Machine Learning Engineer Nanodegree

Capstone Proposal

Sourish Banerjee
April 29th, 2018

Proposal

Domain Background

Steganography is the art of hiding messages in images by altering the least significant bits of the pixels in images with that of the message bits. The result is an image with a message hidden in it. However, the change is imperceptible to the human eye. This is because on changing the least significant bits in the pixels of an image, the pixel values are only altered by a small amount, resulting in a visually similar altered or steg image obtained from the original or cover image. Steganography has been widely used because of its potential capability to hide the existence of sensitive data. In situations where this kind of data hiding is illegal, potentially dangerous or inherently unethical, it becomes necessary to detect the presence of steganography in images.

Steganalysis is the art of detection of steganography in an image. There are two major types of steganography, and the preferred steganalysis methods for them are also different [1]. The naïve method is called LSB embedding. In this method, the LSB bit remains unchanged if the message bit is the same as the LSB bit, otherwise, the bit is altered. Hence, the odd pixels are reduced by 1 in intensity, whereas the even pixel values are incremented by 1. However, this causes an imbalance in the image histogram, which can be easily exploited by statistical methods for steganalysis. The second method of LSB steganography, LSB matching solves this issue by randomly incrementing or decrementing the pixel values by 1 in case of an LSB bit mismatch. This avoids the issue of histogram imbalance and makes it difficult to perform steganalysis by statistical methods alone. In this project, I will look into the problem of steganalysis of LSB matching in greyscale images.

Problem Statement

My main goal is to classify greyscale images based on whether they are corrupted with LSB matching or not. I will use a labelled dataset of greyscale images, both with and without LSB matching and develop a supervised learning workflow to solve this problem.

Datasets and Inputs

I will use the BOSSbase 1.01 dataset for my project. The original dataset has 10,000 greyscale images, each of size 512 x 512. All the images in the dataset are treated as the cover images. From this dataset, I have generated 10,000 corresponding images corrupted with LSB matching steganography with a payload of 0.40 (payload refers to the fraction of pixels in the original image that has been corrupted due to the steganography process). For this purpose, I have used the tool available in [2]. The command to generate the steg files is as follows:

$ python aletheia.py lsbm-sim bossbase 0.40 bossbase_lsb

The final dataset of 20,000 images is the raw dataset for my project. From this dataset, I will generate a csv file, which will serve as the final dataset for my project. Each row of the csv file will represent an image and the columns will correspond to the extracted Correlation Features (CF) of the images as explained in [3]. The target feature will be 0 if the image is a cover image and 1 if the image is a steg image. The code to generate the correlation features for the images will be authentic. As supporting material, I have included a sample of the BOSSbase dataset as well as its counterpart after performing LSB matching. The link to download the complete BOSSbase dataset is provided in the README.md file.

Solution Statement

The final goal of my project is that given any greyscale image, to detect whether it is corrupted with LSB matching or not. My approach will be to build a voting ensemble of classifiers to give a consensus on this problem on the basis of majority vote. To achieve this, I will train several supervised classifiers on the dataset containing CF data of both images which are corrupted by LSB steganography, as well as those which are not. I will then select those trained models which perform the best as the models in my learning ensemble.

Benchmark Model

I will compare the performance of my final classification model against a benchmark model trained by a Gaussian Naïve Bayes learner. This will serve as a check for solvability of the problem undertaken and provide a basis for comparison and interpretation of the performance scores of our final model.

Evaluation Metrics

I will use the F-Score as an evaluation metric for my benchmark and solution model. The F-Score is defined as the harmonic mean over precision and recall for a given test. Precision is the number of correct positive results (True Positives) divided by the number of all positive results returned by the classifier (True Positives + False Positives), and recall is the number of correct positive results True Positives) divided by the number of all samples that should have been identified as positive (True Positives + False Negatives). The above metrics are formalized as follows:

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F-Score = (2 * Precision * Recall) / (Precision + Recall)

Project Design

The basic workflow of my project will include data exploration and pre-processing, followed by learning to generate a classification model.

The first step in data pre-processing is data cleaning. Due to the nature of the CF feature set, I might end up with nan values in some entries of my dataset (potentially caused by overly uniform LSBP, which in turn might cause some correlation and autocorrelation features to tend to infinity; Pearson's correlation coefficient has standard deviation values in its denominator which tend to zero in these cases). These nan entries if present, will have to be removed. I will plot the scatter matrix and heatmap of the features in the dataset to spot correlations between the same. If I notice the presence of several strong correlations, I might perform dimensionality reduction because in such situations, it is often possible without a significant loss in information. Removal of outliers is also often essential, and I might perform the same based on the sensitivity to outliers of the algorithms I will use for data transformation and classification purposes. For example, algorithms like Principal Component Analysis is highly sensitive to outliers.

Once the data pre-processing is complete, I will train several classifiers on each of the original, cleaned and reduced (if applicable) datasets. For each of the classifiers, I will note the performance based on the train time, prediction time, train accuracy, test accuracy, train F-score and test F-score for different training sizes. Based on all these factors, I will choose the 2 best performing classifiers for hyper parameter tuning. Finally, I will export the tuned models as pkl files for my final voting ensemble. The final voting ensemble will take a greyscale image as input, generate its CF feature set, and use a majority vote of the tuned models to predict whether the image is a cover or steg image.

References

[1] Khalind, Omed, and Benjamin Yowell Yousif Aziz. "LSB Steganography with Improved Embedding Efficiency and Undetectability". Computer Science & Information Technology 5.1 (2015): 89-105.

[2] Aletheia – Open Source Image Steganalysis Tool

[3] Liu Q, Sung AH, Xu J, Ribeiro BM. "Image Complexity and Feature Extraction for Steganalysis of LSB Matching Steganography". In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on 2006 Aug 20 (Vol. 2, pp. 267-270). IEEE.