Domain Generation Algorithm Classification

Machine Learn (13016364)

Noppasit Kasikijvorakul (61090016)

30 / 11 / 2020

## 1. Introduction

For the last decades, the raise of internet has become significant and gradually throughout the whole world. IP address is required for connecting to these networks; however, the address is difficult to remember. Therefore, people use domain name system (DNS) to map from IP addresses to hostnames instead. It also provides operations for various web-based applications, email, and distributed systems. Various cyber-attacks utilize DNS due to its ability to penetrate firewall.

Recently, malwares such as ransomware and botnets have caused tons of the damage. The infected receives commands from remote command and control (C&C) server to install additional malwares. These infected machines are used for send spam messages, steal personal data, and launch distributed denial of service attacks (DDoS). In order to receive commands C&C server, the malicious code must have IP address or domain address of the C&C server. If the IP address or domain is blocked, it can prevent the infected machines from connecting to the C&C server. Hence, the attacker employs domain generation algorithm (DGA) to avoid blocking techniques.

The motivation of this training is to classify which domains are legit and which are malicious. The model could be used to detect and block DGA domains. In this dataset, there are three classes of DGA algorithm: goz, Cryptolocker, and newgoz.

## 2. Background

Support Vector Machine (SVM) is a supervised machine learning algorithm. It can be employed for classification or regression. SVM is based on finding the flat boundary called hyperplane that is best for diving a dataset into classes. It is responsible for finding the boundary and maximizing the margin. SVM learning combines aspects of both the instance-based nearest neighbor learning and the linear regression modeling.

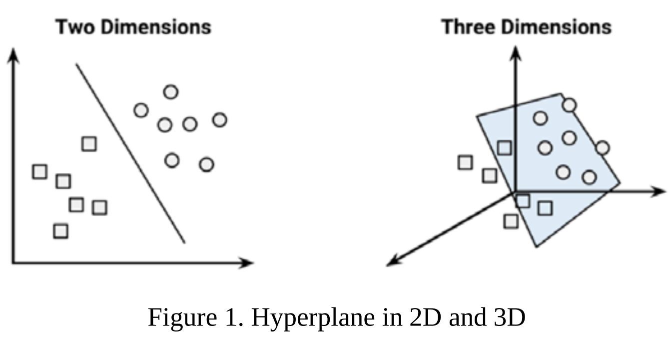


Figure 1. Hyperplane in 2D and 3D

To find the right hyperplane, the support vectors, points that have the greatest separation between two class, are used. For the linearly separable data, find the shortest line between two convex hulls and selects these points as support vectors as shown in Figure 2. For nonlinearly separable data, there are two concepts: soft margin and kernel trick. Soft margin is to find to separate, but tolerate some misclassified. The penalty term – “Cost” or “C” is how much penalty SVM gets when it makes misclassification. The bigger the C, the more penalty. Kernel trick is to utilizes existing features, applies some transformations, and creates new features. Examples of kernel are “linear”, “polynomial”, “radial basis”, “sigmoid”, and “precomputed”. The two most popular kernels are polynomial and radial basis.

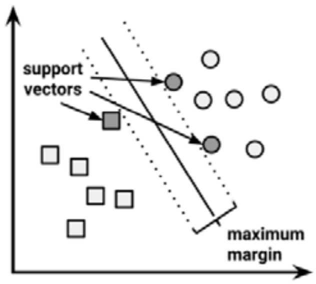


Figure 2. Linear separable case

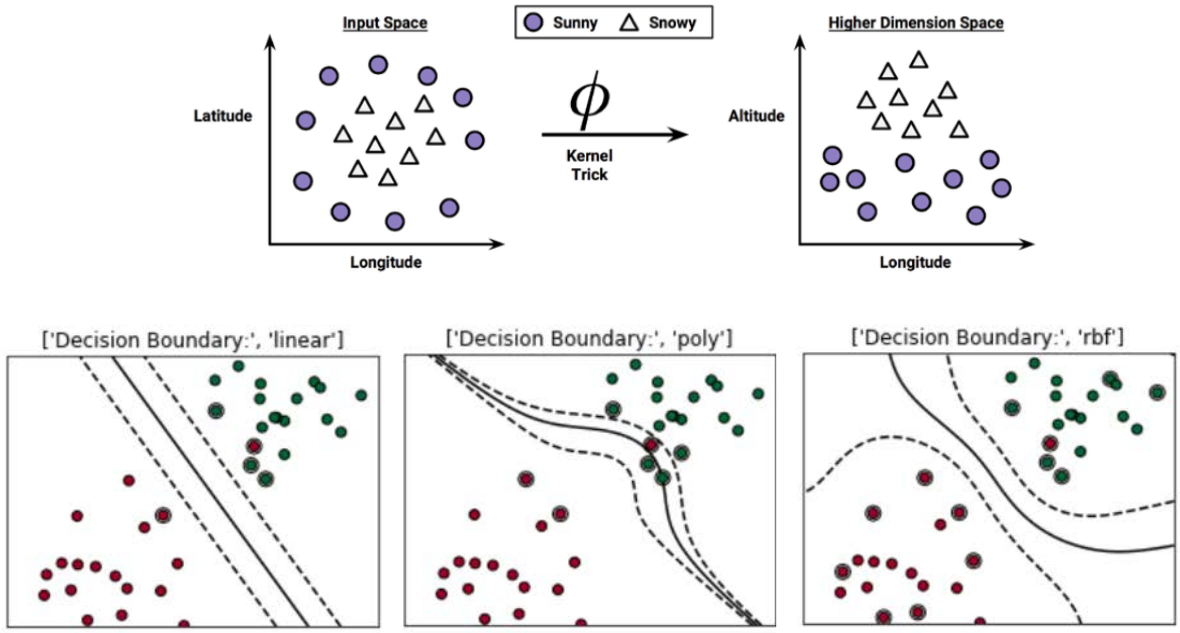


Figure 3. Non-linear separable case

## 3. Methodology

The architecture of the model that is used for classification of domain names in this research has four steps. First, the input, a string of hostname, is separated into a set of characters. Second, all of the characters transform into five features: a) length of the domain name, b) domain name entropy, c) vowel's ratio in a domain name, d) consecutive consonants' ratio, e) proportion of the digits. Third, these features are used in the Support Vector Machine (SVM) using linear and radial basis kernels. Last, the determination whether the domain name is legit or malignant and which subclass does it belong is returned as the output.

## 4. Experiment

### 4.1. Collecting Data

The dataset is a csv file of domains and classification of DGA and legit. In the file there are four columns: host, domain, class, and subclass. The subclass of are also included, which are either legit, cryptolocker, goz, and newgoz. However, only domain and subclass were used in this experiment. There are 133926 datasets; 93748 were used for training set (70%) and 40178 were used for testing set (30%). It was provided by Data Driven Security and can be downloaded from https://datadrivensecurity.info/blog/pages/dds-dataset-collection.html.

4.2. Exploring and Preparing Data

Domain and subclass were chosen from the 4 columns. Then, the subclass is changed into factor class. The four levels are 1 for “cryptolocker”, 2 for “goz”, 3 for “legit”, 4 for “newgoz”. The domain names are extracted to five features: length, entropy, vowel’s ratio, consecutive constants’ ratio, and proportion of digits. The relationship between features can be visualized in Figure 4. The color is represented according to the levels respectively such as blue for 1, green for 2, red for 3. It is illustrated in Figure 1 that blue and red are similar in most features, while others are noticeably differ.

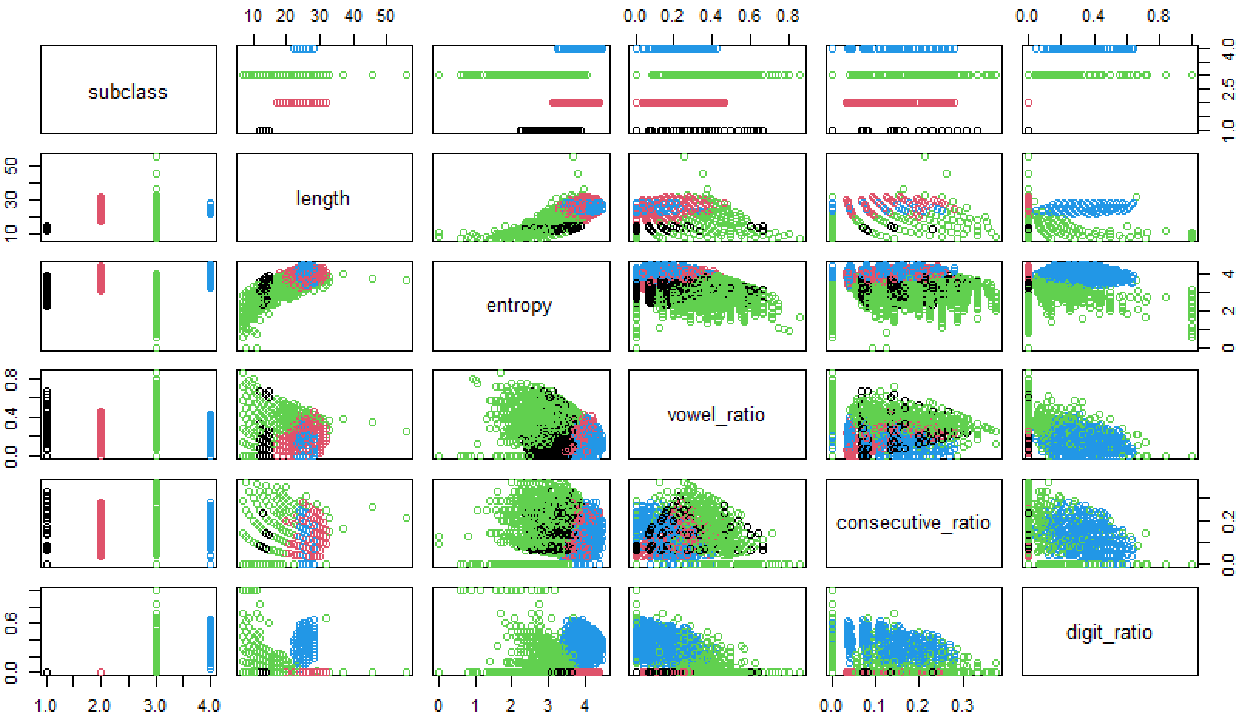


Figure 4. Relationship among 5 features

4.3. Training a Model

The method that is chosen for employment is Support Vector Machine (SVM). Library “e1071” was selected for training the model. There are two kernels that was applied when training, which are linear kernel and radial-based kernel.

4.4. Evaluating Model Performance

As shown in Figure 5 and Figure 6, radial basis kernel is more superior than linear kernel by two percent at 94% accuracy. Linear kernel predicts more wrong results in all classification, especially cryptolocker and goz. Most of the misclassification are misinterpreted of cryptolocker and legit.

```
                 Reference
Prediction     cryptolocker   goz legit newgoz
  cryptolocker         8087     0   723      0
  goz                     0  2213   284      0
  legit                2190    11 23377      2
  newgoz                  0     0     2   3289

Overall Statistics

               Accuracy : 0.9201
                 95% CI : (0.9174, 0.9227)
    No Information Rate : 0.6069
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8539
```

Figure 5. Confusion matrix of prediction using linear kernel

```
                 Reference
Prediction     cryptolocker   goz legit newgoz
  cryptolocker         8729     0   746      0
  goz                     0  2180    33      0
  legit                1548    44 23607      1
  newgoz                  0     0     0   3290

Overall Statistics

               Accuracy : 0.941
                 95% CI : (0.9386, 0.9432)
    No Information Rate : 0.6069
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8925
```

Figure 6. Confusion matrix of prediction using radial basis kernel

4.5 Improving Model Performance

The model can be improved by many techniques, such as changing kernel, finding the best SVM kernel parameters (C, $\gamma$) using Cross-Validation (CV), scaling the data, and bias shift for less false positive and false negative. In addition, changing training model to CNN or even combining models might improve the efficiency. In this testing, CV and scaling were used. In Figure 5 and Figure 7, it can be seen that the accuracy has improved slightly if the data has been scaling. Similarly, the accuracy of the scaled data and after CV has better performance by almost one percent when comparing between Figure 6 and Figure 8.

```
                 Reference
Prediction     cryptolocker   goz legit newgoz
  cryptolocker         8106     8   868      0
  goz                     0  2179    37      0
  legit                2171    37 23477      1
  newgoz                  0     0     4   3290

Overall Statistics

               Accuracy : 0.9222
                 95% CI : (0.9195, 0.9248)
    No Information Rate : 0.6069
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8572
```

Figure 7. Prediction of scaled data using linear kernel

```
                   Reference
Prediction         cryptolocker   goz legit newgoz
  cryptolocker             8756     0   734      0
  goz                         0  2180    35      0
  legit                    1521    44 23617      0
  newgoz                      0     0     0   3291

Overall Statistics

                     Accuracy : 0.9419
                       95% CI : (0.9396, 0.9442)
          No Information Rate : 0.6069
          P-Value [Acc > NIR] : < 2.2e-16

                        Kappa : 0.8943
```

Figure 8. Prediction of scaled data using radial basis kernel and optimal parameters (C = 8, $\gamma = 0.5$)

## 5. Conclusion

The upsurge of the cyber attacks has been critical and extensive. Numerous attackers operate the infected from C&C server via malicious codes. They avoid blocking techniques using DGA to conceal the C&C server. Based on the experiment, the model has accuracy of 94.19% for detecting and classifying DGA or legit. For further improving the model, some of the solutions are employing bias shift to reduce the false negative and trying Convolutional Neural Network (CNN) and bidirectional Long Short-Term Memory (BiLSTM).

## References


Hsu, C., Chang, C., & Lin, C. (2016). *A Practical Guide to Support Vector Classification*. Csie.ntu.edu.tw. Retrieved 30 November 2020, from https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Ren, F., Jiang, Z., Wang, X., & Liu, J. (2020). *A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network*. Electronics | Free Full-Text | Effective DGA-Domain Detection and Classification with TextCNN and Additional Features | HTML. Retrieved 30 November 2020, from https://link.springer.com/article/10.1186/s42400-020-00046-6.

Moraes, D., Wainer, J., & Rocha, A. (2016). *Low False Positive Learning withSupport Vector Machines*. (PDF) Low False Positive Learning withSupport Vector Machines. Retrieved 30 November 2020, from https://www.researchgate.net/publication/298427125_Low_False_Positive_Learning_with_Support_Vector_Machines.

Chen, L. (2019). *Support Vector Machine — Simply Explained*. Support Vector Machine — Simply Explained | by Lujing Chen | Towards Data Science. Retrieved 30 November 2020, from https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496.

R code

```r
library(e1071) #package for SVM
library(acss) #package for get string entropy
library(caret) #for visualizing Confusion Matrix

#Import and Preparing data
domains <- read.csv("C://Users/USER/Desktop/IC SE/3-1/Machine Learning/ML project/legit-dga_domains.csv",
                    stringsAsFactors = FALSE, fileEncoding="UTF-8-BOM")
#Use only domain and subclass
domains <- domains[,c(2,4)]
#Convert subclass into factor
#1 = cryptolocker, 2 = dga, 3 = legit, 4 = newdga
domains$subclass <- factor(domains$subclass)

#Features
#a) domain name length
domains_length <- data.frame(length=apply(domains,2,nchar)[,1])

#b) domain name entropy
domains_entropy <- data.frame(entropy=entropy(domains[,1]))


#c) vowel's ratio in domain name
vowel_ratio <- function(x) {
  nchar(gsub("[^aeiou]","",x, ignore.case = TRUE))/nchar(x)
}
domains_vowel <- data.frame(vowel_ratio=vowel_ratio(domains[,1]))

#d) consectutive consonants' ratio
domains_consecutive_constants <- lapply(gregexpr("[^aeiou0-9][^aeiou0-9]+",domains[,1],ignore.case = TRUE),
                                        function(x) length(x[x > 0]))
domains_consecutive_ratio <- data.frame(consecutive_ratio=unlist(domains_consecutive_constants))
domains_consecutive_ratio <- domains_consecutive_ratio/domains_length

#e) proportion of the digits
digit_ratio <- function(x) {
  nchar(gsub("[^0-9]","",x, ignore.case = TRUE))/nchar(x)
}
domains_digit <- data.frame(digit_ratio=digit_ratio(domains[,1]))

# Merge all features into one data frame
data <- cbind(domains_length,domains_entropy,domains_vowel,domains_consecutive_ratio,
              domains_digit,domains["subclass"])

#separate training and test set
train_amount <- sample.int(nrow(data), size = nrow(data)*0.7)
domains_train <- data[train_amount,]
domains_test <- data[-train_amount,]

#Visualization data with Scatterplot
pairs(subclass ~ ., data = domains_train, col = domains_train$subclass)

#Linear SVM
model <- svm(subclass ~ .,data = domains_train, kernel = "linear", probability = TRUE)
domains_pred <- predict(model,domains_test)

confusionMatrix(domains_pred, domains_test$subclass)
```

```r

#Radial SVM
model2 <- svm(subclass ~ .,data = domains_train, kernel = "radial", cost = 1, gamma = 0.5, probability = TRUE)
domains_pred2 <- predict(model2,domains_test)

confusionMatrix(domains_pred2, domains_test$subclass)

#Tuning using grid search and cross validation
svmTune <- tune(svm, subclass ~ ., data = domains_train, ranges = list(cost = 10^(0:4), gamma = 10^(-6:3)))
summary(svmTune) #cost = 100, gamma = 0.1

#Radial SVM after Cost-Validataion
model3 <- svm(subclass ~ .,data = domains_train, kernel = "radial", cost = 100, gamma = .01, probability = TRUE)
domains_pred3 <- predict(model3,domains_test)

confusionMatrix(domains_pred3, domains_test$subclass)

#scale the data
domains_scale_train <- as.data.frame(sapply(domains_train[,-6], function(x) if(is.numeric(x)) scale(x) else x))
domains_scale_train <- cbind(domains_scale_train,subclass = domains_train[,6])
domains_scale_test <- as.data.frame(sapply(domains_test[,-6], function(x) if(is.numeric(x)) scale(x) else x))
domains_scale_test <- cbind(domains_scale_test,subclass = domains_test[,6])

#Scaled Linear SVM
model_scale <- svm(subclass ~ .,data = domains_scale_train, kernel = "linear", probability = TRUE)
domains_scale_pred <- predict(model_scale,domains_scale_test)

confusionMatrix(domains_scale_pred,domains_scale_test$subclass)

#Scaled Radial SVM (default)
model2_scale <- svm(subclass ~ .,data = domains_scale_train, kernel = "radial",
                    cost = 1, gamma = 0.5, probability = TRUE)
domains_scale_pred2 <- predict(model2_scale,domains_scale_test)

confusionMatrix(domains_scale_pred2,domains_scale_test$subclass)

#Cross Validation scale data
svmTune2 <- tune(svm, subclass ~ ., data = domains_scale_train,
                 ranges = list(cost = 2^seq(-3,11,by = 2), gamma = 2^seq(-7,3,by = 2)))
summary(svmTune2)

#Scaled Radial SVM (Cost = 8, gamma = 0.5)
model3_scale <- svm(subclass ~ .,data = domains_scale_train, kernel = "radial",
                    cost = 8, gamma = 0.5, probability = TRUE)
domains_scale_pred3 <- predict(model3_scale,domains_scale_test)

confusionMatrix(domains_scale_pred3,domains_scale_test$subclass)
```