**ASSESSMENT 2:**

**K-Means Clustering**

**COMP3003**

**Machine Learning**

**2022/2023**

Table of Contents

[Introduction 2](#_Toc124682852)

[Literature Review 3](#_Toc124682853)

[Implementation 7](#_Toc124682854)

[Results, and Performance Analysis 7](#_Toc124682855)

[Discussions and Conclusions 7](#_Toc124682856)

[References 8](#_Toc124682857)

[Appendix 10](#_Toc124682858)

# Introduction

In the first section of this report, a literature review is carried out with a focus on Supervised Learning (SL) and Reinforcement Learning (RL). These are both Machine Learning (ML) paradigms. In the literature there is an investigation into the definition of each paradigm, its real-world applications, and the limitations/major concerns of the paradigm in its current state.

In the second section of the report, an SL application is developed, and the implementation steps discussed. The application being developed in this section is a network intrusion detector based upon the KDD99 labelled dataset. The learning problem being tackled in this scenario is *classification*, which will be implemented through a neural network classifier. In this section, two implementations of the classifier are made: one with PCA and one without.

In the third section of the report, the results of the implementations are discussed, and the performance of the implementations are analysed to investigate the impact of PCA. The impact of any assumptions is also discussed within this section.

In the fourth, and final, section of the report, the findings of this experiment are discussed, and the final conclusions are made. The importance of the results is emphasized and explanations into why those specific results are found.

# Literature Review

**Introduction**

Machine Learning (ML) is the most important component of Artificial Intelligence (AI), declared by Zhang (2020), and irrefutably so, as proven by ML’s omnipresence within all manners of modern industry, such as healthcare, manufacturing, education, financial modelling, policing, and marketing (Jordan, 2015). This is a reasonable deduction when the goals of ML are considered:

1. Automate decision making and data analysis tasks.
2. Reduce the burden on developers to write extensive amounts code.

As Simeone (2018) states, ML is used to let “large amounts of data dictate algorithms and solutions”, which is endearing from a business perspective as well-defined outputs are generated efficiently, without the need of expertise and large codebases that manually specify algorithms and rules. The result is that implementing ML paradigms is often cheaper and quicker than traditional software programming. However, ML is not free of flaws and Plasek (2016) asserts that the biggest errors within ML originate from poor choices of training data, with the impact of unintentionally coding systemic discriminations against race, gender, and ethnicity and other biases into software.

Supervised Learning (SL) and Reinforcement Learning (RL) are two ML paradigms. The study will be conducted with the goal to develop an understanding of SL and the specific tasks that the paradigm is modelled for. RL will be analysed alongside SL to juxtapose the key differences between the two paradigms, and for each paradigm real-world scenarios will be discussed.

**Supervised Learning**

From reading various sources it can be understood that SL is defined as a category of algorithms that tackle specific learning problems wherein the goal, as described by Dangeti (2017), is to train an algorithm to map the relationship between an input variable and an output variable, also known as a target or label. Therefore, the objective of this algorithm is to be able to predict the value of a label for inputs that aren’t in the training set, otherwise described as ‘unobserved’ (Simeone, 2018).

Bishop (2016) and Jordan (2015) describe these input variables as vectors, which can represent complex entities, such as documents, graphs, or images etc. Simeone (2018), Jordan (2015), and Rashidi (2021) state that the output of the algorithm determines the type of learning problem. An output can be discrete, wherein the problem is Classification, and the model is generalizing observations within the dataset to predict which class/group the input belongs to. An output can also be continuous, in which the problem can be recognised as Regression.

Yongquan (2020) asserts that SL can be further decomposed and classified into weakly supervised learning, moderately supervised learning, precisely supervised learning, and a combination of the latter two. In his paper he asserts that the definition of SL is too abstract due to the properties (completeness, exactness, and accuracy) of the labels, and therefore exposing these properties by defining different subcategories of SL enable engineers to better design solutions. Researching the details of these subcategories is out of the scope of this study, however Yongquan (2020) does provide insight into how labels are learnable targets, but in some cases, it can be much more complicated as labels aren’t ideal and first need to be transformed. Yongquan (2020) states that “ideal” infers completeness, exactness, and accuracy. Put simply, each input/entity is individually assigned with a label, and the label accurately describes the ground-truth of the input. Chen *et al.* (2018) discusses how labelling is usually done manually and with such large quantities of data can be time--consuming and require expertise to be done accurately. Overall, both sources indicate that within SL the dependency on labelling is significant. Models are dependent on the data, and therefore if the data is inaccurate, the model is likely to return inaccurate predictions, which can be highly consequential in systems that are reliant on these predictions, such as cancer diagnosis.

Classification is the process of categorizing a dataset into classes. Simeone (2018) states that classification can be divided into binary and multi-class problems. Examples of classification models are (Dangeti, 2017):

* Neural networks
* K-nearest neighbours
* Naïve Bayes
* Decision trees
* Support Vector Machines

Classification is regularly used within health care as agents that can make decisions alongside doctors. At the current stage these algorithms are not reliable enough to fully depend on and are used in conjunction with certified individuals such as in the case of analysing specific images/scans to detect, for example, cancer within patients (Rashidi, 2021). Use of SL in these circumstances enables doctors to make better informed judgements quicker and efficiently. Rashidi (2021) notes that the use of ML within healthcare is flawed due to legal reasons wherein data is not readily available. The data in this context is likely to be of previous patients. Another availability consideration is when such data does not already exist, such as in the case of rare diseases. With limited training data, the reliability of the models become questionable.

On the other hand, regression models are used to predict behaviours, by modelling statistical relationships between predictor and response variables (Gonzalez, 2019), where the latter is influenced by the former. Crime prediction is a specific field in which regression is popularly used. These systems are used to predict the likelihood of specific individuals to commit crimes. Studies frequently calculate this likelihood via applying a score that is reached via weighing socio-economic characteristics and criminal history. Trinhammer (2022) extends these models to also incorporate psychiatric history. Using these systems allows for institutions to allocate resources (Gonzalez, 2019)) to individuals to potentially help rehabilitate them or inversely keep them from endangering the public.

**Reinforcement Learning**

RL is the process of finding the most optimal sequential decisions to make within an environment based on rewards and punishments. Upon taking actions the agent will be given immediate feedback and the state of the environment changes (Simeone, 2018). RL can be generally described as algorithms that are influenced by a history of previous actions and their ‘score’. Therefore, RL is different from SL as the optimial solution is not provided, and it must find it itself through repeated simulations.

RL gained popularity through its use within board games (), which then spread to modern game development, and the development of Non-Playable Characters (NPCs). It can be understood that RL was not a popular approach in early software development due to it often requiring large amounts of time and computation to train and observe.

The most notable turning-point in the development of RL was the AlphaGo project (Chen, 2016). This project aimed to make an AI that could play Go completely without human intervention. In 2015 AlphaGo beat the world champion player (Holcomb, 2018), proving the effectiveness of reinforcement learning and its ability to create responsive agents.

Furthermore, a more recent development of RL is within robotics and autonomous, self-driving vehicles. These agents are often deployed within highly variable environments and therefore there is the challenge of deploying them safely. Kormushev (2013) researched into a variety of concerns of reinforcement learning within robotics. They state that the policy needs to be firmly established to consider requirements that determine the following characteristics: smoothness, safety, gradual exploration, scalability, compactness, adaptability etc. One such task that Komushev (2013) mentions is the seemingly trivial act of making a robotic arm flip a pancake in a pan. In this project, the robot is shown a perfect demonstration of what this looks like and needs to reproduce what it is being shown via trial and error. Dropping the pancake may result in a punishment and therefore the specific motor movements that all build that one action will be scored low. However, if the pancake lands in the pan this is rewarded. In the cases where the pancake lands in the pan but bounces out, the punishment given isn’t as major. This whole process can be time intensive.

**Conclusion**

Overall, SL is about training models to predict or classify data via looking at hidden patterns and the relationships between inputs and targets. Through generalising these relationships, for entities that are not already within the dataset, labels/ targets can be predicted automatically and decisively. RL differs from this as it is used to train agents within specific environments through a trial-and-error, reward-based approach. Through providing a series of actions a score, an agent can discover the most optimistically feasible set of actions to enable reproducibility of a task.

Both have many consequences and are suited for different needs, such as SL is highly desirable within the healthcare industry, but RL is not widely applied (Rashidi, 2021). Likewise, RL has seen major advancements within the modern gaming industry, where SL is not as popular.

Through this research, the different applications and drawbacks of both SL and RL have been discussed. However, with their drawbacks, both paradigms are still being developed and new applications that use these models as their foundation are still being released. ML is still growing and both SL and RL will be widely considered to be integral parts of modern coding, with direct, traditional programming becoming slightly more undesirable in corporate settings as it is a lengthy process.

# Implementation

The focus of this section of the report is to develop a network intrusion detector. The application is based upon the KD99 Intrusion Detection Dataset, sourced from the following link:

# Results, and Performance Analysis

B

# Discussions and Conclusions

B

# References

Bishop, C.M. (2016) Pattern recognition and machine learning. New York, NY: Springer New York.

Chen, J.X. (2016) “The evolution of computing: Alphago,” Computing in Science & Engineering, 18(4), pp. 4–7. Available at: https://doi.org/10.1109/mcse.2016.74.

Chen, Z. and Liu, B. (2018) Lifelong machine learning. San Rafael, CA: Morgan & Claypool.

Dangeti, P. (2017) Statistics for Machine Learning: Techniques for exploring supervised, unsupervised, and reinforcement learning models with python and R. Birmingham: Packt Publishing Ltd.

Gonzalez, J. and Leboulluec, A. (2019) “Crime prediction and socio-demographic factors: A Comparative Study of machine learning regression-based algorithms,” Journal of Applied Computer Science & Mathematics, 13(1), pp. 13–18. Available at: https://doi.org/10.4316/jacsm.201901002.

Holcomb, S.D. et al. (2018) “Overview on DeepMind and its AlphaGo Zero AI,” Proceedings of the 2018 International Conference on Big Data and Education [Preprint]. Available at: https://doi.org/10.1145/3206157.3206174.

Jordan, M.I. and Mitchell, T.M. (2015) “Machine learning: Trends, Perspectives, and prospects,” Science (American Association for the Advancement of Science), 349(6245), pp. 255–260. Available at: https://doi.org/10.1126/science.aaa8415.

Kormushev, P., Calinon, S. and Caldwell, D. (2013) “Reinforcement learning in robotics: Applications and real-world challenges,” Robotics, 2(3), pp. 122–148. Available at: https://doi.org/10.3390/robotics2030122.

Plasek, A. (2016) “On the cruelty of really writing a history of machine learning,” IEEE Annals of the History of Computing, 38(4), pp. 6–8. Available at: https://doi.org/10.1109/mahc.2016.43.

Rashidi, H.H. et al. (2021) “Machine Learning in Health Care and Laboratory Medicine: General overview of supervised learning and auto‐ml,” International Journal of Laboratory Hematology, 43(S1), pp. 15–22. Available at: https://doi.org/10.1111/ijlh.13537.

Simeone, O. (2018) A brief introduction to machine learning for engineers. Boston: Now Foundations and Trends.

Trinhammer, M.L. et al. (2022) “Predicting crime during or after psychiatric care: Evaluating machine learning for risk assessment using the Danish patient registries,” Journal of Psychiatric Research, 152, pp. 194–200. Available at: https://doi.org/10.1016/j.jpsychires.2022.06.009.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare and Joelle Pineau (2018), “An Introduction to Deep Reinforcement Learning”, Foundations and Trends in Machine Learning: Vol. 11, No. 3-4. DOI: 10.1561/2200000071.

Yongquan, Y. (2020) Moderately Supervised Learning: Definition, Framework and Generality [Preprint]. Available at: https://doi.org/10.48550/arxiv.2008.11945.

Zhang, C. and Tong, H. (2020) “Research the application of Artificial Intelligence Technology in Mobile Phone Terminal,” Journal of Physics: Conference Series, 1648(2), p. 22024. Available at: https://doi.org/10.1088/1742-6596/1648/2/022024.

# Appendix

**main.m**

clc

clear

% Import dataset - read into table

filename = "cw\_dataset\_processed.txt";

data = readtable(filename); % 3000 x 42 samples

% -- DATA PREPARATION START -- %

% 1. Remove duplicate samples

data = unique(data, "stable"); % 2854 x 42 samples

% 2. Remove features with same value

toRemove = {}; % Initialise array

for i=1:width(data) % For all columns

column = data(:,i); % Get column at index

% If 1 then all values in the column were duplicates

if height(unique(column)) == 1

toRemove = [toRemove, i]; % Store indexes of duplicate columns

end

end

% Remove each column at index

for i=length(toRemove):-1:1 % Decrement because otherwise indexes will change

index = toRemove{i};

data(:,index) = []; % 2854 x 34 samples

end

% 3. Normalization

w = width(data); % Cant normalize last column (string)

norm = data(:,1:w-1); % So make a copy of the dataset without last col

norm = normalize(norm); % Normalize data

data = [norm, data(:,w)]; % Add last column back to now normalized data

% 4. Shuffle the data

h = height(data); % Get number of rows

shuffle = data(randperm(h),:); % Shuffle rows

classes = shuffle(:,w); % Store targets seperately

data = shuffle(:,1:w-1); % Remove last column again

% -- DATA PREPRATION COMPLETE -- %

% -- PCA -- %

warning('off', 'stats:pca:ColRankDefX');

dm = data{:,:}; % table to matrix

[~, score, latent, ~, explained] = pca(dm); % PCA on matrix

figure(1);

bar(latent,'b');

title("PCA Analysis for KDD99");

xlabel("PCA Components"); ylabel("Variance");

figure(2);

plot(cumsum(explained), 'k-');

title("Cumulative Sum of Variance");

xlabel("PCA Components"); ylabel("Percentage of Variance");

figure(3);

grp = table2array(classes); % table to array targets

idxnormal = find(contains(grp, 'normal')); % find all normal

idxportsweep = find(contains(grp, 'portsweep')); % find all portsweep

idxneptune = find(contains(grp, 'neptune')); % find all neptune

% Plot normal

x = score(idxnormal, 1);

y = score(idxnormal, 2);

plot(x,y, 'bo'); hold on;

% Plot portsweep

x = score(idxportsweep, 1);

y = score(idxportsweep, 2);

plot(x,y, 'ro'); hold on;

% Plot neptune

x = score(idxneptune, 1);

y = score(idxneptune, 2);

plot(x,y, 'go');

title("Principal Component Analysis (PCA)");

xlabel("PC1"); ylabel("PC2");

legend('normal', 'portsweep', 'neptune');

x = score(:,1:2); % Reduce dimensionality from 33 to 2

% -- PCA COMPLETE -- %

% -- FURTHER PREPARE THE DATA FOR NN -- %

totalLabels = length(grp); % Total number of samples

targetsMulti = zeros(3, totalLabels); % Multi classification

targetsBinary = zeros(2, totalLabels); % Binary classification

% Convert labels to numerical representation

% 1 if that sample belongs to that class

for i=1: totalLabels

if contains(grp(i), "normal") == 1

targetsMulti(1,i) = 1;

targetsBinary(1,i) = 1;

end

if contains(grp(i), "portsweep") == 1

targetsMulti(2,i) = 1;

targetsBinary(2,i) = 1;

end

if contains(grp(i), "neptune") == 1

targetsMulti(3,i) = 1;

targetsBinary(2,i) = 1;

end

end

% -- NEURAL NETWORKS -- %

x = transpose(x); % Each column needs to be a sample rather than a row

[net, tr] = neuralNetwork(x, targetsMulti); % Multi classification with PCA

[net, tr] = neuralNetwork(x, targetsBinary); %Binary classification with PCA

x = transpose(dm);

[net, tr] = neuralNetwork(x, targetsMulti); % Multi classification without PCA

[net, tr] = neuralNetwork(x, targetsBinary); % Binary classification without PCA

**neuralNetwork.m**

function [net, tr] = neuralNetwork(x,t)

net = patternnet(10, "trainscg", "mse");

net.trainParam.showWindow = false;

net.divideParam.trainRatio = 0.7;

net.divideParam.valRatio = 0.15;

net.divideParam.testRatio = 0.15;

[net, tr] = train(net, x, t);

end