CS5658 Anomaly Detection Homework1 – MNIST

Due Date: 3/28 23:30

The MNIST dataset is a well-known collection of 70,000 grayscale images of handwritten digits (0 through 9), each represented in a 28x28 pixel format. Comprising a training set of 60,000 images and a testing set of 10,000 images.

In this task, we are going to apply different anomaly detection methods to MNIST. The experiment settings are as follow:

1) Choose a digit as a normal class (label=0) and treat all other classes as anomaly data (label=1).
2) Apply Principal Component Analysis to raw data, reduce dimension to 30.
3) For the training set, only normal digits are included for the training process.
4) For the testing set, we resample the anomaly digits into 10% size of normal digit to simulate the anomaly detection scenario.
5) Fit different models using training samples and predict the label of each testing sample.
6) Calculate the ROC-AUC score of the prediction.
7) Repeat step 1~5 for each digit (0~9) and record the ROC-AUC for each digit.
8) Calculate mean ROC-AUC scores of all digits.

Problem:

1. (20%) K Nearest Neighbor

Implement KNN classifier using Euclidean distance with k=1,5,10 according to the lecture slides. Use the training set which contains only normal digits for distance comparison. Show the mean ROC-AUC of all digits.

2. (20%) Cluster-based

Implement k-means clustering using Euclidean distance with k=1,5,10 according to the lecture slides. Use the training set which contains only normal digits to construct the cluster. Show the mean ROC-AUC of all digits.

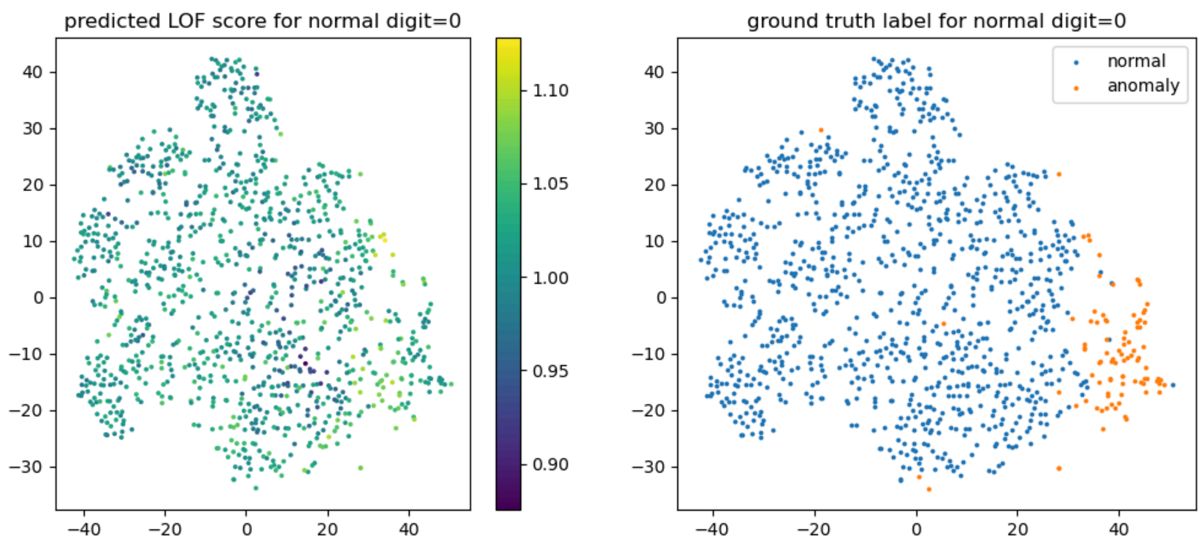
3. (20%) Distance-based

Implement distance-based detection algorithm (k=5) according to the lecture slides. Use the testing set to compute pairwise distance directly, and you will not use the training set in this problem. Show the mean ROC-AUC of all digits. You need to implement the distance function by definition.

   (a) Cosine Distance
   (b) Minkowski Distance (r=1, 2, inf)
   (c) Mahalanobis Distance (Note: The covariance matrix used in Mahalanobis Distance should be calculated using normal training samples only)

4. (20%) Density-based

   (a) Implement Local Outlier Factor detection algorithm according to the lecture slides. Use Euclidean distance as the distance metric. Similar to Problem 3, you will not use the training set in this problem. Show the mean ROC-AUC of all digits.
   (b) Visualize LOF detection results of **normal digit=0** using t-SNE. You need to plot two figures **in the same image**. (1) Color testing set using LOF anomaly scores. (2) Color testing set using ground truth label=0/1.



5. Report (20%, 5% for each method)

Analyze the performance and reasons behind for each method, write down your observations. (e.g. why this method performs bad, why this hyper-parameter performs good.)

Note (Important):

1. Do not modify the sample code segment.
2. You cannot use APIs from "sklearn" or any other libraries that can directly calculate output.(e.g. sklearn.neighbors.KNeighborsClassifier, sklearn.cluster.KMeans, sklearn.neighbors.LocalOutlierFactor…)
3. The report should contain:
   (i.) your implementation code
   (ii.) explanation of code
   (iii.) performance (ROC-AUC)
4. You should provide a "ReadMe.txt" file about how to run your code.

5. Try adding comments as much as you can to better understand your code.
6. You should submit a HW1_{Student-ID}.zip (ex:HW1_123456789.zip) containing only the following files:
    a. hw1.py
    b. ReadMe.txt
    c. report.pdf
    d. tsne.png
7. Make sure you explain everything you want to show in the report, not in your code.
8. All results should be shown in your report, not in the console or the pop-up window, or you will get 0 points.
9. Discussion of homework is encouraged, but you have to write your own.
10. Copying or submitting AI-generated documents/code is strictly prohibited.
11. Scores of late homeworks will be reduced by **20% per day**.
12. If you have any questions, please pose your questions in the eeclass.

Tips:

1. Some helpful functions & libraries:
    a. numpy
    b. torchvision (ref: https://pytorch.org/)
    c. sklearn.metrics.pairwise_distances(ref:https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise_distances.html)
    d. sklearn.manifold.TSNE (ref:https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html)
    e. sklearn.metrics.roc_auc_score (ref: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html)