Run One-Class SVM

Objective

- There are many anomaly-detection techniques. In machine learning, we often have to try several algorithms before we are happy with the performance of our model. One of the most popular algorithms in machine learning is the support-vector machine (SVM). This algorithm is a linear classifier that uses a technique called the “kernel trick” in order to adapt to different datasets and problems. Each kernel has a different set of parameters. In practice, the most popular ones are the RBF kernel and the polynomial kernel. A complete exposition of SVMs is beyond the scope of this milestone, but there are additional resources provided.
- In this milestone, your goal is to run a one-class SVM with an RBF and a polynomial kernel of degrees 2 and 3. You will record their performance on classification metrics such as Average Precision, F1 score, and Precision-Recall. Your goal is to see whether the performance of this algorithm on the task of predicting problems with the thyroid is going to be better than the other families of algorithms we’ll deal with in subsequent milestones. To avoid repeating code when evaluating multiple variants of the algorithms, feel free to use the functions and the code snippet from the starter template.

Importance to project

- This project is about comparing different approaches to anomaly detection. We will compare a one-class SVM to other approaches. The reason we’re starting with this algorithm is that SVMs are well-studied and can work very well for a variety of problems.

Workflow

1. Split the cleaned dataframe from the previous milestone into train and test data, ensuring it’s done in a stratified way.

2. Quantify the number of outliers in the sample to allow this number to be used as a parameter in the algorithms.
3. Separate out the samples corresponding to just the inliers class in the training data: this will be used for training.
4. Import the scikit-learn package (install it if it is not already installed using `pip install -U scikit-learn`)
5. Instantiate three one-class SVM by calling the method `sklearn.svm.OneClassSVM()` three times within a dictionary. For each instance, specify `nu=outliers_fraction*4` and different kernel parameters for the three different instances as follows. (See "Resources" below for more details about kernel trick.)
   - Use `kernel='rbf'` for specifying the Radial Basis Function (RBF) as the kernel for the first instance.
   - Use `kernel = 'poly', degree = 2` as the kernel for the second instance.
   - Use `kernel = 'poly', degree = 3` as the kernel for the third instance.
6. Initialize a DataFrame to hold the classification metrics of interest (Precision, Recall, F1 Score, Average Precision, TN, TP, FN, FP).
7. Fit each of the previously instantiated models on the inliers data from step 3.
8. Evaluate the fitted models on the test features using `custom_classification_metrics_function` from the starter template and store the classification metrics in the DataFrame you initialized in Step 6. Note that, unlike supervised classification algorithms, one-class SVM doesn't have a native `predict_proba()` method. See "Notes" below on what to edit.
9. Inspect the results.
   - What do you observe based on the F1 score, Precision, Recall, and Average Precision? Does performance increase or decrease with different kernels and degrees?
10. Copy the metrics to another DataFrame, which can be used to compare performance of the algorithms across different milestones.

Notes

- You can use the provided pre-processed data as the starting point for Milestone 2 if you had difficulty with Milestone 1.
- We are using only the subset of the training data that belongs to the inlier class.
- You will need to modify the custom_classification_metrics_function function from the starter template to make it suitable for the scikit-learn's Anomaly Detection algorithms Right after the command test_pred = classifier.predict(X_test) you will need to insert these two lines of code because scikit-learn's predict method generates labels with -1 for the anomalies and 1 for the inliers and we need to change it 1 and 0 to make it compatible with how we typically think of binary labels.

```
test_pred = test_pred*-1
test_pred[test_pred==-1]=0
```

Also, you need to remove the line for predict_proba() and insert the following:

```
decision_score_list = classifier.decision_function(X_test)
scaled_decision_score_list = MinMaxScaler().fit_transform(decision_score_list.reshape(-1, 1))
y_scores = [1-item for sublist in scaled_decision_score_list for item in sublist]
```

- Data scientists working on skewed data problems need to keep two things in mind: defining and capturing the right metrics and interpreting those metrics.
    - **Defining and capturing the right metrics**: In the custom classification metrics function, we have intentionally captured the macro-averaged metrics for the Precision, Recall, F1, and Average Precision scores. Specifying the macro version ensures that scikit-learn corrects for the skewed data and weights both classes equally.
    - **Interpreting metrics**: The correct interpretation relies on the use case and there is no one-size-fits-all solution. In the case of thyroid disease detection, we would typically want as many of the positive instances (in this case, the outlier class) to be captured at the expense of generating more false positives. This is because, even if there are false positives, a detailed screening can clear them out; but if a patient's case is not detected at all (i.e. false negatives), then they are sent home and lost to follow-up evaluation.

Deliverable

The deliverable for this milestone is a Jupyter Notebook with a report on your results. Use the custom function from the starter template to get binary classification metrics.

Write down your observations and conclusions. Which type of SVM performs better on this problem? Take a note of this performance, as we will compare it to other algorithms.

Upload a link to your deliverable in the Submit Your Work section and click submit. After submitting, the author's solution and peer solutions will appear on the page for you to examine.

Help

Feeling stuck? Use as little or as much help as you need to reach the solution!

resources

Machine Learning in Action by Peter Harrington

Chapter 6, "Support vector machines," explains vector machines further.

"Supervised vs. Unsupervised learning, and train/test" from *Machine Learning, Data Science and Deep Learning with Python* by Frank Kane provides a good introduction to machine learning with Python.

"Support Vector Machines (SVM) Overview" from *Machine Learning, Data Science and Deep Learning with Python* by Frank Kane provides a foundation to understand SVMs in general and as an anomaly detection algorithm.

Additional resources

- SVM scikit-learn page
- One-class SVM scikit learn page
- Support Vector Machines Part 1 (of 3): Main Ideas!!!
- Scikit-learn page on Novelty & Outlier Detection
- SVM Original Paper on Novelty Detection
- Wikipedia article on One-Class Classification
- One-Class SVM example
- Wikipedia article on Kernel method

- Visualizing decision boundary: comparing anomaly detection algorithms for outlier detection on toy datasets