Machine Learning Engineer Nanodegree

Capstone Proposal

Udacity
April 25nd, 2019

Proposal

Domain Background

KDD Cup 1999 Data is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining.

The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections.

This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

Here is a paper made an Intensive Preprocessing of KDD Cup 99 for Network Intrusion Classification Using Machine Learning Techniques.

Problem Statement

The problem is to build software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders.

The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections.

Datasets and Inputs

Data will be used is (NSL-KDD) dataset set it’s a data set suggested solving some of the inherent problems of the KDD'99 data set.
It’s a subset of (805050 rows of TCP connections) with 42 feature describing connection details.
Each connection is labeled as either normal or as an attack, with exactly one specific attack type. The Label consists of 5 unique classes (Normal, DOS, PROBE, R2L, U2R)
The number of each class in the data set is:

Class	Train	Test
Dos	391458	229853
Normal	97278	60593
PROBE	4107	4166
R2l	1126	16347
U2R	52	70

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic.

Solution Statement

It’s a class classification problem which can be solved with several algorithms like Neural networks, Naive Bayes, Decision trees, Support vector machine.
There are two suggested approaches to solve our problem:

First: To think of the task as a binary classification task to only classify between bad and good connections and mark the 4 intrusions (bad connection) with the same label in the data set as it's mainly our target which will help to overcome unbalance in the data and make the problem easier.

Second: is to work on a multi-classification task and try to classify all classes including different types of intrusions which will be harder but more benefit and it's common to be used this way.

The suggested solution to this problem to build deep learning model using
1D Convolutional neural network, it’s very effective when we expect to derive interesting features from shorter (fixed-length) segments of the overall data set and where the location of the feature within the segment is not of high relevance.

I think Naive Bayes and SVM will be also a good choices for this classification task and will be in mind if CNN doesn’t do well.

Then I'll compare its results with another model which will be Ensemble Method.
To overcome unbalanced data, we will use class weight and we may down sample "Dos" class.
We will use google Colab GPU to train our model on it.

Benchmark Model

We will build a model with Ensemble Methods (Gradient Boosting) to use it as a benchmark model and train, test the model with same data set then compare its result with our CNN to make clear, objective comparisons between the two.

Evaluation Metrics

We will use the accuracy metric for training the model, but we will tune classes weights to handle unbalanced data.
We will use AUROC and confusion matrix to evaluate model performance.

Project Design

Project workflow will be as the following:

1. Import and install dependencies on google Colab.
2. Load NSL-KDD database.
3. Examine data samples and features.
4. Break training and testing data, into features and labels.
5. Features encoding and scaling and downsampling.
6. Define and train Benchmark model (Gradient Boosting).
7. Define the CNN Model Architecture:
   a. Layer1: Convolution1D (60)
   b. Layer2: MaxPooling1D (2)
   c. Layer3: Flatten
   d. Layer4: Dense (128)
   e. Layer5: Dense (5)
8. Compile and train the model and (tune class weights)
9. Load the model with the best validation accuracy.
10. Build a confusion matrix function.
11. Compare classification performance f1 score, precision and recall with benchmark model.