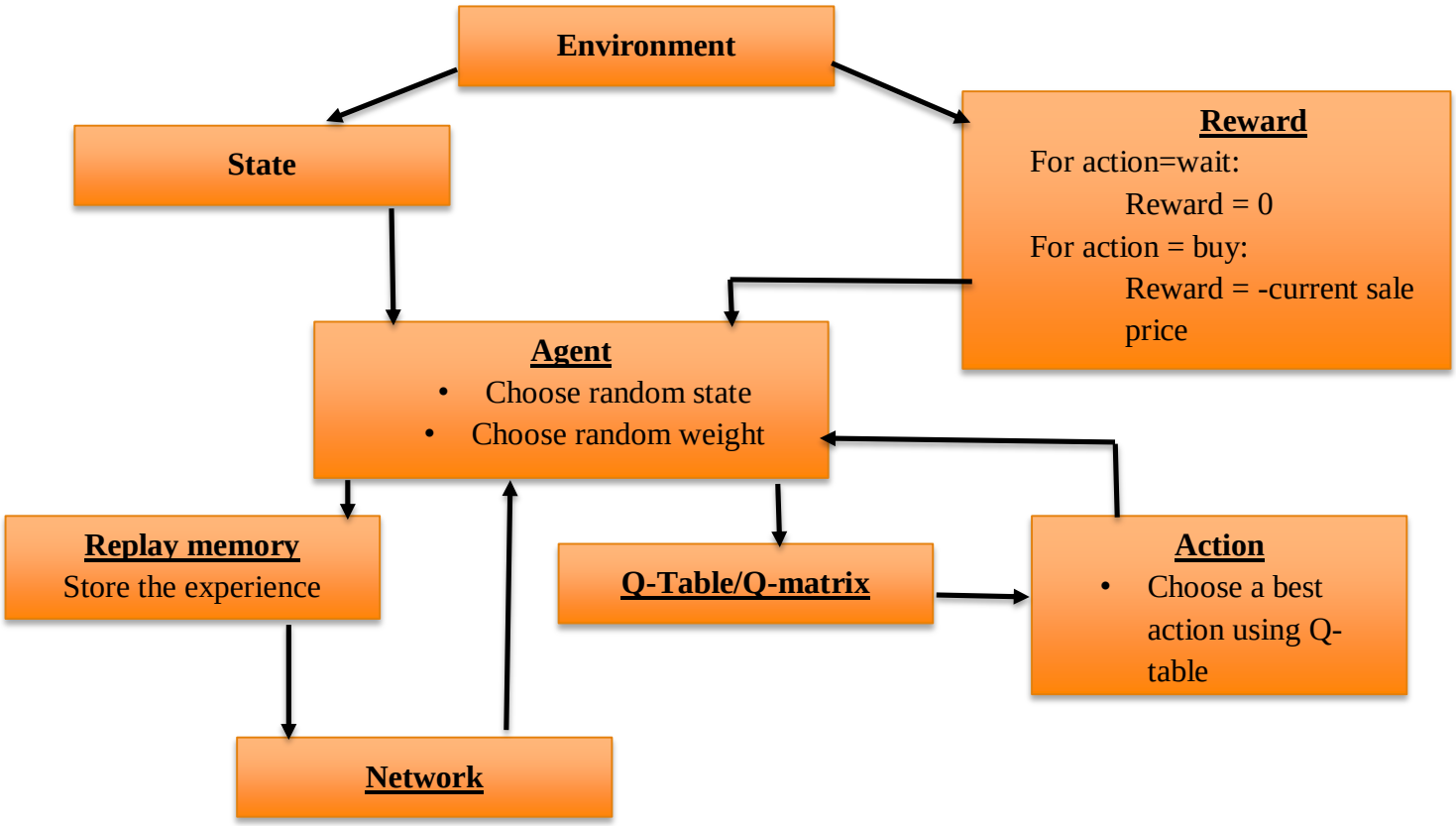
Architecture of Q-Learning



4.5 Deep Q-Network

In the DQN algorithm, first we divide our state into two dimensions like our previous Q learning model. We put flight number, departure date, departure time, airline, source, the destination in the first dimension, and in the second dimension, we put the date and time of purchase of the ticket. Following the size of our dimension, we create a Q-table that has three dimensions. The first two dimensions of the Q table are for state property and the third dimension for action (buy or wait). Then we set negative values of the flight price as the reward. Next, we utilize a replay memory to store the value of the state action function. We use the replay memory to break down the connection between two consecutive samples. It is also used to avoid repeated data. Then Initialize the random weights. Then Initialize the starting state. For each time step. Select an action via exploration or exploitation. Perform selected action. Obey reward and next state. Store experience in replay memory. Calculate loss between output Q-values and target Q-values. Need a pass to the target network for the next state. Updates weights to minimize loss.

Architecture of DQN



4.6 Q-Learning Result

As discussed in the previous chapter, we have implemented a Q-learning model that can predict the optimal time to buy the ticket or wait for a better price in the future. The model we implemented can check when the q-table values are maximum then show this time for buying a ticket. And when the q-table values are not maximum then show this time for the wait. The decision depends on the maximum value of the q-table. In the following table we found our result from the model:

Data	Training Data	Test Data
Number of data	12224	1000
Number of buys	97	3
Number of waits	12127	997
Number of buys (%)	0.79%	0.03%
Average price buy of the tickets	1009.099	1694.212

Table 4.2: Q-Learning result

4.7 DQN Result

Following our previous Q-learning model, we implemented another model that is an advanced version of our previous model. The prediction of this model we analyze is much better than our previous model. It has improved the values of the q-table that helps the agent make better predictions. This model is a bit time-consuming but the result that comes after the training of DQN is satisfying over our previous model. The result is as follow:

Data	Training Data	Test Data
Number of data	12224	1000
Number of buys	312	47
Number of waits	11912	953
Number of buys (%)	2.55%	4.75%
Average price of the tickets (buy)	916.814	1153.0

Table 4.3: DQN result

4.8 Comparing the result of Q-learning & DQN

As we see in table 4.2 and table 4.3, the DQN model produced a much better result than our previous model Q-Learning. The interesting thing is the number of buys rises in the DQN model. And the avg. price of the ticket (buy) decries significantly. As per our calculation, the DQN model saves 107 Rs. (10.39 %) per ticket than the Q-Learning model.

Model	Number of buys (%)	Avg. price of buys
Q-Learning	0.75%	1029
DQN	2.75%	922

Table 4.4: Comparing the result of Q-learning & DQN

Future Work

The next time we will try to improve our algorithms. We will try to implement some other reinforcement learning algorithms. We think those algorithms will yield better output for airline price prediction.

Chapter 5

Conclusion

This is satisfying for us that DQN improved our result. DQN makes the prediction much better than our previous model Q-learning. This is not easy to implement Reinforcement Learning algorithms to this type of dataset but we tried to do something better. Reinforcement learning is exciting for both of us and we enjoyed the time we work on this project.

References

1. Oren Etzioni, Craig A. Knoblock, Rattapoom Tuchinda and Alexander Yates “To Buy or Not to Buy: Mining Airfare Data to Minimize Ticket Purchase Price” Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, August 2003.
2. Niki Agrawal and Ramnik Arora “Reinforcement Learning for Flight Ticket Pricing” Stanford ML project CS-229, 2019.
3. Jun Lu “Machine learning modeling for time series problem: Predicting flight ticket prices” international journal of innovative technology and exploring engineering, 2017.
4. William Groves and Maria Gini “A regression model for predicting optimal purchase timing for airline tickets” International journal of engineering research and technology, 2011.
5. Supriya Rajankar, Neha Sakharkar and Omprakash Rajankar “Predicting The Price Of A Flight Ticket With The Use Of Machine Learning Algorithms” International journal of innovative technology and exploring engineering , Volume : 08 , June 2019.
6. K. Tziridis, Th. Kalampokas, G.A. Papakostas and K.I. Diamantaras “Airfare Prices Prediction Using Machine Learning Techniques” IEEE-xplore-European Signal Processing Conference (EUSIPCO) 2017.
7. Manolis Papadakis “Predicting Airfare Prices” Stanford ML project CS-229, 2012.
8. Jinho Lee, Raehyun Kim, Yookyung Koh, and Jaewoo Kang “Global Stock Market Prediction Based on Stock Chart Images Using Deep Q-Network” IEEE Access, November 2019.
9. Carl de Marcken “Computational Complexity of Air Travel Planning” ITA Software, fall 2003.
10. Achyut Joshi, Himanshu Sikaria and Tarun Devireddy. “Predicting Flight Prices in India” ReserchGate, May 2017.