Gaming Meets AI- SoC Project Report

Submitted by- Aditya Jain

Task 1:

In the first week, I covered the following topics:

- **GitHub**:- We were introduced to the basics of GitHub and the advantages of using it in the first week. I went through a few more videos on youtube and explored a bit on my own in to get accustomed to the basic commands.
- **Python**:- I went through the resources provided, and also some other youtube playlists to get accustomed to coding with python. I had started learning Python as soon as I had decided to apply for SoC, and thus I already knew some basics. With the help of online videos/tutorials, I learned about the numpy library.
- **Introduction to ML/DL/RL**: We were provided provided resource links for learning the basics of ML, DL and RL. I went through some supplementary videos in order to cement the concept as I watched the videos. Also, I looked up lecture slides in order to do a quick revision of difficult topics.

Resource links:

- GitHub Basics Video
- Python Tutorial Playlist
- Python Tutorial Website
- CS229 Stanford Lecture Videos
- Additional resource for Bootstrap Aggregation, Random Forest and Boosting

Task 2:

The following topics were covered:

- **Probability and Linear Algebra**:- Revised the basic principles of probability and studied the properties of some common probability distributions such as Bernoulli, Multinoulli and Gaussian distributions. Also, brushed up linear algebra (mostly the part relating to eigen-decomposition, lagrange multipliers and Jacobian and Hessian matrices). Studied about the use of Hessian matrices in gradient based optimizations and lagrangian multipliers in solving constrained optimization problems.
- **Machine learning basics**:- Studied about the concept of over-fitting, under-fitting, hyperparamters and validation sets in detail. Also watched some youtube videos to understand the same. Re-encountered supervised and unsupervised learning algorithms (this time in a lot more detail). Went through youtube videos again, to understand stochastic gradient descent better.
- **Deep Feedforward neural networks**:- Revisited the concept of perceptrons. Studied about multi-layered perceptrons to approximate non-linear functions(like XOR) and then sigmoid neurons when values in a continuous interval are required as output. Learned about the basic components of a deep forward network: cost function, activation functions, output function and learning algorithm. Studied about the various types of each component that we can have, and the advantage of choosing one over the other and the short-comings of each of them. Learned of the main points that one should keep in mind while deciding the depth and width of a neural network, and related this with the concept of

representation power and Universal Approximation Theorem. Finally, went through the mathematical working of the back-propagation algorithm required for computing the gradients efficiently for the training process.

Resource links-

- Deep learning book by Ian Goodfellow
- Youtube resource playlist for machine learning basics
- Video series by 3Blue1Brown for neural networks
- NPTEL lectures for deep learning

Task 3:

- **Introduction to Reinforcement learning**:- Went through the first five chapters of the reinforcement learning book by Sutton and Barto.
- The first chapter gave a basic introduction mostly involving examples and explainations of the fundamental elements of RL such as policy, rewards, etc.
- Studied the n-armed bandit problem as an example of single-state problems. Focused on the theme of exploration and exploitation and learned about geedy action selection, epsilon greedy technique and gradient bandits, and the advantage or disadvantage of each of these in different situations. Studied about the contextual bandits as an extension of the k-armed bandits under an associative setting.
- In the third chapter we learned about markov property and the approximation of control processes as markov devision processes and also went through examples in order to understand the agent-environment interaction. We learned about value functions, concept of return maximization and optimal policy.
- In chapter four, the use of dynamic programming in order to eliminate unnecessary computation was highlighted. Bellman equation was brought to use again and we studied about policy iteration, value iteration, and the concept of policy evaluation and improvement with respect to generalized policy iteration.
- Chapter five introduced Monte Carlo methods as ways to determine value functions and optimal policies. Advantages of Monte Carlo Methods over Dynamic Programming in certain situations were highlighted. I studied about on-policy and off-policy approaches in order to do away with the assumption of exploring starts.

Resource links-

- Reinforcement learning book by Sutton and Barto
- Youtube play-list for revisions of chapters
- Youtube video for Monte Carlo Methods
- Youtube video for Multi-Armed Bandits
- Stanford Lectures on RL

Final coding:

- Before starting the main project, I went through official pytorch tutorials and coded up a single network DQN for relatively simpler games such as CartPole and MountainCar on OpenAI gym.
- After achieving good results with CartPole and MountainCar, I went through the research papers published by deepmind for Atari games. Despite the differences between between 2048 and Atari

games, the paper gave a general idea of DQN algorithm and it's implementation. After this, I also went through a thesis on using DQN for 2048 which mainly discussed the effects of different encoding and reward functions on the performance of the agent.

- Finally, I started the actual project. Since I had coded DQN previously, I had an idea of the general logic and structure of the code. After finishing the code completely, I modified the network architecture using some prior knowledge and some additional research. I introduced a target network as well in order to stabilise the learning. Also, I added gradient clipping in order to prevent overflowing of gradients. I experimented with the 'tau' value, i.e., the frequency of training and updating of Q-Model and Target Model

Resource links-

- Network Architecture Design
- Official PyTorch Tutorials
- DeepMind paper for Atari games
- Thesis on using DQN for 2048 Game