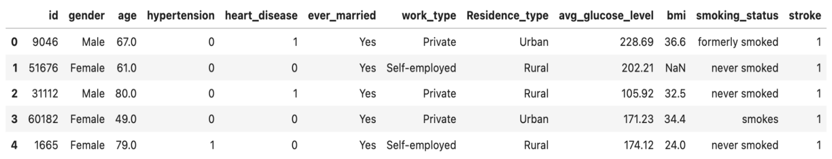
EN.530.641: Statistical Learning for Engineers Final Project

Author: Guangjing Zhu, Siyuan Sun, Yile Wang
Supervisor: Dr. Jin Seob Kim

Task 1: Machine Learning

Preprocessing

Source of Data: Stroke Prediction Dataset[1]
Dataset Description: The dataset includes several attributes that detail the patient's health status, demographic information, and lifestyle choices.



	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

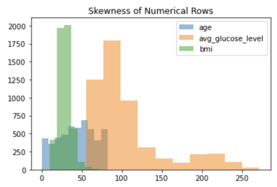


Figure 1: Overview of dataset

Figure 2: Skewness of Numerical Features

The missing values (NaN) in the body mass index (bmi) column were filled with the mean value for that column. 'gender', 'ever_married', and 'Residence_type' are binary categorical variables. Integer coding was used for these variables since there are only two categories. 'work_type' and 'smoking_status' are non-binary categorical variables without an ordinal relationship. One-hot encoding was utilized for these variables. Examination of the histogram plots below shows that the 'age', 'avg_glucose_level', and 'bmi' distributions have differing ranges and are skewed. Since Logistic Regression, Support Vector Machine, and Random Forest models were the models chosen to apply, the standard scaler transform was more appropriate. Specifically, Logistic Regression assumes the independent variables follow a Gaussian distribution. Support Vector Machines are very sensitive to the scale of the input features, and the performance can change significantly with unscaled data.

Results

The optimal Logistic Regression model uses a 'C' value of 10 for moderate regularization and up to 1000 iterations with the 'lbfgs' solver, suitable for small, multiclass datasets, ensuring accuracy and preventing overfitting. The optimized Support Vector Machine (SVM) has a high 'C' value of 100 for low error tolerance, a 'gamma' of 0.01 for a smoother decision boundary, and utilizes the 'rbf' kernel, ideal for non-linear data. Its design aims for high accuracy, though it risks overfitting. The selected Random Forest model employs 'log2' max_features and 400 trees, balancing complexity and generalization, and enhancing accuracy by diversifying decision trees. Comparatively, the SVM was the most effective, with the lowest test error of 0.151, followed by Logistic Regression at 0.159 and Random Forest at 0.168. The SVM's superior accuracy indicates better generalization from training to unseen data for this problem.

[1] https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

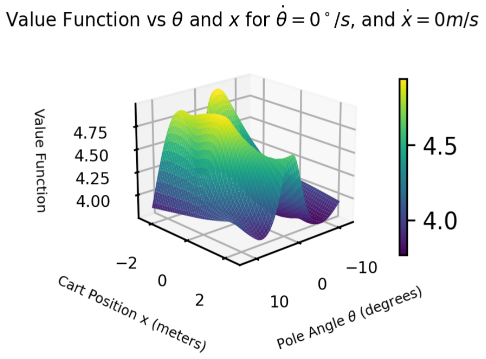
Task 2: Reinforcement Learning

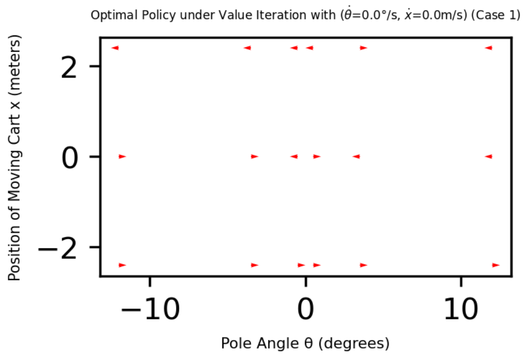
2.1: Dynamic Programming (DP)

Both policy and value iteration, the value function vs θ and x_dot under being plotted under 3 different selections of θ_dot and x_dot:

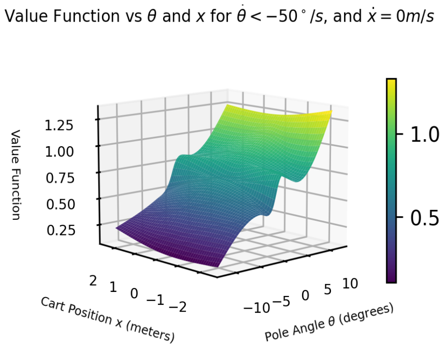
2.1.1: Plot of Value Function and Optimal Policy Under *<u>Policy Iteration</u>*:

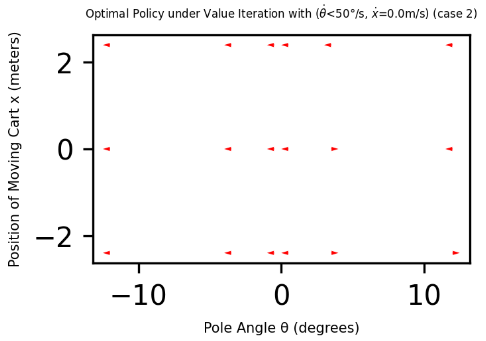
Case 1): Central bins for both θ_dot and x_dot;



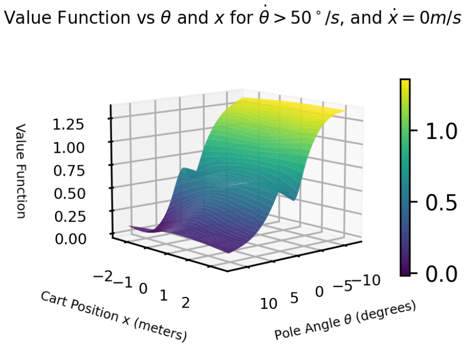


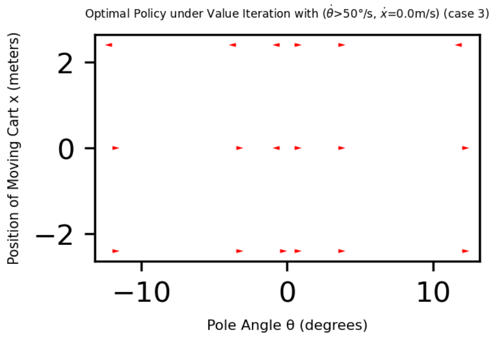
Case 2): Lowest bin for θ_dot and central bin for x_dot





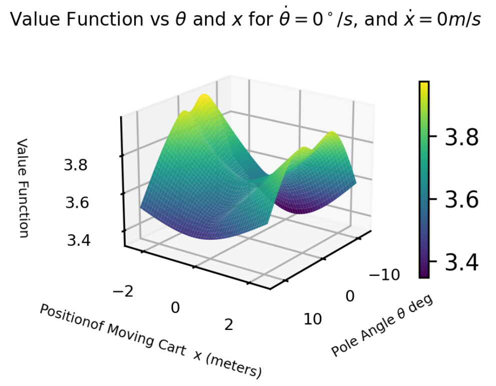
Case 3): Highest bin for θ_dot and central bin for x_dot.

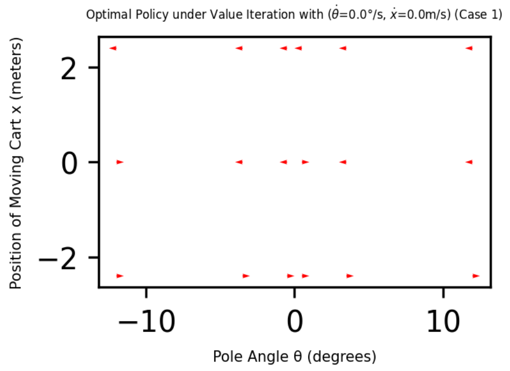




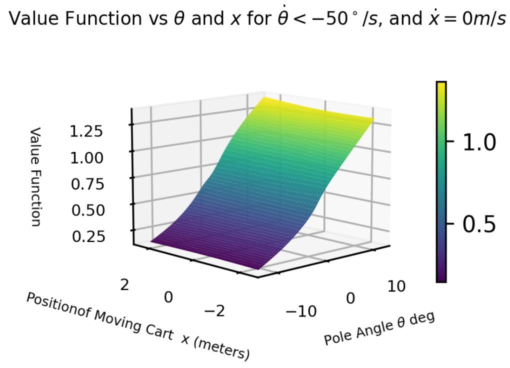
2.1.2: Plot of Value Function and Optimal Policy Under *Value Iteration*:

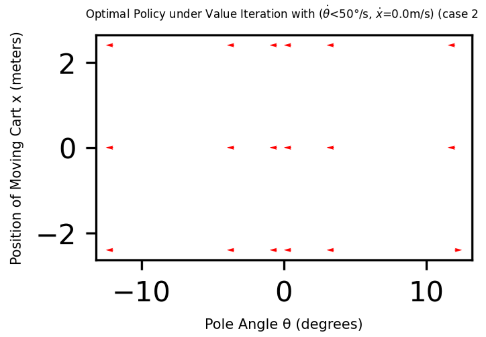
Case 1): Central bins for both θ_dot and x_dot;



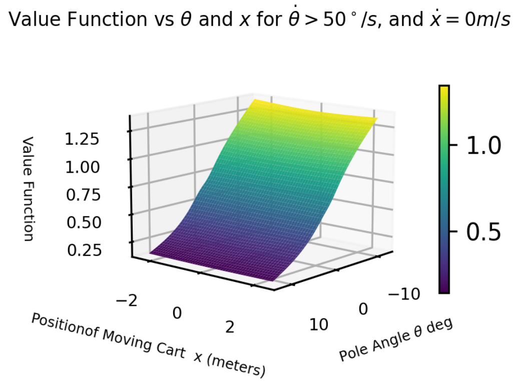


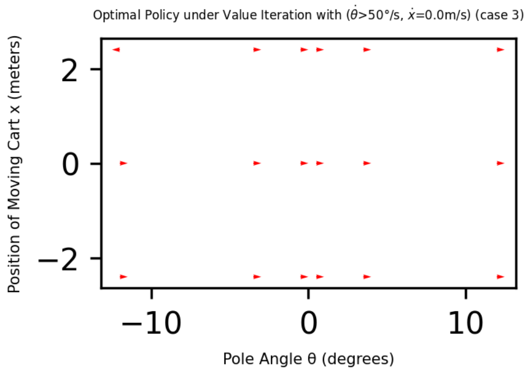
Case 2): Lowest bin for θ_dot and central bin for x_dot





Case 3): Highest bin for θ_dot and central bin for x_dot.





- Analysis of value function based on Policy iteration and plots

Policy iteration involves updating a policy array to maximize expected returns for each state. The value function shows two peaks for upright pole positions, indicating a preference for stable, vertical states, aligning with the goal of balancing the pole. In scenarios with high angular velocity, the value function increases with larger pole angles, suggesting the system values states allowing momentum-based corrections towards the vertical. Overall, policy iteration's value functions confirm the system's prioritization of near-vertical pole positions, consistent with the objective of maintaining balance.

- Analysis of value function based on value iteration and plots

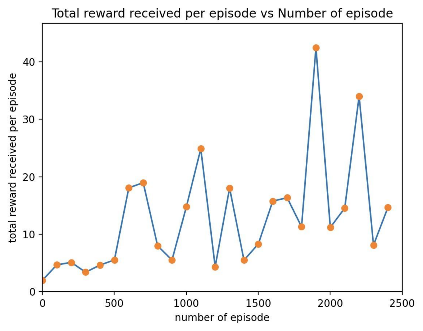
Value iteration updates the value function iteratively until convergence, then derives an optimal policy based on this final value function. In the cart-pole system, the value function peaks around the pole's upright position, showing a preference for stable, vertical pole states. For a rapidly falling pole, the value function favors states that leverage the pole's angular momentum for control. These findings align with the cart-pole system's objective of balancing the pole vertically, demonstrating the effectiveness of value function visualizations in capturing state desirability.

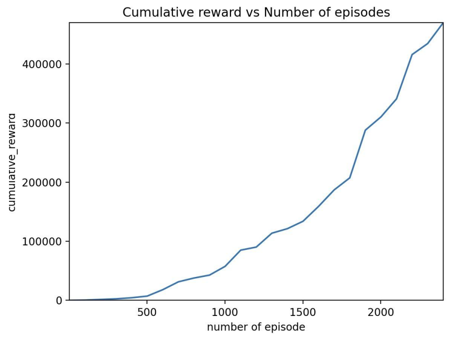
2.2 Temporal Difference (TD(0) or one-step TD) with Q-learning

The Outcome of TD 1&2

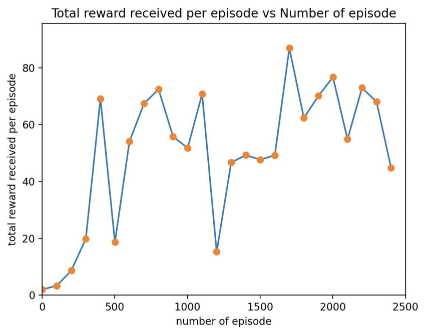
1:Plots of total reward received per episode vs. number of episodes, with three different discount rates.
2. Plot of the cumulative reward (the sum of all rewards received so far) as a function of the number of episodes, with three different discount rates.

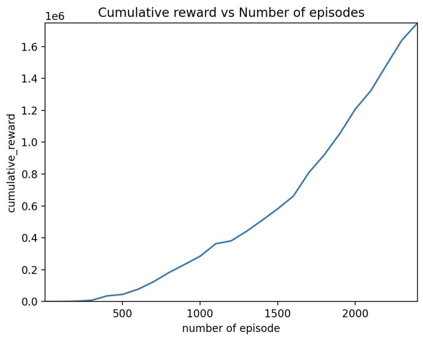
Case 1: Learning rates 0.01, Iteration 1-2500, Interval 100



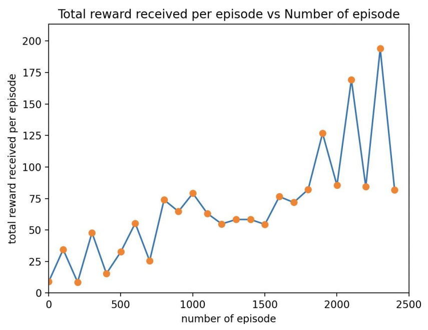


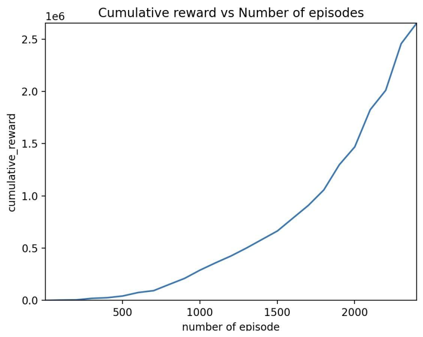
Case 2:Learning rates: 0.05 Iteration 1-2500, Interval 100





Case 3:Learning rates: 0.1 Iteration 1-2500, Interval 100





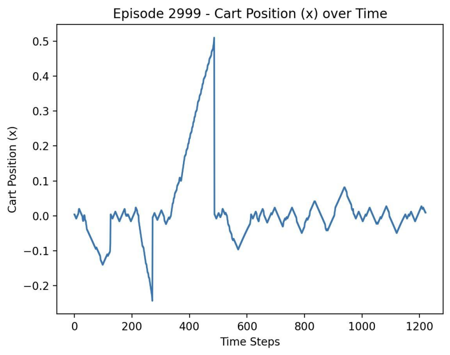
Analysis of of TD 1&2:

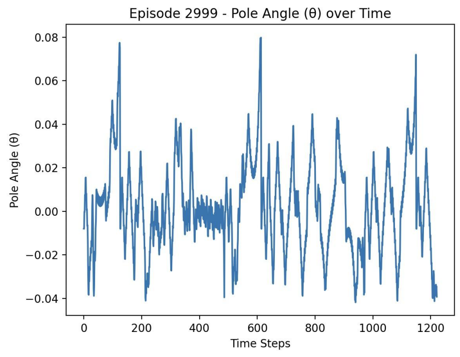
1.No matter what the value of the learning rate is, with the increase in the number of cycles, the value of the average reward also has an upward trend. 2. By comparing the three different learning rates, we find that when the learning rate increases, the total reward received per episode increases. This indicates that the training effect is better and the number of times the CartPole can maintain the balance is greater.

The Outcome of TD 3:

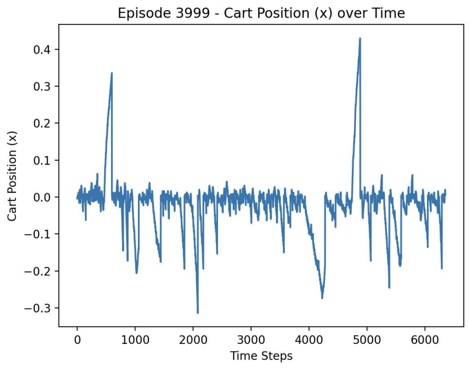
Plots of x and θ as functions of time for some successful episodes together with the number of time steps during which the pole does not fail). We use the reward value to select the best of all the loops.

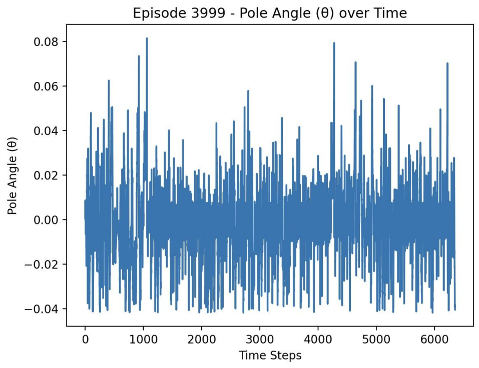
Case1:Learning rates: 0.05 Iteration: 2000





Case 2:Learning rates: 0.05 Iteration: 3000





Analysis of of TD 3:

While the learning rate is the same but iteration is different, the section x and θ that best maintains the balance of the CartPole are selected for presentation. It can be seen that when iteration is 3000, the CartPole will keep balance significantly longer, travel more to and from the center, and the Angle will also swing more to and from the center.

Team Contribution

Guangjing Zhu：Task 2 Temporal difference (TD(0) or one-step TD) with Q-learning coding, Task 2 deliverable
Siyuan Sun: Task 2 Dynamic Programming Coding (Value function of policy iteration and value iteration, optimal policy plotting. Task 2 deliverable.
Yile Wang: Task 1 Machine Learning data collecting and coding, Task 1 deliverable

Reference

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://github.com/stej4/CartPole_reinforcement_learning?tab=readme-ov-file
https://gym.openai.com