Anomaly Detection with Hidden Markov Model Reinforcement Learning

Group 15

CMPT 318 D100

Fall 2023

Amy Cao *301375613*

Christopher Peer *301377897*

Khaled Taseen *301457597*

Manmeet Singh *301476559*

***Abstract*—Supervisory Control Systems allows automation and enhances cost efficiency in a variety of fields to keep society functioning. However, reliance on automation introduces an increase in cyber risk such as the attack surface for advanced persistent threats and cascading effects. Thus, there is a need to detect anomalies in data streamed from Supervisory Control Systems. In this paper, Unsupervised Intrusion Detection is applied on stream data using Hidden Markov Models, time series analysis, and forecasting.**

CONTENTS

I Introduction ii

II Problem iii

III Methodology iii

III-A Data Sets iii

III-B Selection of Features with Principal Component Analysis iv

III-C Selecting Response Variables and Observation Time Window vi

III-D Training Hidden Markov Model vii

III-D1 Building the model vii

III-D2 Log-Likelihood . viii

III-D3 Bayesian Information Criterion value viii

III-E Testing Hidden Markov Model ix

IV Results ix

IV-A Anomaly Detection x

V Retrospective x

V-A Problems Encountered x

V-B Lessons Learned xi

V-C Problems Encountered in Q-Learning xi
V-D Lessons Learned from Q-Learning xii

VI Reinforcement Learning for Investment Decision Making xii
VI-A Model Implementation . . . xii
VI-B Hyperparameters xii
VI-C Training and Policy Derivation xiii
VI-D Results and Interpretation . . xiii
VI-D1 Q-Table Analysis xiii
VI-D2 Policy Interpretation xiii
VI-E Application to Unseen Data . xiv
VI-E1 Optimal Action Table for Unseen States xiv

References xiv

VII Contributions xv

I. Introduction

Technological advances lead to greater cost efficiency and quality of service in many fields of work. In particular, Supervisory Control Systems have been steadily introduced with increasing integration of computation, networking and physical processes among many industrial sectors vital for essential societal operations. Included are electric power grids, public water utilities, and smart transportation networks. As some of the sectors adopting Supervisory Control Systems are critical infrastructure sectors, the technological system must be robust and secure for any major disruption would cause severe consequences for public safety and national security. A major threat to Supervisory Control Systems is cyber threat.

As reliance on automation increases, so does the attack surface for Advanced Persistent Threats (APT) and the risk of cascading effects increases. APT try to remain undetected for the longest possible period; therefore, to mitigate damages caused by this form of attack, secure systems must have efforts in detecting anomalies in data.

This paper tackles the challenge of detecting anomalous data by employing tactics of using unsupervised models trained in a supervised model, namely a Hidden Markov model. The particulars of the Hidden Markov Model is that there are hidden states, and a set of observations. These hidden states are consistent in time, but the observed SCADA data will differ in some pattern by the hand of cyber attacks. We strive to determine the probabilities of a set of observations occurring given the hidden states and knowing the patterns of regular data. Various Hidden Markov Model models of different parameters is to be trained and tested, in which the top performing model selected to detect probability in anomalous data. The model features are extracted from a training data set using Principal component analysis (PCA). The metrics used for model selection are log-likelihood and Bayesian information criterion (Bayesian Information Criterion). With a well trained model that is tuned to real data, we can detect anomalies in data by observing the Hidden Markov Model model yield low probabilities of such anomalous data occurring.

II. Problem

Supervisory Control Systems have been steadily adopted into many industrial sectors vital for essential societal operations. With reliance on automation within critical sectors increases, so does the attack surface for Advanced Persistent Threats (APT) and the risk of cascading effects increases. The characteristic of APT is that they try to remain undetected for the longest possible period of time. In addition, existing vulnerabilities expose critical infrastructure to a range of adversarial scenarios. With the attack space increasing with the expanding digital space, the need to analyze cyber risk and create preventive strategies prevail. A type of defense against cyber attacks is detecting the attack. When there are external influences on a system, there may be anomalies in the data. The paper explores anomaly-detection based intrusion detection methods used for cyber situational awareness in the analysis of automated control processes.

III. Methodology

To detect anomalies in data, we need to know what regular data without anomalies is like. We approach the problem by training a machine learning model to pick up patterns of regular data, and then we apply it to new data and make predictions on how probable it is for the data sequence to occur. A large data set is used to train the model, and a smaller data set is used to test the model for under or over fitting. Of course, the original data sets must be transformed to fit the use case. The top-performing model will be used to detect anomalies in the 3 anomalous data sets to provide our research results.

As there are implicit states in the data "hidden" from us, we use a Hidden Markov Model to factor in the hidden states. We train various Hidden Markov Models with different parameters to find the optimal specification of fitting the data. The metrics used for determining how good the model is in regards to complexity and fitment will be log-likelihood and Bayesian Information Criterion.

A. Data Sets

All data sources used are provided by the Simon Fraser University CMPT 318 course. All data sets are a continuous stream of multivariate time series data as like Supervisory Control and Data Acquisition (SCADA) data. The contents are of normal electricity consumption data. The data structure has each row being an instance of date time separated in date, hours, and minutes. Therefore the index would be the "Date" and "Time" columns. In addition, each data point also has 7 attributes:

- Date
- Time
- Global_active_power
- Global_reactive_power
- Voltage
- Global_intensity
- Sub_metering_1
- Sub_metering_2
- Sub_metering_3

"Date" and "Time" values are all populated, but the remaining attributes may be NULL values. We handle the NULL values with linear interpolation by calling the package zoo na.approx() function [2] rather than omitting the entire data row. The rationale is that because we are working with time series data that is rich in quality (each row differs from the next by a second and there is approximately 3 years worth of data), the interpolation will be adequately accurate.

The baseline data set used for training and testing Hidden Markov Model models is about 3 years of data with 1556444 rows. In section III Methodology B. Extracting Features With Principal Component analysis, we discuss partitioning the baseline data set into train and test data sets, with 80 percent of the data for training and the remaining 20 percent for testing.

The ratio 80/20 was chosen as it follows the Pareto Principle and is a standard ratio for partitioning data sets into train and test. The Pareto Principle states that 80 percent of the benefit can be achieved by 20 percent of the effort. It also agrees that the remaining 20 percent of the benefit is achieved with 80 percent of the effort.

Once a top-performing Hidden Markov Model model has been selected, we allow it to make predictions on 3 other data sets with anomalous data. Each of these anomalous data sets have about a year of data; that is 518816 rows of data.

Date	Time	Global_active _power	...
16/12/2006	17:24:00	NA	...
16/12/2006	17:25:00	5	...
16/12/2006	17:26:00	NA	...

Table 1. Snippet of data set.

B. Selection of Features with Principal Component Analysis

Machine learning requires selecting good features for the model. In our usage, there are 7 attributes we can select to be a feature. However, not all of these attributes hold the same importance. We can either do feature selection among the 7 attributes, or find a method to select new features. We went with the feature select approach because it is included in the problem description set by the professor.

To obtain the features needed to train our Hidden Markov Model model, we employ Principal Component Analysis for feature selection, and then use the transformed features as responses for our Hidden Markov Model model.

Feature selection by Principal Component Analysis is a method of dimensional reduction to reduce the complexity of modeling large data sets while preserving as much information as possible. [7] In the R programming language, there are two methods for doing PCA. The first being the "prcomp" function, which does a singular value decomposition of the covariance and correlations between variables. The second method is the "princomp" function, which uses eigen decomposition of the covariance and correlations between samples. [5] We used the "prcomp" function because it is generally the preferred method for numerical accuracy. [6]

PCA selection can be used on features that are correlated. [8] We first confirm that the original features are correlated by forming a correlation matrix. The visualization in Fig. 1 can be can be interpreted as follow:

- The higher the value, the most positively correlated the two variables are
- The closer the value to -1, the most negatively correlated they are

It is determined that the features are correlated and we may proceed with preparing the data for PCA.

We prepare the data for PCA by scaling the baseline data set. This helps standardise our data sets later

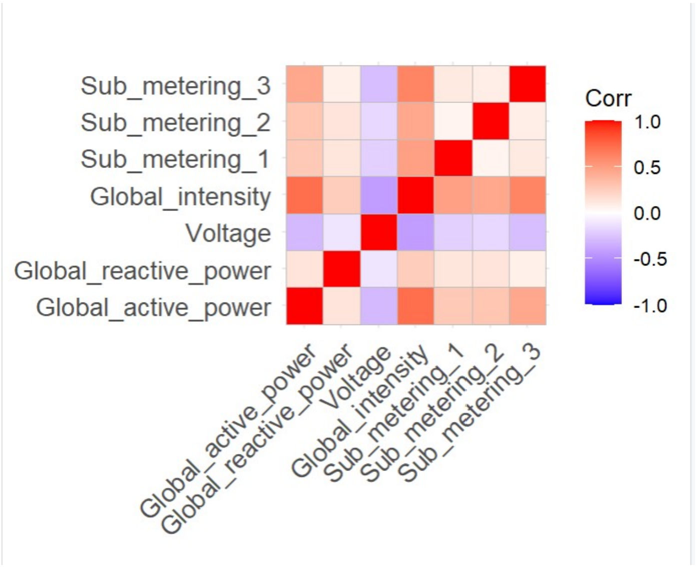


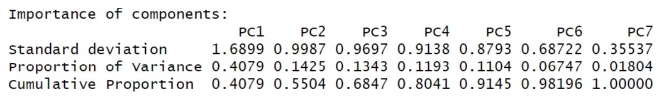
Fig. 1. Correlation Matrix of Features

when we split the baseline data set into testing and training data sets. To scale is to normalize the data by calculating the mean and standard deviation of the entire vector, and then subtract the mean and divide by the standard deviation for each element in the vector.

$$z = (value - mean)/standarddeviation \quad (1)$$

Next, we call the "prcomp" function which will return return a Prcomp object. We view the summary of the cumulative amount of variance explained relative to the total variance of the principal components in the Prcomp object.

As seen in the summarized table of explained variances of the principal components from Fig. 2, it is



```
Importance of components:
                          PC1    PC2    PC3    PC4    PC5     PC6     PC7
Standard deviation     1.6899 0.9987 0.9697 0.9138 0.8793 0.68722 0.35537
Proportion of Variance 0.4079 0.1425 0.1343 0.1193 0.1104 0.06747 0.01804
Cumulative Proportion  0.4079 0.5504 0.6847 0.8041 0.9145 0.98196 1.00000
```

Fig. 2. Principal Component Analysis Summary

observed that the principal components PC1 has the largest explained variance of 40.8%.

A Scree plot can be used to visually represent the summarized table in a lined bar graph. The Term Project description set by the professor advises a selection of two to three features. It is observed from Fig. 3 that PC1 covers a large portion of the variance in the data. By ranking the variables in the data set by highest absolute loading score for Principal Component 1 Fig. 5, the top three scoring variables were select to be response variables. Ranking the loading scores can be done in many ways, depending on how many principal components are included in the calculation. Using only Principal Component 1 to rank the variables gives a good summary of model features without introducing excessive amounts variance.

At first the top three variables were selected, but with covariance analysis the decision was made to use only the top two. 4

It holds that the top two contributors to the principal components would be the following features from the original data:

- Global_intensity

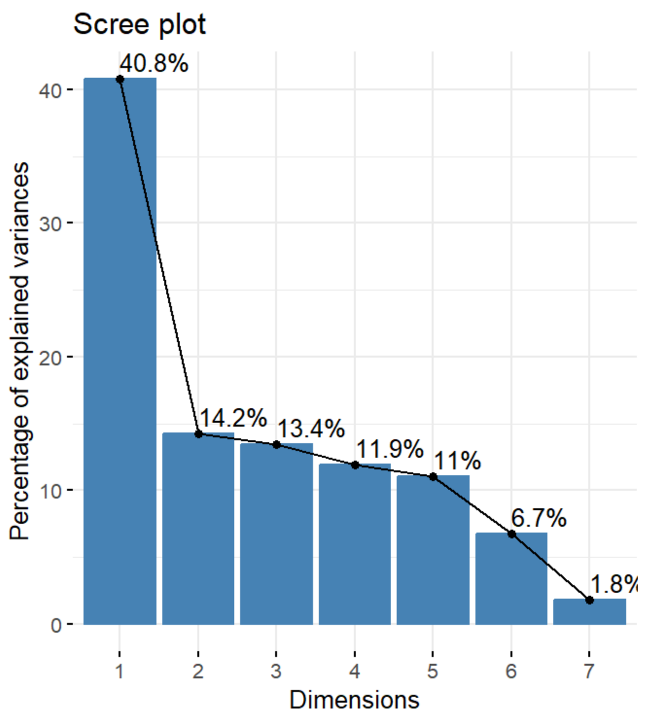


Fig. 3. Scree Plot of Principal Components

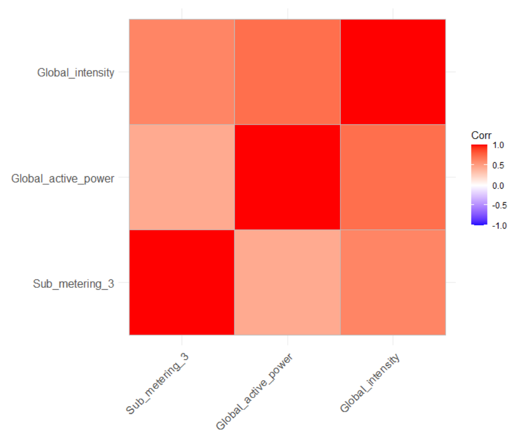


Fig. 4. Correlation Matrix between responses

- Global_active_power

C. *Selecting Response Variables and Observation Time Window*

After standardising and scaling our baseline data set, we need to further process the data set by selecting a time window and splitting the baseline into train

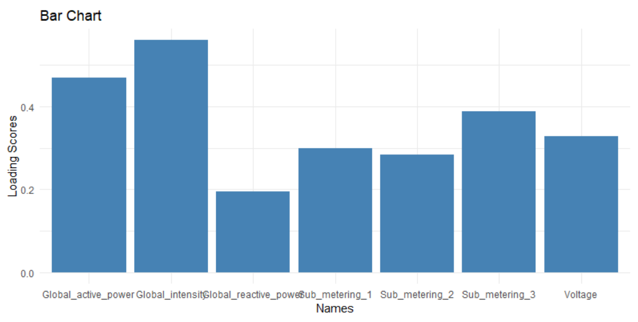


Fig. 5. Square Cosine Plot

and test data sets.

We selected our window to occur every Wednesday from 8:00 to 14:00. Generally, consumers of electricity have the same routine during this time frame as they are at work or in school. Compared to the weekends where consumers may or may not be home, Wednesdays from 8:00-14:00 tend to hold a routine for many.

For the purposes of creating good training and testing data, we split our baseline data set by the following method.

1) We selected data from our baseline that occurs only within the window periods.
2) We grouped our windowed data by week so each group contained only a singular window
3) We created a 1D Boolean mask which had a length that matched the number of weeks in the data set, and populated it with true-false samples at a frequency of 80% True, 20% False.
4) We masked the grouped data set, any grouped data masked as true was concatenated into the training data set, any grouped data masked as false was concatenated into the test data set.

The result was a test data set and a training data set with windowed data sampled randomly and concatenated sequentially.

After conducting PCA we extract the 2 main features, Global_active_power and Global_intensity. By doing PCA analysis we have reduced the dimensions of our data and reduced the complexity of the model. We then window and mask these features in a process similar to splitting the test/train data, by reusing the binary mask from before. The windowed and sequentially concatenated features are then used as response variables in the Hidden Markov Model.

D. Training Hidden Markov Model

1) Building the model

Hidden Markov Models require the number of hidden states as a parameter. As we are unable to gleam such insights with what we have, we proceeded with a brute-force solution to determine the number of hidden states.

To find the optimal number of hidden states for our model, we fitted a model for every multiple of 4 hidden states (up to 24 states) and recorded the calculated log-likelihood and Bayesian Information Criterion values. These two metrics are used to determine the best model. The top performing model will be selected to make predictions on anomalous data sets.

We want to strike a good balance between the log-likelihood and Bayesian Information Criterion values, for increasing the number of states as a parameter will increase log-likelihood, but also increase the Bayesian Information Criterion values.

By sparsely iterating over the range of possible hidden states, we can gather data on optimal fitment characteristics without fitting an excessive number of models. The optimal number of hidden states is the intersect of log-likelihood and Bayesian Information Criterion values plotted out on a graph. From the data gathered in Fig. 6, we extrapolated the optimal number of hidden states to train our model with as nstates = 12

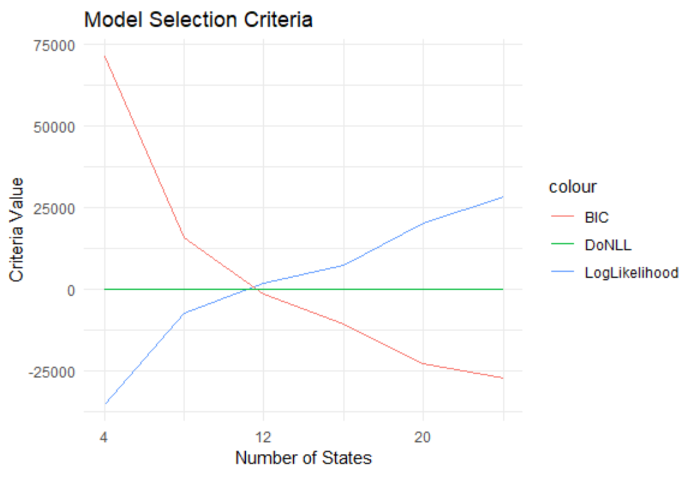


Fig. 6. Log-likelihood and Bayesian Information Criterion over nstates

2) *Log-Likelihood*

The log-likelihood value generated by model fitment allows us to analyse the fitment of the training data to a given model. It describes the likelihood of a sequence of observations occurring given the parameters of the fitted model. If this number is too low, it means the model cannot adequately characterise our training data. If this number is too high the model may over-perform and won't be able to generalise to unseen data. Over-performing is also known as over-fitting. The goal is to have an appropriately fitting model as described in Fig. 7. [4]

	Log-likelihoods by nstates					
n:	**4**	**8**	**12**	**16**	**20**	**24**
l:	-35396	-7383	1793	7071	62013	98345.679

Table 2. Log-likelihood values for nstates

3) *Bayesian Information Criterion value*

The Bayesian Information Criterion is calculated using the following equation:

$$-2log(L) + k \cdot log(o) \tag{2}$$

where log(L): log-likelihood of the model fitment, k: number of model parameters, o: number of observations. A well fitted model will minimise the Bayesian Information Criterion value. Bayesian Information Criterion is a more nuanced performance metric than just log-likelihood. The Bayesian Information Criterion criterion awards good fitment likelihood, and penalizes excessive model complexity which causes over-fitment.

By analysing our data we found a range of optimal hidden states that would produce a model with a good fitment on our training data, and low enough complexity to generalise to our testing data. For our model we chose nstates=12; this produced a log-likelihood: train.logLik : 4956.416 and a Bayesian Information Criterion: -3145.243

	BIC Values by nstates					
n:	**4**	**8**	**12**	**16**	**20**	**24**
BIC:	71125	15784	-1538	-10724	-11889	-18950

Table 3. Bayesian Information Criterion Values for nstates

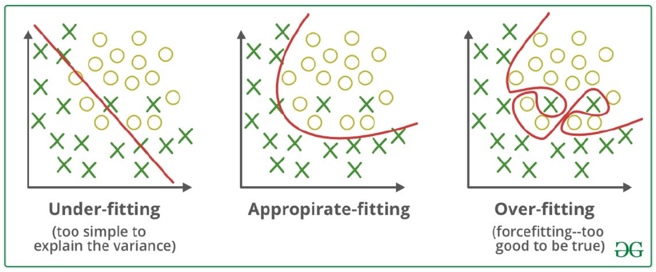

Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Fig. 7. Overfitting vs. Underfitting

E. Testing Hidden Markov Model

After training, the model is compared against unseen test data using the forward backward function.

A test model was made from response variables extracted from the scaled test data. The parameters were extracted from the trained model and fitted to the test model. The forward-backward function is then run on the test model, which is an implementation of Baum-Welch algorithm. This does a forward pass and a backward pass over a set of probability matrices within the test model, and fits it to the parameters of the trained model. By extracting the values from the lamba (sca) matrix and summing the negative log of the values, the log-likelihood for a given set of observations to be consistent with this model is generated. A small log-likelihood would indicate that the set of observations is not consistent with the model, a large log-likelihood would indicate the observations are consistent.

The results as followed:

test Log Like : 2097.12 normalised Log Like : 0.2008735

By the performance characteristics measured indicates that the model fitment is acceptable for the number of states it has. The previously unseen test data also fits the model with a normalised log-likelihood higher than the model itself. This indicates the model is good at detecting non-anomalous data, implying it can differentiate between anomalous and non-anomalous data.

IV. Results

Model fitment performance values: train log Likelihood : 4956.416 normalised Log Likelihood : 0.1101426

Test data evaluation: test Log Likelihood : 2097.12 normalised Log Likelihood : 0.2008735

From these results we can see the model fitment is sufficient

A. Anomaly Detection

Similar to the testing fitment, a model is created for each anomalous data set by extracting response variables from the scaled and windowed data. The data is scaled to the characteristics of the scaled train data set. The forward backward function is applied to get log-likelihoods of the consistency of the anomalous data with the trained model.

anom1 Log-Like : -334807.2 anom1 fitment : -17.885

anom2 Log-Like : -334050.1 anom2 fitment : -17.84455

anom3 Log-Like : Nan, the forward backward algorithm encountered Nan values, indicating the attempt at fitment created extreme values within the model parameters, caused by poor fitment with the model fitted on normal data. This indicates the presence of a significant anomaly. anom3 fitment : fitment was calculated from the Log-Likelihood

In all three cases the log-likelihood is exceptionally small, indicating some form of anomaly is present in each of the three three data sets.

V. Retrospective

Upon the conclusion of the research in detecting anomalies in electric consumption data with hidden states, many thoughts have been gathered in retrospective. There had been abundant problems encountered, with some easily resolved, but with others requiring hours of discussion and thought. We hope to share insights on some of the problems encountered and how they could have been prevented, and what we have learned in the process of understanding.

A. Problems Encountered

There is only code file shared amongst the team to handle the Unsupervised Intrusion Detection. This caused difficulties in version control, as multiple people were making changes to the same lines which induced merge conflicts. Ideally, each member of the team works on a different part of the project in a parallel work form, but that is difficult to do as many of the tasks required a step by step process. To reduce the redundant work of fixing merge conflicts or overlapping work, we opted to do pair programming instead for the Unsupervised Intrusion Detection. While the Hidden Markov Model models were training, we would switch focus towards writing the report to keep efficiency high.

Another problem encountered was the lack of theoretical understanding. A lot of time was spent reading articles, documentations, and papers to gain an understanding of PCA and Hidden Markov Model, and applying it to our use case. For instance, there was a lot of confusion on the difference between feature selection and feature extraction regarding PCA. We also had difficulties interpreting log-likelihood results as sometimes we'd obtain positive values, and sometimes negative. In addition, the team does not have technical background on the R programming language, so there was a substantial amount of learning required to get small amounts of progress. Of course, the productivity of the team increased over time as we became more familiar with the techniques and technical skills required.

Hardware and time constraint was also a considerate problem in the development of the project. We trained various Hidden Markov Model models on modest hardware, that is laptops with the specifications of ordinary consumers. As such, training the models would take at least an hour, and we would repeat the process multiple times to ensure the code is working as expected. With the long wait time between each incremental attempt, progress was hindered. This led to a greater need for optimization, and how we could modify the code to speed up the progression.

Understanding the project requirements was also a challenge. Each time the team was confused or uncertain about the problem description or defined approach, we would have to email the Teaching Assistants of the course and wait for a response, or wait until lecture hours to discuss in person. The short delay in acquiring answers to questions was also a factor in the timeline of the project progress.

B. Lessons Learned

We learned that effective communication is vital for group work. By distributing tasks early on and communicating who is attending to which task, we can prevent version control issues and streamline the work process.

Code optimization was also a lesson learned. By modifying the code slightly, we can reduce the time taken to run the process. If we were to approach the project differently, we could find a method for discovering the number of hidden states of a Hidden Markov Model rather than using a brute-force method of checking each incremented nstate for the optimal model. Conversely, such a task will require in-depth research and may consume much more time than the brute-force approach.

We also learned more about machine learning processes and theoretical knowledge in data science. We can apply this knowledge in future projects.

C. Problems Encountered in Q-Learning

During the implementation of the Q-learning algorithm, we faced several challenges. The complexity of the model required meticulous tuning of hyperparameters to balance the trade-off between exploration and exploitation effectively. Debugging the model to ensure correct updating of the Q-table was time-consuming, as it involved tracking the state transitions and reward assignments over numerous epochs.

Another challenge was the integration of the Q-learning model with our data preprocessing pipeline. Ensuring that the state space and action sets were correctly mapped and utilized within the Q-learning framework required careful coding and testing.

D. Lessons Learned from Q-Learning

The Q-learning part of the project taught us valuable lessons in reinforcement learning. We learned that the choice of hyperparameters significantly influences the learning process and the quality of the resulting policy. Small adjustments to these parameters could lead to vastly different outcomes, emphasizing the need for a thorough understanding of their roles.

The project also highlighted the importance of a robust data preprocessing routine. Preparing the data in a manner suitable for reinforcement learning, such as ensuring the correct format for states and actions, was crucial for the success of the model.

Furthermore, we learned the practical aspects of implementing a reinforcement learning model from scratch, such as the importance of iterative testing and validation. The experience gave us insight into the nuances of model training and the patience required to achieve a well-performing agent.

VI. Reinforcement Learning for Investment Decision Making

A. Model Implementation

Our reinforcement learning model employs the Q-learning algorithm, a model-free reinforcement learning technique used to find the optimal action-selection policy for a given Markov decision process. In our scenario, the agent makes investment decisions across various sectors like Stocks, Real Estate, Commodities, Cryptocurrencies, and Forex, with budgets ranging from 1 to 100 million dollars.

The Q-learning model's state space consists of integer values representing different budget levels. The actions correspond to different investment choices. The rewards are computed based on the profit generated from these investments, with profits being random numbers within specified limits. The Q-table, a central component of Q-learning, is initialized to zeros for all state-action pairs.

B. Hyperparameters

The model includes several hyperparameters that influence its learning behavior:

- **Alpha** (α) **- Learning Rate**: Set at 0.1, this parameter determines the extent to which new information affects the existing Q-values. It balances the importance of new and past information.

- **Gamma (γ) - Discount Factor**: With a value of 0.9, it quantifies the importance of future rewards in the agent's decision-making process.
- **Epsilon (ϵ) - Exploration-Exploitation Trade-off**: Not used in the model since the model is designed to prioritize exploitation over exploration and since the rewards landscape is relatively stable and well-known.

C. Training and Policy Derivation

We generated a training dataset based on simulated experiences and trained the Q-learning agent over multiple epochs. The policy, derived from the Q-table, indicates the optimal action for each state. It was computed by identifying the action with the highest Q-value in each state.

D. Results and Interpretation

1) Q-Table Analysis

The Q-table post-training shows the learned values of taking specific actions in given states. The table reflects the agent's understanding of the most beneficial actions at different budget levels, shaped by the balance between immediate and future rewards.

2) Policy Interpretation

Policy table output:

```
[1 3 1 0 3 3 3 1 3 2 2 1 1 1 1 0 4 4 4 2 2 0 2 2 3
0 1 3 2 2 1 3 1 1 1 0 2 2 3 4 4 0 1 1 0 2 1 1 4 4
3 3 1 1 2 2 1 0 4 1 0 2 2 1 0 1 1 0 1 1 3 4 2 4 1
0 1 1 0 2 3 2 1 2 3 3 2 1 3 1 1 4 0 0 2 2 2 2 0 0]
```

The policy array provided represents the optimal actions to take for each state in the state space, as determined by the Q-learning model. The array is zero-indexed, meaning that the action for state 1 is at index 0, for state 2 is at index 1, and so on up to state 100 (if you have 100 states). Each number in the array corresponds to an action from the list of actions ['Stocks', 'Real Estate', 'Commodities', 'Cryptocurrencies', 'Forex'] indexed as [0, 1, 2, 3, 4], respectively

Here's how you can interpret the policy:

Adaptive Decisions: The policy suggests which investment option is considered optimal for each budget level (state). A change in the number indicates a change in the optimal investment strategy as the budget increases.

Patterns in Investment Strategy: If certain numbers appear more frequently in contiguous ranges, it may indicate that certain investment strategies are preferred for certain budget levels. For example, if '0' appears frequently at lower budget levels, it may suggest that 'Stocks' are considered the best investment for lower budgets.

Diverse Strategies: The presence of all action indices in the policy implies a diverse range of investment recommendations, suggesting that the agent does not overly commit to one type of investment but rather adapts its strategy based on the specific state.

E. Application to Unseen Data

When applied to unseen data, the model predicts optimal investment actions for new budget scenarios. This showcases its capability to generalize learning and provide decision-making support in unexplored financial situations.

1) Optimal Action Table for Unseen States

TABLE I
OPTIMAL ACTIONS FOR UNSEEN STATES

State	Optimal Action	State	Optimal Action
15	Real Estate	31	Real Estate
16	Stocks	32	Cryptocurrencies
17	Forex	33	Real Estate
18	Forex	34	Real Estate
19	Forex	35	Real Estate
20	Commodities	36	Stocks
21	Commodities	37	Commodities
22	Stocks	38	Commodities
23	Commodities	39	Cryptocurrencies
24	Commodities	40	Forex
25	Cryptocurrencies	41	Forex
26	Stocks	42	Stocks
27	Real Estate	43	Real Estate
28	Cryptocurrencies	44	Real Estate
29	Commodities	45	Stocks
30	Commodities		

The optimal action table for unseen states reveals the model's recommendations for investment types at various budget levels. This insight can aid in strategic decision-making in investment scenarios, highlighting the potential application of reinforcement learning in finance.

REFERENCES

[1] https://cran.r-project.org/web/packages/depmixS4/vignettes/depmixS4.pdf

[2] https://www.rdocumentation.org/packages/zoo/versions/1.8-12/topics/na.approx

[3] https://arxiv.org/abs/2004.13914

[4] towardsdatascience.com

[5] https://docs.oracle.com/cd/E83411_01/OREUG/principal-component-analysis.htm#OREUG-GUID-BA6F4694-92C3-4E9D-9FC6-766F518E9E12

[6] https://stat.ethz.ch/R-manual/R-patched/library/stats/html/prcomp.html

[7] https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[8] https://datascience.stackexchange.com/questions/89522/is-there-any-different-between-feature-selection-and-pca-if-there-is-could-a
~:text=(2)%20PCA%20can%20be%20used,from%20among%20the%20transformed%20features.

[9] https://www.datacamp.com/tutorial/pca-analysis-r

VII. Contributions

- Amy Cao 301375613

  Part 1: 30% Part 2: 50% Part 3: 0%
- Christopher Peer 301377897

  Part 1: 70% Part 2: 20% Part 3: 0%
- Khaled Taseen 301457597

  Part 1: 15% Part 2: 15% Part 3: 50%
- Manmeet Singh 301476559

  Part 1: 15% Part 2: 15% Part 3: 50%