Anomaly Detection
Lab 4 - One Class SVM

Andrei Hîji

November 2024

In this lab we will use anomaly detection algorithms like OC-SVM and DeepSVDD.

1 OC-SVM, SVDD main ideas

OC-SVM

$$\min_{w,\xi\in R^m,\rho\in R} \frac{1}{2}\|w\|^2 + \frac{1}{m\nu}\sum_{i=1}^{m}\xi_i - \rho \tag{1}$$

$$\text{s.t. } \langle w, x_i\rangle \geq \rho - \xi_i, \xi_i \geq 0 \quad \forall i \in \{1, \cdots, m\}. \tag{2}$$

Main parameters:

- kernel used
- ν - upper bound on the fraction of outliers and lower bound on the fraction of SVs

SVDD

$$\min_{c,R\geq 0,\xi\in R^m} R^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } \|c - x_i\| \leq R^2 + \xi_i, \xi_i \geq 0 \quad \forall i \in \{1, \cdots, m\}.$$

Deep SVDD

$$\min_W \frac{1}{n}\sum_{i=1}^{n}\|\phi(x_i; W) - c\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L}\|W^l\|_F^2$$

- L - number of hidden layers
- W^l - the weights of layer l
- c - the center of the hypersphere

A few methods that will be useful to you throughout this lab session are:

- **generate_data()** is used to create a synthetic dataset where normal samples are generated using a multivariate Gaussian distribution while outliers are generated by a uniform distribution; its parameters include:
  - number of training/testing points: n_train, n_test
  - number of features: n_features
  - proportion of outliers: contamination
  - range of values: offset
- **sklearn.model_selection.train_test_split** splits datasets into random train and test subsets and include the parameters:
  - Input data: *arrays
  - Test data size and train data size: test_size=None, train_size=None
  - random_state=None
  - shuffle=True
  - stratify=None - used to split in a stratified fashion
- **sklearn.metrics.roc_auc_score** computes ROC AUC:
  - True labels: y_true
  - Obtained scores: y_scores
- **scipy.io.loadmat** loads data from a MATLAB file and includes the following parameters:
  - file_name
  - Dictionary in which to insert matfile variables: mdict=None
  - appendmat=True : to append the .mat extension to the end of the given filename
- **pyod.utils.utility.standardizer** transforms data to zero-mean and unit variance; it includes the parameters:
  - Training samples and test samples: X, X_t=None
- **numpy.quantile** computes the q-th quantile of the data along the specified axis (the value below which the specified percentage of data falls); its parameters include:

  - Input array: a
  - Probability or sequence of probabilities of the quantiles to compute. Values must be between 0 and 1 inclusive: q

- **numpy.random.uniform** draws samples from a uniform distribution; its parameters include:
  - Lower boundary and Upper boundary of the output interval : low, high
  - Output shape: size

2 Exercises

2.1 Ex. 1

1. In the first exercise you will use **generate_data()** function from **pyod.utils.data** to generate a 3-dimensional dataset with 300 training samples and 200 test samples using a contamination of 0.15.

2. You will fit an **OCSVM** model (from **pyod.models.ocsvm**) using the training data and then you will predict the labels for test data. Use the linear kernel and the contamination rate specified above. Compute the balanced accuracy and the ROC AUC for the test data.

3. Create 4 3D subplots where you will plot the test and training data (using different colors for inliers and outliers) for both the ground truth and the predicted labels.

4. Observe how the predicted labels change when you use an RBF kernel.

5. Perform steps 2 and 3 for **DeepSVDD** model from **pyod.models.deep_svdd**

2.2 Ex. 2

1. In this exercise you will use the cardio dataset from ODDS (`https://odds.cs.stonybrook.edu/cardiotocogrpahy-dataset/`). You will split the dataset into a training set consisting of 40% of the data and a testing set.

2. You will use a cross-validation grid-search over a predefined map of possible parameters of **OneClassSVM** (from sklearn.svm) in order to find the ones that offer best results. The map of possible parameters must contain different kernels, different values for gamma parameter (for kernels that use it) and different values for the ν parameter (including the contamination rate of train set).

3. **GridSearchCV** from sklearn.model_selection will be used to perform the grid-search over the training set using Balanced Accuracy as score. The same metric will be reported for the model trained with the best parameters discovered by **GridSearchCV** (tested on the test set).

4. **GridSearchCV** will have to perform standardization; for this, it will use a **Pipeline** from **sklearn.pipeline** that will perform standardization using **StandardScaler** before fitting the **OneClassSVM** model.

5. You have to pay attention to the labels returned by **OneClassSVM** model (from **sklearn**): -1 for outliers and 1 for inliers. If you want to convert them to the **pyod** format you can use

$$label\_pyod = (-1 * label\_sklearn + 1)/2$$

   In the current exercise you will perform the conversion from **pyod** format to **sklearn** for the ground truth labels

6. Print the best parameters discovered by **GridSearchCV** and the balanced accuracy obtained by the model (trained with those parameters on the entire training dataset) when evaluating the testing set.

2.3 Ex. 3

1. For this exercise we will need to download the shuttle dataset from ODDS (`https://odds.cs.stonybrook.edu/shuttle-dataset/`). Load the data using **scipy.io.loadmat()** and use **train_test_split()** to split it into train and test subsets (use 50% of data for testing). Normalize your data accordingly.

2. Fit **OCSVM** and **DeepSVDD** models (**from pyod**) using the training data and compute the balanced accuracy (**BA**) and the area under the curve for the testing data for both models (**ROC AUC** - using **sklearn.metrics.roc_auc_score**).

3. Use different architectures for **DeepSVDD** and compute the 2 aforementioned metrics for each case.