Rerverse Engineer: Lab 3 Report

Jesus Lopez
Dr. Saidur Rahman

September 16, 2024

Contents

Chapter 1

Introduction

Before we jump into the exercises, which are the main component in this lab. I wanted to mention a couple things that are related to machine learning.

1.1 Types of Machine Learning

1.1.1 Supervised Learning

Definition

Defined as when the model gets trained on a labelled dataset. Labelled datasets have bot input and output parameters.

Practical Example

A **practical example** is in malware classification where samples of both malware and benign software are labeled.

Advantages

The Advantages of Supervised learning is that models can have high accuracy as they are trained on labeled data. Another advantage is that it is possible to use pre-trained models, this can save time and resources when developing new models.

disadvantages

Talking about disadvantages, it has limitations in knowing patterns, it might be possible that it will struggle with unseen data or a different pattern that was not presented in the training data.

1.1.2 Semi-Supervised Learning

Definition

It is a machine learning algorithm that works in between supervised and unsupervised learning, because it used labelled and unlabelled data. This machine learning algorithm is used when the dataset is expensive and time consuming, so that labeled data is costly and time consuming.

Practical Example

A practical example is using a small set of labeled phishing emails and a large set of unlabeled ones to improve detection accuracy.

Advantages

One of the advantages is that it can lead to better generalization as compared with supervised learning because it can take both kind of data.

Disadvantages

One of the main disadvantages is that the unlabeled data can impact in the model performance accordingly. also is that it still requires some amount of labeled data.

1.1.3 Unsupervised Learning

Definition

In unsupervised learning, algorithms discovers patters and relationships using unlabeled data. the main difference is that it does not involve providing the algorithm with labeled target outputs. One of the main goals is often to discover patterns, similarities, etc.

Practical Example

A practical example might be anomaly detection for identifying potential security breaches in a network traffic.

Advantages

As mentioned before, one of the advantages is that unsupervised learning can help us in discovering hidden patterns. It also does not require labeled data and reduces the effort of data labeling.

Disadvantages

One of the disadvantages is that without using labels, it may be difficult to predict the quality of the model's output.

1.1.4 Reinforcement Learning

Definition

Is an algorithm where an agent learns to make decisions by interacting with an enviroment, aiming to maximize a cumulative reward over time.

Practical Example

A practica example might be automating firewall policies for network traffic filtering.

Advantages

One of the advantages is that it has autonomous decision-making that is well suited for tasks and it can learn to make a sequence of decisions. This technique is preferred for long-term results rather than short term.

Disadvantages

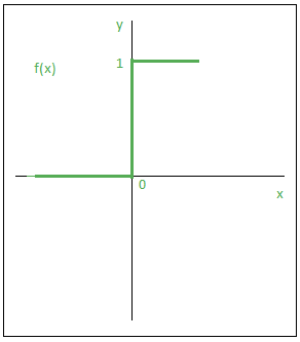
One of the main disadvantages is that reinforcement learning is quite expensive computationally speaking and time-consuming.

1.2 Activation Functions

Another topic I wanted to introduce/explain are activation functions. Well, an activation function is used to determine the output of a neuron based on its input. This ones take a weighted sum of the input features, applies the function and then produces and output that can be passed to the next layer.
Of course, following are ton the complete list but its reasonable to mention some of them

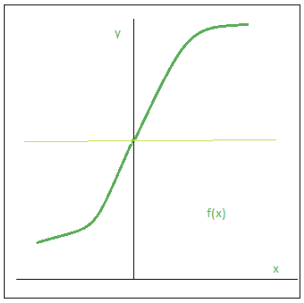
1.2.1 Step function

This is one of the simplest, take as an example a threshold value and if the value of net input says y is greater than the threshold then the neuron is activated. it looks something like this:



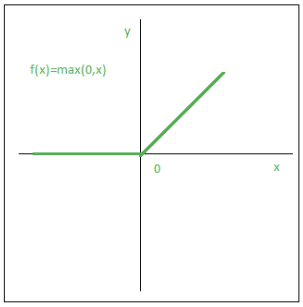
1.2.2 Sigmoid Function

This one is widely used. it is often referred as an smooth function. One of the biggest advantages is that is not linear, is not black or white. it looks something like this:



1.2.3 ReLU

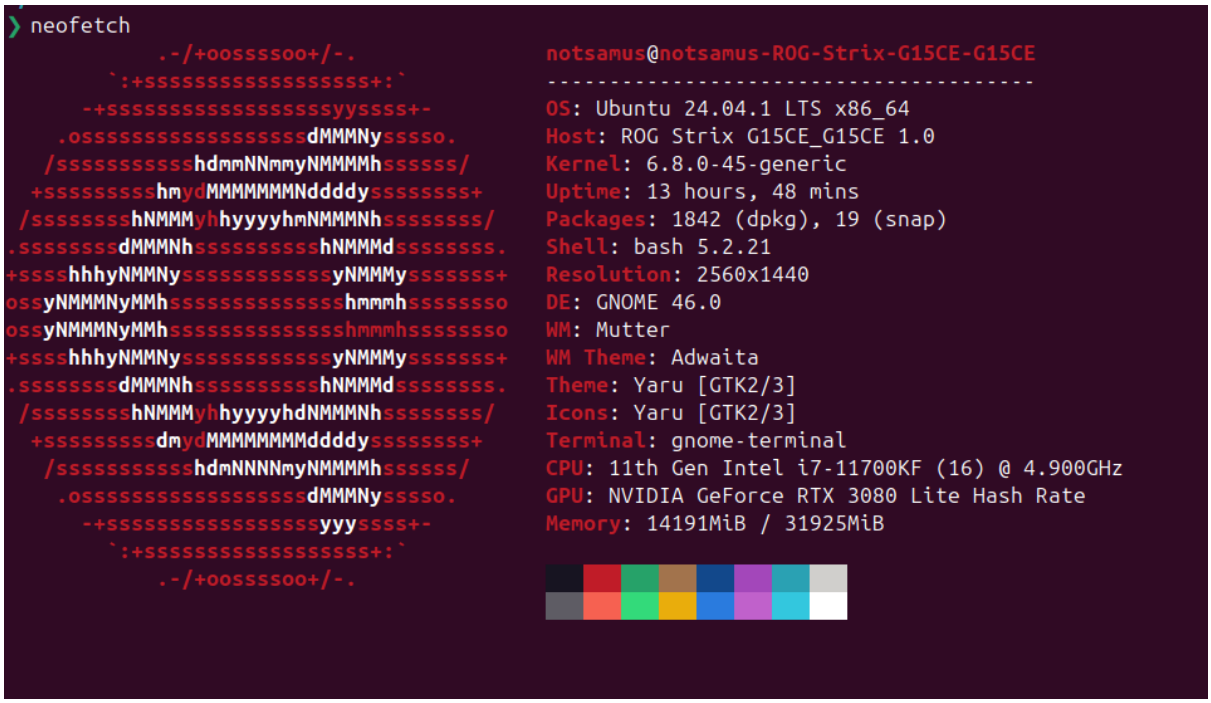
This is the rectified linear unit. it is most widely used activation function and it look like this:



Chapter 2

Getting started

For this lab we are going to be using a ubuntu enviroment. This are the system specifications:



It is necessary to install it from the jupyter web page with:

```
pip install jupyerlab or pip install notebook
```

once installed it is necessary to run it with the following command:

```
jupyter lab or jupyter notebook
```

side note: in ubuntu or pop os, it can be installed from the appstore that is inlcuded within the system.

Chapter 3

Supervised Learning

3.1 Introduction

In this exercise we are going to be using The multi-layer Perceptron with Supervised Learning

3.2 Assignment: Supervised Learning with MLP

3.2.1 Objective:

- Process the given EMBER dataset to include only malware and benign samples.
- Train a MLP for binary classification to differentiate between malware and benign samples.
- Use StandardScaler to standardize the dataset → Link

3.2.2 Steps Taken:

- Preprocess
  - Only keep the samples with malware and benign labels
  - split the dataset into training and testing sets.
- Build an MLP for binary classification
  - Use at least one hidden layer.
  - Choose a suitable activation function (e.g., ReLU, Sigmoid).
- Hyperparameter Tuning:
  - Explore and tune the following Hyperparameters:
    - Learning rate
    - Number of hidden layers
    - Number of neurons in each hidden layer
    - Activation function (e.g., ReLU, Sigmoid)
    - Batch size
    - Optimizer (e.g., Adam, SGD)
    - Epochs

3.2.3 Screenshots

Data Pre-Processing

[1]:
```
import os
import numpy as np
import pandas as pd
import lightgbm as lgb
import matplotlib.pylab as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import roc_auc_score, roc_curve
```

[2]:
```
data_dir = './preprocessed_cleaned_and_scaled/'
```

[3]:
```
ls -l ./preprocessed_cleaned_and_scaled/
```

```
total 1025980
-rw-rw-r-- 1 notsamus notsamus 413663673 Sep 26 18:18 XY_test.npz
-rw-rw-r-- 1 notsamus notsamus 636924587 Sep 26 18:19 XY_train.npz
```

[4]:
```
# Load the compressed test .npz file
test_data = np.load(data_dir + 'XY_test.npz')

# Load the test data
X_test = test_data['X_test']
y_test = test_data['y_test']
print("Test arrays loaded successfully.")
```

```
Test arrays loaded successfully.
```

[5]:
```
smalltestX, smalltestY = X_test[:50000], y_test[:50000]
```

[6]:
```
small_X_train, small_X_test, small_y_train, small_y_test = train_test_split(
    X_test, y_test, test_size=0.1, random_state=42)
```

[7]:
```
small_y_train.shape, small_y_test.shape
```

[7]:
```
((180000,), (20000,))
```

1.1 Data Preprocessing

Build an MLP model

Installing Tensorflow

[10]:
```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score, roc_curve
import tensorflow as tf
from tensorflow.keras.initializers import HeNormal
from sklearn.preprocessing import StandardScaler

# Verify GPU availability
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
if len(tf.config.list_physical_devices('GPU')) > 0:
    print("Using GPU for training!")
else:
    print("GPU not available, using CPU.")
```

```
Num GPUs Available:  1
Using GPU for training!
```

[13]:
```
# Normalize the input data to prevent exploding gradients
scaler = StandardScaler()
small_X_train = scaler.fit_transform(small_X_train)
small_X_test = scaler.transform(small_X_test)

# Step 1: Create the MLP Model with deeper architecture
model = Sequential()

# First hidden layer (1024 units)
model.add(Dense(1024, input_dim=small_X_train.shape[1], activation='relu', kernel_initializer=HeNormal()))

# Second hidden layer (512 units)
model.add(Dense(512, activation='relu', kernel_initializer=HeNormal()))

# Third hidden layer (256 units)
model.add(Dense(256, activation='relu', kernel_initializer=HeNormal()))

# Fourth hidden layer (128 units)
model.add(Dense(128, activation='relu', kernel_initializer=HeNormal()))

# Output layer for binary classification
model.add(Dense(1, activation='sigmoid'))

# Step 2: Compile the model with Adam optimizer and a lower learning rate
adam = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])
```

```
/home/notsamus/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `inpu
t_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

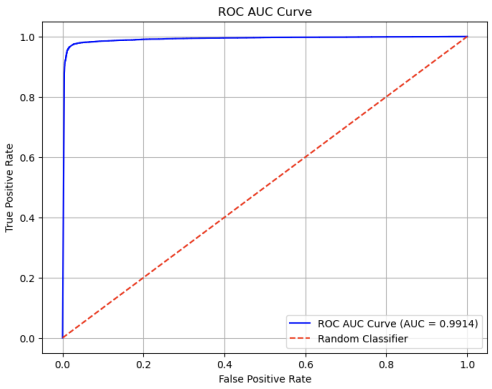
[14]:
```
model.fit(small_X_train, small_y_train, epochs=20, batch_size=256, verbose=1)
```

```
Epoch 1/20
704/704 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9132 - loss: 0.2100
```

1.2 Build an MLP model

```
[19]: import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score, roc_curve

# Step 1: Calculate the ROC AUC score
roc_auc = roc_auc_score(small_y_test, y_pred_prob)
print(f"ROC AUC Score: {roc_auc:.4f}")

# Step 2: Plot the ROC AUC Curve
fpr, tpr, thresholds = roc_curve(small_y_test, y_pred_prob)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC AUC Curve (AUC = {roc_auc:.4f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Classifier')

# Adding labels and title
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC AUC Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

```
ROC AUC Score: 0.9914
```



1.3 ROC table

3.2.4 Analysis

For this lab we are going to be using the data pre-processed, I tried using the data and put it on the GPU. The problem is that it was requiring 7.10 GB of Vram and for some reason. I Proceeded to use the pre-processed data that was given by the professor, on picture **1.1** its the section of data pre-processing. Almost everything was running smoothly, when using TensorFlow we have to download the CUDA version also. It can be installed in the following way:

```
pip install tensorflow[and-cuda]

```

to test if it runs it is simple a manner of:

```
import tensorflow as tf
print('using GPU') if len(tf.config.list_physical_devices('GPU'))>0 else print(
    'using CPU')

```

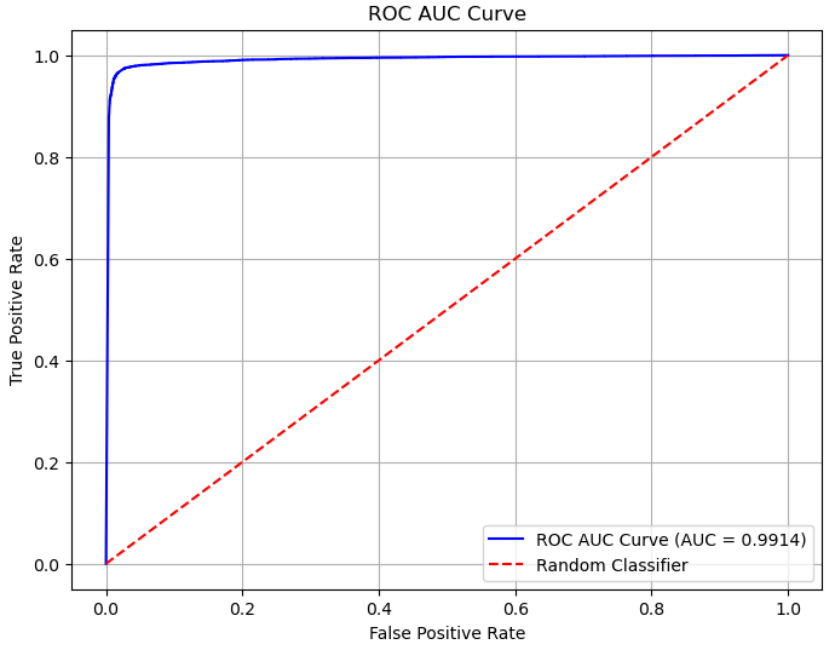
In this case we get that:

```
[2]: import tensorflow as tf

[4]: print('using GPU') if len(tf.config.list_physical_devices('GPU'))

usingGPU
```

After we validated the use of GPU (which is the best way to run it). We are going to continue as seen on picture **1.2** is the section Build an MLP model which stands for *Multi-layer Perceptron*. After that we are to do model fit and in picture **1.3** we are going to plot with the help of matplotlib the ROC table. The **Receiver Operating Characteristic** curve is created by plotting two parameters, which stands for **True Positive Rate** (TPR) and for **False Positive Rate** (FPR).

I think that the AUC is the measure of how well the model distinguishes between the two classes:

- when **AUC = 1.0** Perfect.
- when **AUC = 0.5** it performs no better thatn random chance.
- when **AUC < 0.5** is worse than random.



In this case we have that the ROC AUC curve is located in the top-left corner and it indicates that is a better performance of the model. The diagonal line that is dotted, its the random classifier.

Chapter 4

K-Means Clustering for Binary Classification

4.1 Objective

Use the K-Means algorithm for binary classification on the EMBER dataset (malware vs benign samples).

4.2 Steps Taken

- Pre-process the ember dataset
- Apply K-Means with k = 2 to perform clustering on the pre-processed dataset.
- Map clusters to binary classes (malware or benign):
  - Cluster 1 can be mapped to malware (Class 0).
  - Cluster 2 can be mapped to benign (Class 1).
- Evaluate your model:
  - Calculate accuracy, precision, recall, F1-score, and plot the confusion matrix.
  - Explain how well the clustering aligns with the actual classes.

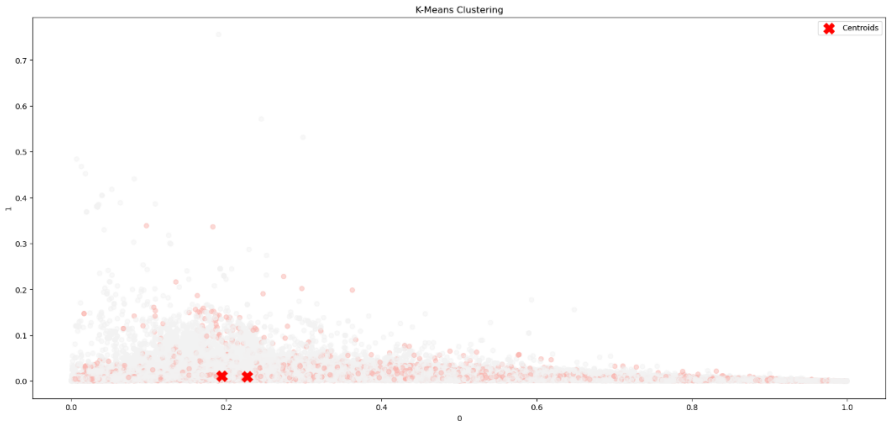
4.3 Analysis

I tried creating the K-means I followed a guide online but i could not made it work properly. I included the jupyter notebook on the submission of this assignment to seek for feedback and what it went wrong. This is the code that was used fof the k-means:

[36]:
```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score, precision_recall_fscore_support
import seaborn as sns
```

•[43]:
```
data = np.load(data_dir + 'XY_test.npz')
X = data['X_test']

X = pd.DataFrame(X)

kmeans = KMeans(n_clusters=2, random_state=0)
clusters = kmeans.fit_predict(X)

X['Cluster'] = clusters
plt.figure(figsize=(20, 9))
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=X['Cluster'], cmap='Pastel1', alpha=0.5)

# Plotting the cluster centers
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='X', s=200, label='Centroids')

plt.title('K-Means Clustering')
plt.xlabel('0')  # Replace with actual feature name if needed
plt.ylabel('1')  # Replace with actual feature name if needed
plt.legend()
plt.show()
```

This is the graph:



These are the results:

0]:
```
mapped_labels = [1 if cluster == 1 else 0 for cluster in clusters]

# Evaluate the Model
accuracy = accuracy_score(y, mapped_labels)
precision, recall, f1, _ = precision_recall_fscore_support(y, mapped_labels, averag

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
Accuracy: 0.54507
Precision: 0.5262648748819916
Recall: 0.90306
F1 Score: 0.6649975331188006
```

Chapter 5

Reflections

5.1 Problems we encountered

One problem I encountered is that when installing jupyter notebooks with the instructions on the documentation, it would not work. I tried aroud five times and it would not write the jupyter command in the path. Then I switched to anaconda and most problems solved.

Another problem I encountered is when tried to download the files from the google drive, it will downloaded in separate parts and when extracting it wont extract correctly.

In the end I had to switch to a Linux distribution, I used Ubuntu and installed PyTorch and TensorFlow. with PyTorch when i was going to train the data, i got the problem that it was requiring 7.10 GB but for some reason the GPU was now willing to do that.

after solving some of the problems the other thing I have issues with was that I could not make the kmeans work propperly, the accuracy on my model was too low and the result, the graph was horribly

5.2 Conclusion

I like the idea of using machine learning for some security aspects, I think this kind of topics are a great addition to our learning, i had problems running the k-means but it does not mean that I got a horrible experience. of course i learned the differences between the different kinds of machine learning algorithms. i would like to experience AI as an alternative of my computer science degree. This is a side comment in my discussion of the lab, but I believe some honeypots implement unsupervised learning to learn patterns and detect intruders. i would be interesing exporing something in that area.

Chapter 6

References

I am not an expert in Machine learning and I think it is sufficient to put the page where I got most of the definitions that I used on this document

GeekforGeeks Types of Machine Learning and GeeksforGeeks Activation Functions

Also I want to thank the teacher for helping us to build the models.