Anomaly Detection Using One-Class SVM

Alex Lapinski

I. Introduction

For this final project of CS 613 I have examined how machine learning, and more specifically, anomaly detection can be applied to Data Quality Analysis. Specifically in this project I have examined two main papers [1] and [2]. In these papers, the authors describe use of one-class svms and a more novel approach called SVDD.

One-Class SVMs are a modification on the more traditional binary classification problem solved by SVM classifiers. In a binary classification problem, we have 2 classes, and attempt to find a hyperplane to separate these two classes of data. In a one class svm, we only train the hyperplane using one class of data. This is typically the normal data. We start with the origin of the dimensional space as the only outlier and attempt to find a hyperplane that will maximize the distance between the origin and the normal data. Finally, we test this classifier using both normal and anomalous data in the hopes that the hyper plane does not include the anomalous data as normal data.

The more novel approach of SVDD takes this one-class SVM a step further. Instead of using a hyperplane to define the normal data, it instead uses a closed shape, specifically a hypersphere. Since a sphere is closed around the one class of 'normal' data, it can allow for anomalous data appearing anywhere outside of the hypersphere, not just between the hyperplane and the origin. Ultimately, this should provide a more robust structure to define the normal data and detect more of the outlier data than a hyperplane would detect. It also ensures that more of the normal data can be contained within the hypersphere ensuring fewer false negatives.

We can then take these two approaches to outlier detection and apply them to the problem of data quality. Data quality is a difficult problem to solve, and is rather important in the professional field of data science and data management. It can even be extended to detect faults in a system without the need to put that system into a faulty state. We can do this by training the anomaly detection model on the easy to produce 'good' data. We then test this model in production against any anomalous data available, or when that data does become available.

We can continue to improve this model as more data becomes available and use this classifier to even detect data once it is made available. Since evaluation of a datapoint against a trained model is rather quick, we can even embed this detector within running systems with out introduction of overhead. Througout the next few sections, I will describe the process I went throught to prepare the data, select hyperparameters for the model and finally evaluate the performance of the two classifiers.

II. Problem

For this project, I examined 3 separate datasets in order to apply one-class svm (using hyperplanes and hyperspheres). The general approach was to first cleanse and process the data sets, then perform grid search to find the best parameters for each kernel for each dataset. Once the optimal parameters were determined (using training data), the final training/testing was performed. The last step of this project was to briefly examine the use of hyperspheres, specifically named 'SVDD'. These were explored by other researchers as a way to minimize the space around the normal data. [2]

For each dataset I first processed the raw data by reading the files into python. The first two data sets 'Water Treatment Plant'[5] and 'Banknote Authentication'[5], these datasets were pre-cleansed by the authors of the dataset. This meant that I did not need to eliminate columns due to features which had faulty data and did not need to do any pre-processing other than standardization and classification. The hard disk drive data [7] however did require some cleansing of data and features.

Once the data was prepared, I performed cross validation in order to search for the best hyperparameters for each kernel and dataset. The next step was to use the original data to test the performance of each kernel side by side and to briefly experiment with the svdd. The final step is the analysis of the performance between different kernels on 3 data sets compared against the SVDD technique. Specifically on the performance of each method.

In the following sections, I will describe briefly my approach to exploring the datasets, cross validation and parameter search, and finally testing of each of the best parameters for one-class svm and svdd.

A. Processing Data

One item that was common to all data sets was the fact that some samples were missing data. For some data sets this was represented as '?' or 'NaN'. Since the One-Class SVM requires that all columns have some value, I needed to determine what to do with these samples. For the water treatment and banknote data, there was simply not enough samples to eliminate those with missing values. Not to mention that if I eliminated those samples, and those were anomalous, that would defeat the purpose of this project. Instead, I explored the water treatment dataset with various strategies for imputing the missing values from other training data. The strategy which worked best (SkLearn provides 3 strategies) was to use the most frequent value for the missing values.

The sklearn library provides a mechanism to train an imputer on a dataset and then use that imputer later to fill in missing values. I chose to use this mechanism and trained the imputer on the training dataset (similar to how we standardize data

and use the mean). I then used this trained imputer when standardizing test data to be fed to the classifier.

1) Processing Water Treatment Plant: The water treatment plant had a total of 38 attributes, 527 samples and 13 classes. Each of the attibutes in this data set represents either the measured input or output values from parts of the water treatment plant. Given that this data represents a complex system and also included data for non-normal performance, was a good choice for outlier detection. We have data available for the normal operation, which is typically easy to obtain, and data during non-normal scenarios for testing against a trained model.

This is a relatively small sample set and required separation of the 13 classes into two classes 'normal' and 'anomalous' classes in order to make use of one-class svm and svdd. The individual classes (previously identified by the author) were easily separated into normal operations and anomalous scenarios. The classes I included in my normal data set were 'class 1', 'class 5', 'class 8', 'class 9', 'class 10', 'class 11' and 'class 12'. The classes included in the anomalous data set were 'class 2', 'class 3', 'class 4', 'class 6', 'class 7' and 'class 13'.

Once the normal and anomalous groups were identified, I was able to separate the date rates (each record in this dataset was labeled by date) into two groups. These groups were then labeled with '1' for normal and '-1' for anomalous. These intermediate data sources are stored and processed within the 'processed' folder of data for this project.

2) Processing Banknote Authentication: The banknote authentication data was more limited in comparison to the water treatment plant data, however, it did have many more samples available in the anomalous data set. This dataset was made up of data extracted from processed images of both authentic and forged banknotes. These attributes represented the metadata about the images and were labeled as both genuine (class 0) and forged (class 1). The processing for this dataset was just to read in the source of 1372 total samples and then re-label each sample to '1' for normal and '-1' for anomalous. This ensured that the same training and searching algorithms could function without change.

Again, the processed files were stored in a separate location to be used by the subsequent steps of this project.

3) Processing HDD: The hard disk drive data was a bit more problematic for two reasons. First there was significant more data available for use. Data was available for the past calendar year, separated by quarters. Each quarter had over 300 megabytes, one file for each day, each file around 65,000 samples. This meant that there was more than enough data to separate into both training and testing data. The other problem was that the feature space was unprocessed. It was simply raw data collected from tests of hard drives each day. I first needed to examine the features and find out that some of the columns were in hexadecimal, while others were in decimal. I then needed to convert the hexadecimal values to decimal, so they could be normalized before training. Even after conversion, there were 11 columns in the data (of 36 in total) where the standard deviation was zero. This caused a problem while attempting to normalize the data (dividing by zero) and required the elimination of these features entirely.

Since I had so much data available in this space, I selected an entire group of data points that were not examined until after the best parameters were selected. This was to ensure the accurate performance measurement of the different kernels and svdd.

B. Parameter Search

For the parameter search, I first segmented the same data sets that were previously prepared (including some anomalous data). The cross-validation / parameter search scripts then looked at all available data (except for hdd's test set) and performed various numbers of folds. Since the amount of data varied between each dataset, I decided to ensure that each iteration of the cross validation had at least 50 samples, and then the remaining data was used to test the model. For the HDD data, I chose 100 instead of 50 as more data was available.

Based on a paper available on the libsvm website [**?**], I was able to select ranges of parameters in a log space. That is for the nu parameter, I selected values ranging from 2^-15 to 2^0. I selected 10 values in this range which were exponentially increasing. I compared each value of each possible parameter for each kernel. This lead to an exhaustive search for a best parameter in this rough search space. Once this first pass was complete, I narrowed the range to +1/-1 of the 'best' parameter value. I continued this for 2 finer grained searchs to end up with the final best parameter.

This grid-based search of nested loops ensured that all possible combinations were attempted of parameters. I was then able to narrow in on a nearly optimal selection of parameters for each dataset. These paramters are saved and included within the project submission. In order to determine which paramters were better than others, I chose to use the f1-measure. [4] This measure was selected based on a different paper, who was also attempting to find the best parameters for novelty detection.

During the initial exploration of the data, cross validation and performance, I noticed that the precision and recall graphs (as well as ROC curves) kept generating a straight vertical line. That is, the model was always identifying 100% of all outliers (a good thing, but hard to find the best parameters visually). Instead, I needed to choose a combination of precision and recall that let me find a choice of parameters that not only caught all outliers, but minimized the number of false negatives (incorrectly identified outliers).

C. Evaluation

Once the ideal parameters were identified, I setup a separate script to train all of the combinations of one-class svm using various kernels. The best parameters from the previous section were used for each dataset. Each of the datasets were tested just once with the orignal dataset (except for hdd) and trained using only normal data. $\frac{1}{3}$ of the normal data was kept for testing the trained model, and all of the anomalous data was kept for testing as well.

For each of the datasets, I computed numerous measures, Precision, Recall, FPR (False Positive Rate), Accuracy and F1 Measure. In the next section, I will compare the differences in these results. In order to compute any of these measures I needed to identify the 4 base components. Number of False Positives, False Negatives, True Positives and True Negatives. Since, I was using what ammounts to a binary classification problem, I was able to use my anomalous data as one class, and the normal data as another class.

The computation of each of these counts was rather easy due to the way the testing on the classifiers was performed. The first test on a classifier sent in only normal data. So to compute the true positives, i count all of the items in the output from that iteration which were labeled correctly as '1'. False negatives are determined as those that were identified as '-1' in the normal test set. False positives are those in the anomalous test set which the classifier identified as '1' and true negatives, those in the anomalous dataset the classifier correctly identified as '-1'.

III. Results

For each of the one-class SVM training exercises, I was able to compute a wide range of performance metrics, however, with the svdd, since this tool was already built in c/c++ and did not appear to have any other performance metric than accuracy, I will only be able to compare the SVDD and one-class svm (hyperplane) using the rbf kernel and accuracy.

Before comparing the SVDD and Hyperplane one-class SVM, I will first examine the results between each of the kernels and each of the best performing kernel between each dataset.

First, the Water Treatment dataset.

	Polynomial	RBF	Linear	Sigmoid
Accuracy	0.65	0.95	0.59	0.92
F1 Measure	0.79	0.97	0.73	0.96
Precision	0.92	1.0	0.95	0.96
Recall	0.69	0.94	0.60	0.95
FPR	1.0	0.0	0.45	0.54

If we take a look first at the false positive rate, since we ultimately want to evaluate this machine learning technique to see how much of the anomalous data we can capture, we want to make sure our false positive rate is as low as possible. This means, that right away, the polynomial kernel is out, with 1.0 FPR, and the sigmoid does not perform as well. Actually, our best performing kernel for this dataset is actually the RBF with 0.0 FPR and 1.0 Precision. we correctly classify all of our normal data and all of our abnormal data. we only have 5% of our normal data mis-classified as anomalous.

Next, we review the bank authentication data

	Polynomial	RBF	Linear	Sigmoid
Accuracy	0.0	0.76	1.0	0.18
F1 Measure	inf	0.3	1.0	0.3
Precision	0.0	0.4	1.0	0.18
Recall	0.0	0.24	1.0	0.84
FPR	1.0	0.09	0.0	1.0

This dataset seems to be easily separable by a hyperplane, since the linear kernel performed perfectly. There really isn't a point to evaluate any of the other kernels as the linear kernel was perfect.

The last dataset to compare for one-class svm is the hdd dataset.

	Polynomial	RBF	Linear	Sigmoid
Accuracy	0.94	0.94	0.94	0.94
F1 Measure	0.97	0.97	0.97	0.97
Precision	0.99	1.0	1.0	1.0
Recall	0.94	0.94	0.94	0.94
FPR	1.0	0.0	0.0	0.0

For this dataset, we have nearly identical performance for everything except the polynomial kernel. This means we could select between any one of the three kernels 'rbf', 'linear' or 'sigmoid' and most likely the linear kernel would be selected in the field as this could be quickly trained and produce as good results as either rbf or sigmoid without requiring complex highger dimensions.

For each of these 3 datasets, we notice that the polynomial kernel performs horrible when it comes to the false positive rate, and in fact, identifies all of our anomalies as normal data. This indicates that we should never pick the polynomial kernel and instead should pick between rbf or linear kernel since they perform best overall. The last area to review is the performance between the rbf kernel for one-class svm and the svdd. For each dataset, I will compare the accurracy only, as that was the only output from the SVDD library. The RBF kernel was selected since that is the default kernel used by the SVDD algorithm, and it makes sense to have as much similarity in the dimensional space as possible.

	SVDD	One-Class RBF
Water Treatment	0.95	0.95
Banknote	0.9	0.76
HDD	1.0	0.94

From this result, we see that using SVDD for anomaly detection is as good or better than using one-class SVM. Even if we look at the performance for banknote authentication data, we see that while we do not get 100% classification as we did with the linear kernel in one-class svm, we do reach much higher than the RBF kernel in one-class svm. The outcome of this is that whenever we want to perform outlier or anomaly detection, that SVDD is a very good alternative to using one-class SVM.

IV. Conclusion

Throughout this entire project the majority of the work was spent in cleaning and exploring the datasets. This is rather representative of real-world applications of machine learning and was a good experience to prepare for further work in this space. If we take a closer look at the actual work in this project. That is to examine the use of One-Class SVM and SVDD for anomaly detection, and more specifically to

determine the quality of data at hand, we can easily see that both of these methods for anomaly detection are applicable and can be carried forward into the field. Upon reviewing the performance of the SVDD versus the one-class SVM, there is still more work to be done. Specifically, related to the metrics and measurement of the SVDD performance. In order to say it is always a clear winner over one-class SVM we must make sure that while the accuracy is high, the false positive rate is also as low as possible. If we do not, then we may easily indicate that the gap in accuracy could be the mis-classification of just our outliers if they represent a small percentage of the total population.

Once it has been established that SVDD is either better or worse compared to one-class SVM we can take this learning algorithm and move it into the real world. Specifically, we can train the model perhaps once a week or once a month on data being entered into a database for a financial system of record, then on each insert to the database, determine if the data is anomalous or normal. If it is detected to be anomalous, we can alert the mantainers of the system creating that data that in fact there may be an issue in the software. This approach is my next planned extension of this work at my current company.

REFERENCES

[1] Scholkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., & Platt, J. (1999). Support Vector Method for Novelty Detection. Nips, 12, 582-588.

[2] Tax, D. M. J., & Duin, R. P. W. (2004). Support Vector Data Description. Machine Learning, 54(1), 45-66. Retrieved from http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3189749\&tool=pmcentrez\&rendertype=abstract\nhttp://link.springer.com/10.1023/B:MACH.0000008084.60811.49

[3] Rajasegarar, S., Leckie, C., Bezdek, J. C., & Palaniswami, M. (2010). Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks. IEEE Transactions on Information Forensics and Security, 5(3), 518-533.

[4] Larry M. Manevitz, & Malik Yousef, (2001). One-Class SVMs for Document Classification. Journal of Machine Learning Research 2

[5] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[6] libsvm - beginners guide https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[7] Hard Drive Data and Stats - https://www.backblaze.com/b2/hard-drive-test-data.html