COMP 5107
Final Project: Building a Pattern Recognition System on the NSL-KDD Dataset
Yasaman Shahrasbi

Table of Contents

1. **Introduction** 1
2. **Dataset** 1
   2.1 Preprocessing 1
3. Find Mean and Covariance 2
   3.1. Maximum Likelihood 2
   3.2. Bayesian 3
   3.3. Parzen Window 3
4. Classifiers 4
   4.1. Quadratic 4
   4.2. K-Nearest Neighbor 5
   4.3. Ho-Kashyap 5
5. Training and Testing 6
6. Final Notes 6

1. Introduction

This project aims to compare the performance (accuracy measure) of three different classifiers, namely Quadratic, K-Nearest Neighbors, and Ho-Kashyap, in a binary classification problem. According to the results, Quadratic and Ho-Kashyap algorithms work the best in favor of both classes. In the following sections, we will introduce the dataset, methods to obtain mean and covariance for each class, each of the classifiers, a brief explanation of the classifiers, the results, and a discussion are provided.

2. Dataset

The NSL-KDD dataset from the Canadian Institute for Cybersecurity website is used for this project. This dataset has 42 features, which are the characteristics of internet traffic captured by an intrusion detection network. The labels of this dataset show whether traffic is normal or an attack.

2.1. Preprocessing

Two KDDTrain+.TXT and KDDTest+.TXT files are first merged and shuffled. As it is required for this project to choose six features for the experiments, using the Sklearn's Extra Tree classifier, the six most important features which are "dst_host_count", "srv_count", "dst_host_srv_count", "dst_bytes", "src_bytes", and "count" are chosen. As the next step of preprocessing, the data is normalized between the range of (-2, 2) using the min-max scaler. Also, as shown in figure 0, this dataset is balanced, meaning that the number of instances from both classes are nearly the same; However, in order to speed up the process of training and testing, 4000 instances from each class are chosen.

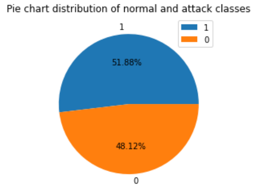

fig0

In figures 1, 2, and 3, the points along domains 0-1, 2-5, and 3-5 are shown before diagonalization, and figures 4, 5, and 6 show the points after diagonalization.

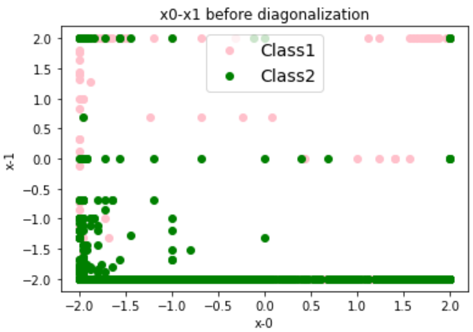

Fig1

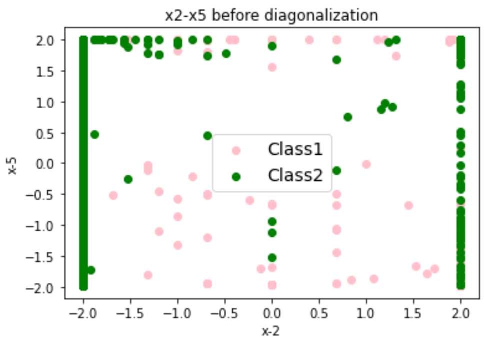

Fig2

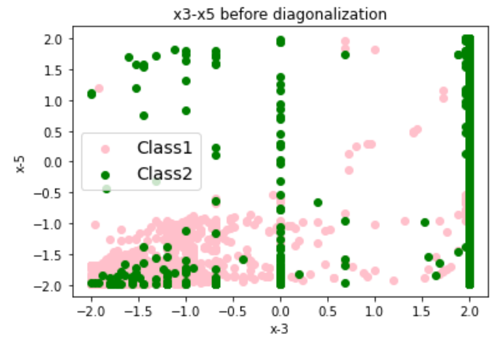

Fig3

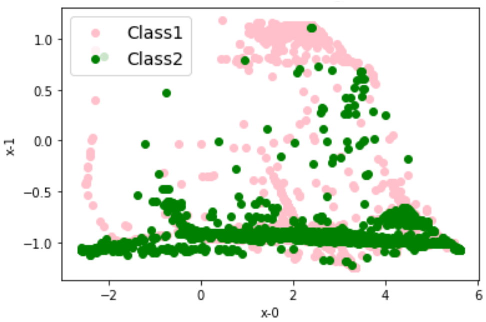

Fig4

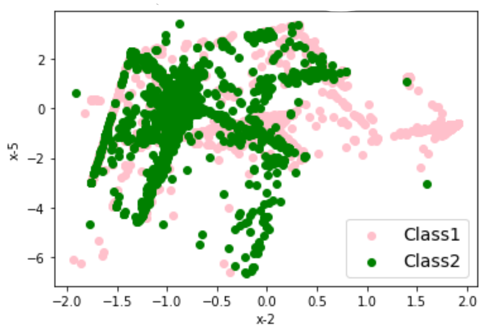

Fig5

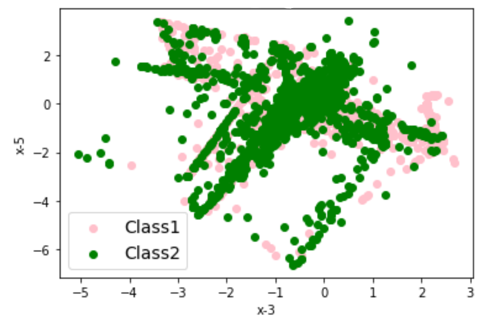

Fig6

---

3. Find Mean and Covariance

3.1. Maximum Likelihood

Mean and covariance calculated with maximum likelihood method is used for diagonalization, and the covariance is used for calculating Bayesian mean.

$$ml\_mean = \frac{1}{n}\sum_{i=0}^{n-1} x_i \qquad ml\_covariance = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - ml\_mean)\,(x_i - ml\_mean)^T$$

Mean of class 0: $\begin{bmatrix} -1.02794 \\ -1.27896 \\ -0.59696 \\ -1.09716 \\ -1.39633725 \\ 0.06876 \end{bmatrix}$ Mean of class 1: $\begin{bmatrix} -1.84089 \\ -1.51986 \\ 1.87459 \\ 1.26597 \\ 1.03349412 \\ -1.95541 \end{bmatrix}$

Covariance of class 0:

$$\begin{bmatrix} 2.849 & -0.128 & -0.288 & -0.622 & -0.347 & -1.938 \\ -0.128 & 2.099 & 1.537 & 1.212 & 0.436 & -1.417 \\ -0.288 & 1.537 & 2.793 & 1.804 & 1.057 & -1.841 \\ -0.622 & 1.212 & 1.804 & 1.996 & 1.123 & -1.277 \\ -0.347 & 0.436 & 1.0577 & 1.123 & 1.055 & -0.717 \\ -1.938 & -1.417 & -1.841 & -1.277 & -0.717 & 3.865 \end{bmatrix}$$

Covariance of class 1:

$$\begin{bmatrix} 0.531 & 0.023 & 0.011 & 0.066 & -0.027 & -0.005 \\ 0.023 & 1.027 & -0.103 & -0.342 & -0.513 & -0.006 \\ 0.011 & -0.103 & 0.348 & 0.288 & 0.279 & -0.007 \\ 0.066 & -0.342 & 0.288 & 1.643 & 1.583 & -0.090 \\ -0.027 & -0.513 & 0.279 & 1.583 & 2.030 & -0.083 \\ -0.005 & -0.006 & -0.007 & -0.090 & -0.083 & 0.112 \end{bmatrix}$$

3.2. Bayesian

$$\boldsymbol{covariance_0} = \boldsymbol{identity_6}$$

$$\boldsymbol{mean_0} = [1\ 1\ 1\ 0\ 1\ 1]$$

$$bl\_mean = \frac{1}{n}covariance\left[\frac{1}{n}covariance + covariance_0\right]^{-1} mean_0 \ + \ covariance_0\left[\frac{1}{n}covariance + covariance_0\right]^{-1}\left(\frac{1}{n}\sum_{j=0}^{n-1} x_j\right)$$

Mean of class 0: $\begin{bmatrix} -1.028 \\ -1.279 \\ -0.597 \\ -1.098 \\ -1.396 \\ 0.070 \end{bmatrix}$ Mean of class 1: $\begin{bmatrix} -1.840 \\ -1.520 \\ 1.875 \\ 1.267 \\ 1.035 \\ -1.95 \end{bmatrix}$

3.3. Parzen Window

In this method, a Gaussian kernel function is put around all the points when each new point x_i arrives, and the contribution of all the points are calculated for the new point.

Mean of class 0: $\begin{bmatrix} -1.027e-03 \\ -1.278e-03 \\ -5.969e-04 \\ -1.097e-03 \\ -1.396e-03 \\ 6.875e-05 \end{bmatrix}$ Mean of class 1: $\begin{bmatrix} -0.001 \\ -0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ -0.001 \end{bmatrix}$

Covariance of class 0: diagonal matrix of $\begin{bmatrix} 0.003 \\ 0.0023 \\ 0.003 \\ 0.002 \\ 0.001 \\ 0.004 \end{bmatrix}$ Covariance of class 1: diagonal matrix of $\begin{bmatrix} 0.0007 \\ 0.0012 \\ 0.0005 \\ 0.0018 \\ 0.002 \\ 0.0003 \end{bmatrix}$

In figures 7 and 8, the final learned distribution along the 3rd and 6th dimensions for class 0, and in figures 9 and 10, for class 1, are shown.

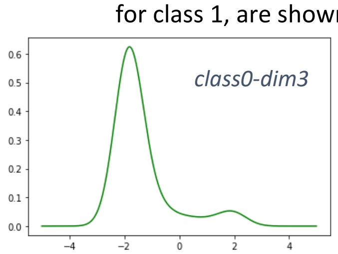

fig7

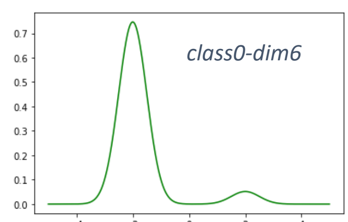

Fig8

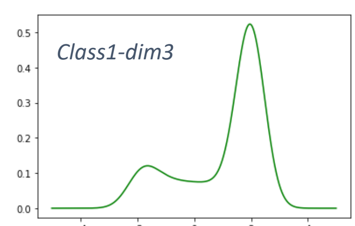

Fig9

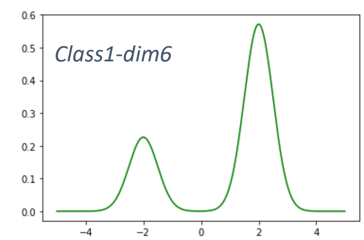

Fig10

---

4. Classifiers

In this project, three different classifiers are implemented. (Quadratic, K-Nearest Neighbor, and Ko-Hashyap)

4.1. Quadratic Discriminant Function

$$X^T AX + B^T X + C \lesseqgtr 0$$

$$A = \frac{covariance_1{}^{-1} - covariance_0{}^{-1}}{2}$$

$$B = -mean_1{}^T covariance_1{}^{-1} + mean_0{}^T covariance_0{}^{-1}$$

$$C = \frac{1}{2}Ln\frac{|covariance_1|}{|covariance_0|} + Ln\frac{P_0}{P_1} + \frac{mean_1^T covariance_1^{-1} mean_1 - mean_0^T covariance_0^{-1} mean_0}{2}$$

Figures 11, 12, and 13 show the discriminant function along domains 0-1, 2-5, and 3-5, before diagonalization, when using Maximum likelihood method to calculate the mean and covariance of the instances, and figures 14, 15, and 16 correspond to after diagonalization.

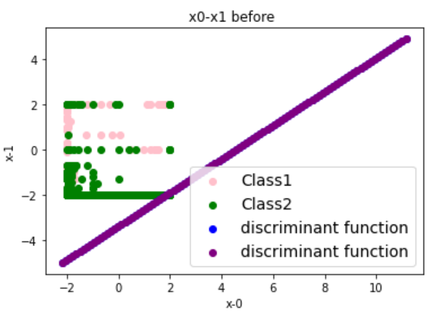

Fig 11

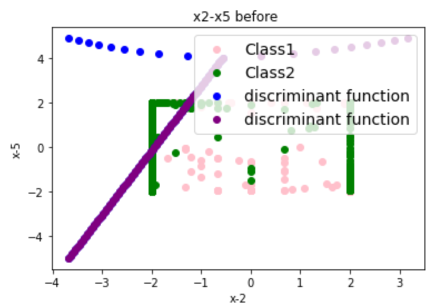

Fig 12

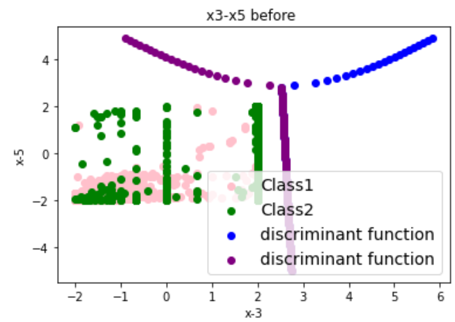

Fig 13

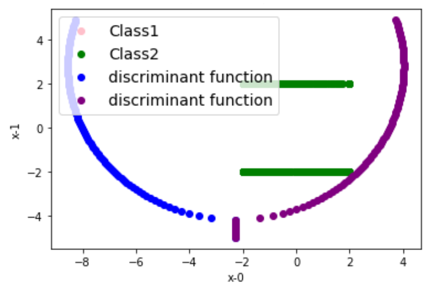

Fig 14

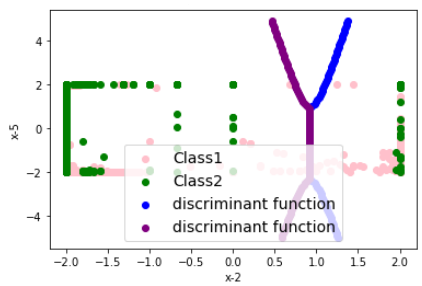

Fig 15

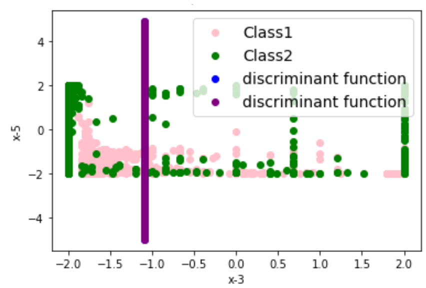

Fig 16

Figures 17, 18, and 19 show the discriminant function along domains 0-1, 2-5, and 3-5, before diagonalization, when using Bayesian method to calculate the mean and covariance of the instances, and figures 20, 21, and 22 correspond to after diagonalization

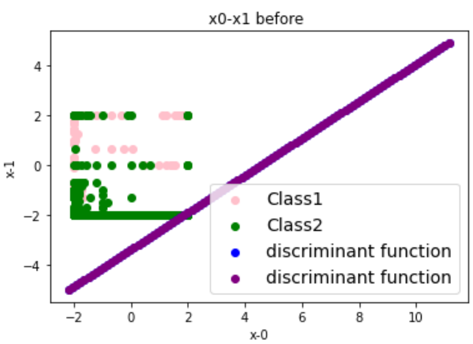

Fig 17

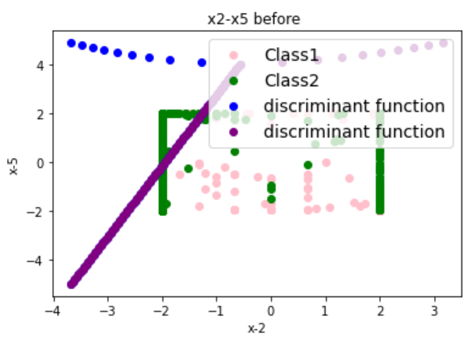

Fig 18

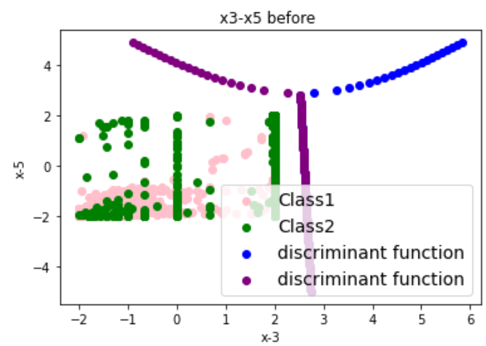

Fig 19

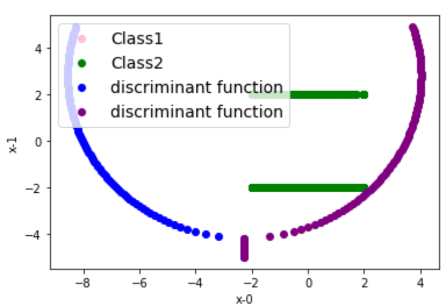

Fig 20

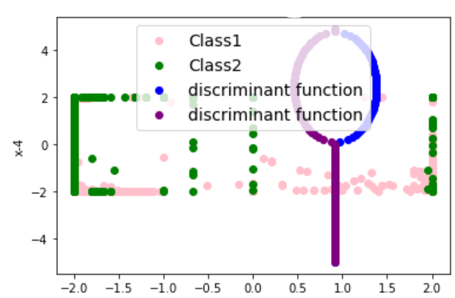

Fig 21

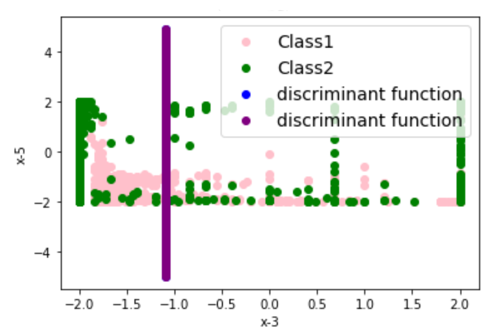

Fig 22

4.2. K-Nearest Neighbor

Pseudo code for KNN algorithm

```
for i in range(len(test_set_0)):
distance = infinity
label = -1
    for j in range(len(train_set_0)):
        d_0 = |test_set_0[i] – train_set_0[j]|
        d_1 = |test_set_0[i] – train_set_1[j]|
        if d_0 < d_1 and d_0 < distance:
            distance = d_0
            label = train_y_0[j]
        else if d_1 < d_0 and d_1 < distance:
            distance = d_1
            label = train_y_1[j]
    if label == 0 and class_test_0[i] == 0:
        tp += 1
    else if label == 1 and class_test_0[i] == 0:
        fp += 1

for i in range(len(test_set_1)):
distance = infinity
label = -1
    for j in range(len(train_set_0)):
        d_0 = |test_set_1[i] – train_set_0[j]|
        d_1 = |test_set_1[i] – train_set_1[j]|
        if d_0 < d_1 and d_0 < distance:
            distance = d_0
            label = train_y_0[j]
        else if d_1 < d_0 and d_1 < distance:
            distance = d_1
            label = train_y_1[j]
    if label == 0 and class_test_1[i] == 0:
        fn += 1
    else if label == 1 and class_test_1[i] == 0:
        tn += 1
```

4.3. Ho-Kashyap

Pseudo code for Ho-Kashyap algorithm (*source*)

```
m = number of instances, 6 = number of features
learning_rate = 0.8
k = 0
number_of_iterations = 100
X_train = \begin{bmatrix} 1 & a_{00} \cdots & a_{05} \\ \vdots & \ddots & \vdots \\ -1 & a_{m0} \cdots & a_{m5} \end{bmatrix}
a = array of ones. size = 6
b = array of ones. size = m
while k < number_of_iterations:
    e = X_train * a – b
    b = b + learning_rate [e + |e|]
    a = (X_train^T X_train)^{-1} X_train^T b
    k += 1
results = X_test * a
for i in results[0: len(X_test_0)]:
    if i > 0:
        tp += 1
    else if i < 0:
        fp += 1
for i in results[len(X_test_0): ]:
    if i > 0:
        tn += 1
    else if i < 0:
        fn += 1
```

5. **Training and Testing**

In this project, all the experiments are done using five-fold cross validation for both instances in class zero and class one.

Accuracy – Before Diagonalization	Class0	Class1
Quadratic - Maximum Likelihood	0.778	0.952
Quadratic – Bayesian	0.778	0.952
1-Nearest Neighbor	0.48	0
Ho-Kashyap	0.941	0.812

Table 1

Accuracy – After Diagonalization	Class0	Class1
Quadratic – Maximum Likelihood	0.778	0.952
Quadratic – Bayesian	0.778	0.952
1-Nearest Neighbor	0.49	0
Ho-Kashyap	0.8754	0.854

Table 2

6. **Final Notes**

First, according to part 3.1, since the mean calculated from the Maximum likelihood method is used to calculate the Bayesian mean, these two means are almost the same. Also, the covariance matrix calculated in the maximum likelihood method is used for the Bayesian method. Therefore, as can be seen in table1, the accuracy of quadratic functions using these two methods are the same.
Moreover, in this experiment, the number of neighbors for KNN model is equal to one. According to table1 and table2, 1NN is not working well for class 1. The reason could be that since the mean of instances in class 0 and class 1 are near each other, i.e., the points in both classes are so close to one another, 1nn cannot properly classify points for these classes.
The most promising algorithm is Ho-Kashyap algorithm. This algorithm finds the separating hyperplane for linearly separable data and makes sure to minimize the number of misclassified instances. According to table2, diagonalization has improved the accuracy for class1, but not for class0.

Lastly, in order to extend this system to be a complete statistical pattern recognition system, we can change the system in a way that can be able to deal with multi-class classification problems. Also, more complex algorithms can be added and compared, for example SVM, that can be used for linearly non-separable data. Moreover, since 1NN does not work well in this dataset, we can compare 3NN's and 5NN's performances and see if an improvement can be achieved.

Please refer to this link for all the plots, and the source code.