Smoothing Gamma Ray Spectra to Improve Outlier Detection

Vincent Barnabé-Lortie, Colin Bellinger, Nathalie Japkowicz
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada K1N 6N5
Email: {vbarn021, cbell052}@uottawa.ca, nat@site.uottawa.ca

***Abstract*—Rapid detection of radioisotopes in gamma-ray data can, in some situations, be an important security concern. The task of designing an automated system for this purpose is complex due to, amongst other factors, the noisy nature of the data. The method described herein consists of preprocessing the data by applying a smoothing method tailored to gamma ray spectra, hoping that this should decrease their variance. Given that the number of counts at a given energy level in a spectrum should follow a Poisson distribution, smoothing may allow us to estimate the true photon arrival rate. Our experiments suggest that the added data preprocessing step can have large impact on the performance of anomaly detection algorithms on this particular domain.**

I. Introduction

In many situations, particularly in large public events, monitoring radiation levels can be an important security concern. In such cases, most spectra produced by a gamma-ray detector show nothing but background levels of radiation.

It is critical, however, to be able to rapidly identify those spectra that tell a different story: if, for example, an individual walked by a detector with a dangerous amount of radioactive material, we should, hopefully, be able to detect it quickly by analyzing the spectra.

Given the enormous number of spectra that may come in hourly, however, it is not possible for a team of physicists to comb through every single one in search of anomalies. We also typically wish to obtain results faster than human resources could provide. Typically, automated methods like template fitting and peak fitting are used to detect anomalies. Machine learning methods, however, are also particularly suited to this problem, as they could learn to discriminate anomalous spectra from background levels of radiation.

The research described in this paper is not our first attempt at solving this problem using machine learning. We have had some success in the past using a variety of different methods [1]. The scope of this paper, however, is limited to one of our latest attempts at increasing classification performance on this particular domain, where we use smoothing to counter the high variability effect of the Poisson process of gamma-ray arrivals. Specifically, we apply a specific smoothing method first introduced in [2], which was designed specifically to emphasize important information about gamma-ray spectra, to preprocess the data before using it to train a model.

Our experiments, where we compare the performance of classifiers trained and tested on the smoothed data to our previous results, yield encouraging results, suggesting that the smoothing regularly leads to significant performance gains. We do not claim that the specific method of smoothing we used is superior to other methods; only that smoothing can have a positive impact in general. For this reason, we do not directly compare the smoothing method by Burr et al [2] to other smoothing methods.

The remainder of this paper is structured as follows: In Section 2, we describe the problem at hand in more detail, as well as previous work in this domain. In Section 3, we detail the proposed method, which makes use of Burr et al's smoothing method. Then, Section 4 explains the experimental methodology we followed to study the merits of this method. The results of our experiments are documented in Section 5 and discussed in Section 6.

II. Background and Related Work

This section describes methods commonly applied to deal with gamma-ray anomaly detection - namely outlier detection methods and one-class classification learners - and the smoothing method for gamma-ray data introduced in [2].

A. *Outlier Detection*

There are many techniques that tackle the problem of outlier detection directly, rather than framing it as a classification problem.

First, there are statistical testing based methods, which typically assume that the "background" class follows a certain kind of statistical distribution (often Gaussian), estimates the parameters of that distribution, and mark as outliers points that are unlikely to have been generated by that distribution (e.g. they deviate from the mean by more than 3 standard deviations). If we assume a multivariate gaussian distribution, then the Mahalanobis distance [3] can be used to score the level of anomaly of an instance. Similarly, deviation-based approaches such as proposed in [4] judge points to be outliers when their removal from the dataset causes a significant decrease in its variance.

Second, there are depth-based methods [5] which see the domain as an onion where each layer is the convex hull of the data when all outer layers are removed. In those methods, the outliers are the objects on the outer layers.

Third, distance-based approaches, of which there are many variants, generally judge the level of "outlierness" of an instance based on how far its nearest neighbors are from it. For example, DB(ε,π) Outliers [6] are points for which, given a radius ε and a percentage π, at most π percent of all other points are closer than ε.

Fourth, there are density-based approaches, such as the Local Outlier Factor [7], which consider the density around a point relative to the density around its nearest neighbors. If a point is in a very sparse area relative to its neighbors, then it is likely to be an outlier.

Without going into further detail, there are also clustering-based techniques [8], which cluster the data and find instances that are far from cluster centers; information theoretic approaches such as, for example, the Kolomogorov complexity [9]; and many more.

Finally, there are classification-based anomaly detection techniques, which use a learned classification model to distinguish between background instances and anomalies. While traditionally these models were trained on instances of both classes, many methods now learn from instances of only one of the classes. These methods are discussed in Section II-B.

B. One-Class Learning

When treating anomaly detection as a classification problem, we are often faced with the problem of rarity: examples of the anomalous class are not only vastly outnumbered by those of the background (or normal) class, they are also rare in absolute terms. The consequences of these problems are well known to be severe [10], even moreso when the data is noisy [11], in which case background instances that are particularly noisy may look very much like the anomalies.

In addition, we may not have instances representative of all of the outlier subconcepts. For example, in the domain of gamma-ray spectra, we may have multiple subconcepts of the anomalous classes: anomalies caused by a Uranium source, anomalies caused by a Technetium source, etc. To train a binary classifier for the task of outlier classification, we may need examples of each of these subconcepts, which is not practically feasible.

Fortunately, an alternative to binary learners, the one-class classifiers, which learn only one concept, and discriminate between what does and does not belong to that concept, can be used to alleviate both of these issues. For instance, one-class support vector machines (like ϵSVM [12]) can be used to learn a hyperplane that wraps around the class to learn. In addition, in [13], it was shown that the unsupervised variant of feedforward neural networks, the autoassociator, when taught to recognize a concept and reject instances that do not belong to it, could outperform its supervised equivalent, the multilayer perceptron, on certain binary classification tasks.

These one-class learners are not affected by rarity of the minority class when they are trained to recognize the majority class. In addition, since they flag anything that does not look like the learned concept as anomalous, they can deal very well with previously unseen subconcepts of the minority class.

C. Smoothing Gamma-Ray Data

The presence of noise in the data, particularly in the majority class, may perturb the learning of the minority class even further, especially when it is rare, since we may end up with more noisy majority instances leaking into the minority class' space than there are representative examples of that class. This problem, overlap, has recently been studied in more detail [14] and shown to be as serious a problem if not more so than the class imbalance.

Again, noise is also an issue that applies to the gamma ray spectra domain. The number of photon counts a detector will see at a given energy level follows a Poisson distribution, parameterized by their true arrival rate. The variance of these counts is therefore also linked to this rate, and the counts, when samples are short, can be quite noisy.

A popular method for noisy data is to, when possible, apply smoothing methods, in the hope that it might reduce the variance. In [2], Burr et al. propose a smoothing technique designed specifically for spectral data.

As they explain, when spectra are measured over short time intervals, the effect of the Poisson process' variance are particularly obvious. If the expected number of counts at a particular energy level in a given time interval is 1, for example, then it wouldn't be very surprising to actually see none at all, or to see 1 or 2 more.

However, it can be expected that the arrival rates of neighboring energy level bins would be related. Anomalies, such as spikes, should then be perceivable in a series of neighboring bins.

By applying a smoothing procedure, which evens out the counts in the energy bins so that they follow a smooth curve, we might then be able to get values that are closer to the actual Poisson arrival rates. In addition, a variance-stabilizing transformation is applied before smoothing, and reversed after smoothing: for these Poisson variables, taking the square root transforms their distribution to a gaussian with a variance of exactly 0.25.

What Burr et al. claim, however, is that traditional smoothing methods such as kernel smoothing and cubic spline smoothing will often de-emphasize the peaks and valleys of a spectrum (understandably), while those peaks and valleys actually contain all of the information relevant to anomaly detection.

They propose, as a solution, to apply a multiplicative bias correction (MBC), where the ratio between the original spectra and the result of the first pass of smoothing is computed, then itself smoothed before being reapplied multiplicatively to the smooth spectra. This results in smooth spectra where the peaks and valleys have been re-emphasized.

The impact of cubic spline smoothing as well as the multiplicative bias correction are shown in Figure 1. The original spectrum is very noisy, and the cubic spline smoother produces a nice smooth curve, but the peaks and valleys could use more emphasis. This is what the MBC smoother attempts to correct.

The smoothing with multiplicative bias correction method by Burr et al was not used in the context of a machine learning

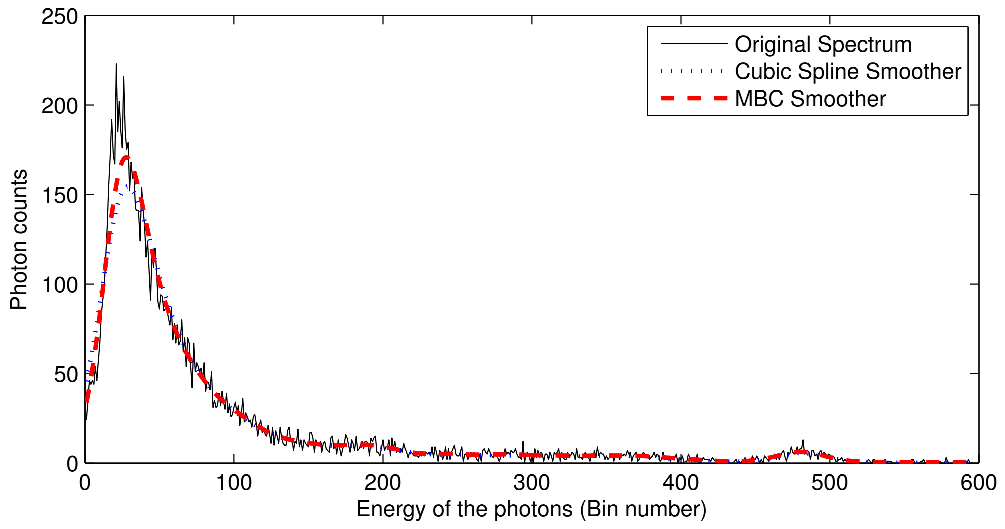


Fig. 1: Effect of cubic spline smoothing and the multiplicative bias correction on a 1-minute sample of data

system. As described in the following section, we propose to assess its impact on the performance of an anomaly detection system applied to the domain of gamma-ray data.

III. Proposed Method

The method we examine consists of an additional preprocessing step (smoothing with multiplicative bias correction) preceding the training of a classifier and testing of its performance. The gamma ray spectra data are submitted to the preprocessing process before being split into training and testing sets, or folds for cross-validation.

Obviously, once such a system is deployed, any new instances would have to be run through the same preprocessing steps before being fed to the classifier.

To each individual spectrum, we apply a two-pass smoothing procedure inspired by [2], as was briefly described in section II-C:

First, we fit a smooth curve to the spectrum. There is a variety of smoothers from which to choose, including kernel-based regression smoothers [15] such as the Nadaraya-Watson smoother [16], [17], wavelet smoothers, which have been used for sodium iodide spectra before [18], and smoothing splines [19]. In the paper by Burr et al. [2], cubic splines seemed to perform reasonably well, and they will constitute our choice of smoother in our experiments. Our MATLAB experimental environment offers an implementation of cubic smoothing splines through the `csaps` function.

Once smoothing splines have been fitted to a spectrum, we apply the multiplicative bias correction to re-emphasize the peaks and valleys. This is done by taking the ratio of the initial spectrum and its smoothing splines, applying a smoother to that ratio (once again, we used MATLAB's `csaps` cubic smoothing splines), then reapplying the smoothed ratio multiplicatively to the smooth spectrum.

The full procedure is illustrated in algorithm 1.

Algorithm 1 Smoothing preprocessing pseudocode

```
$firstPass \leftarrow cubicSplines(originalSpectrum)$
$ratios \leftarrow originalSpectrum \oslash firstPass$
$smoothRatios \leftarrow cubicSplines(ratios)$
$secondPass \leftarrow smoothRatios \odot firstPass$
```

IV. Experiments

In our experiments, we apply the smoothing procedure before training a classifier on the smoothed data, and compare the results to those obtained when using the data without smoothing. The following subsections detail the base classifiers that were used in these experiments, the datasets we used, and the methodology we followed to assess the results.

A. Classifiers used

1) Classifier based on the Mahalanobis distance: The first classifier that was used in conjunction with the smoothing preprocessing procedure was a parametric classifier based on the Mahalanobis measure of distance, as used in previous work on this particular domain [1].

During the training phase, the parameters of a multivariate gaussian distribution are estimated from a training set consisting of instances belonging to the background (negative) class only. This distribution is what the classifier assumes non-anomalous instances should look like.

Then, during the testing phase, we use the parameters previously estimated to compute the Mahalanobis distance of testing instances to the background class' distribution. This is a simple computation, and it consists of applying the following formula for the Mahalanobis distance:

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1} (x-\mu)}$$

Where μ and S are respectively the mean vector and covariance matrix of the training set.

This gives us a score of how different an instance is from the background class: an anomaly score. We can use these scores to identify anomalies by simply setting a threshold over the anomaly score, past which an instance is considered to be an anomaly.

$$prediction(x) = \begin{cases} anomalous & \text{if } score(x) \geq threshold \\ background & \text{if } score(x) < threshold \end{cases}$$

The particular threshold to be used can be determined by using a held-out set of instances, separate from both the training and test instances, and selecting the value that results in a desired rate of false or true positives. When selecting based on the number of false positives, for example, we would take all negative examples of our held-out set, sort them by their anomaly scores, and select as our threshold the score of the instance for which the desired portion of the set (false positive rate) is above that instance.

If we set the threshold for a particular false positive rate, then we must evaluate the classifier with respect to its sensitivity, whereas if we set the threshold with a specific true positive rate in mind, it is the specificity that will tell us how well the classifier did.

Another way to think of the anomaly scores produced by the Mahalanobis distance formula is as an ordering of the instances by how likely we estimate them to be anomalies. This ordering interpretation will be particularly useful when plotting ROC curves [20] for these classifiers, as described in Section IV-C.

2) One-Class SVM: As popular as support vector machines [21] are for binary classification problems, there are similar one-class learning algorithms that have proven to be quite useful for outlier detection problems [12], [22]. In these experiments, we use the ϵSVM method introduced in [12] by Schölkopf et al. This, once again, is in continuation of previous work on this domain [1].

As with regular support vector machines, ϵSVM makes use of a kernel function to map the data to a higher dimensional space, where it may easier to separate it with linear methods. A popular choice of kernel is the radial basis function.

The SVM algorithm defines a hyperplane with a maximum margin separating the two classes by using a series of training instances that lie on or within the margin. Those instances are called support vectors. The training of an SVM takes into account the tradeoff between the width of the margin (which is greater when the regularization parameters are larger) and the classification error.

The optimization problem for regular support vector machines, where the ξ_i's are terms quantifying the error made on each training instance, is formulated as such:

$$\begin{aligned} \underset{w,\xi,b}{\text{minimize}} \quad & \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \right\} \\ \text{subject to} \quad & y_i(w \cdot \phi(x_i) - b) \geq 1 - \xi_i \text{ for all } i = 1, \ldots, n, \\ & \xi_i \geq 0 \text{ for all } i = 1, \ldots, n. \end{aligned}$$

In the ϵSVM formulation, however, the class is no longer important, and we instead try to find a hyperplane that separates, in our high dimensional space, the instances belonging to the class being learned from the origin.

In this formulation, rather than the parameter C being used to set how smooth the hyperplane should be, we have a parameter ν which sets a lower bound on the number of support vectors and an upper bound on the fraction of outliers.

$$\begin{aligned} \underset{w,\xi,\rho}{\text{minimize}} \quad & \left\{ \frac{1}{2} \|w\|^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho \right\} \\ \text{subject to} \quad & (w \cdot \phi(x_i)) \geq \rho - \xi_i \text{ for all } i = 1, \ldots, n, \\ & \xi_i \geq 0 \text{ for all } i = 1, \ldots, n. \end{aligned}$$

We can then use the parameters w and ρ of the model to make predictions for new instances:

$$prediction(x) = \begin{cases} anomalous & \text{if } w \cdot \phi(x) \leq \rho \\ background & \text{if } w \cdot \phi(x) > \rho \end{cases}$$

If we wanted a ranking of the "anomaly score" of multiple instances, we could simply use $-(w \cdot \phi(x))$.

When using ϵSVM, there are two parameters that need to be finely tuned in order to obtain satisfactory performance. In our experiments, we found that for the ν and γ (gaussian kernel bandwidth) parameters of the LibSVM implementation, the values of 0.0001 and 0.001 (respectively) lead to the best results.

B. Datasets

For the purpose of evaluating the impact of the smoothing pre-processing method described in section III, we make use of a dataset given to us by the Radiation Protection Bureau of Health Canada.

Instances of the dataset correspond to samples from gamma-ray detectors that were collected over periods of 1 minute each. Each sample is associated to a timestamp, which can be used as identifiers when communicating with the Radiation Protection Bureau.

The data was collected over a period of approximately one year (from February 2010 to March 2011) in the city of Vancouver, Canada. The detectors collected gamma ray counts over about 600 channels, though domain experts confirmed that only the first 250 should carry information relevant to the task at hand. The other 350 were therefore not used.

Out of the 39023 data instances we have in this dataset, only 23 are anomalies, for a class skew of 1696 to 1. Of these 23 anomalies, we have some for 3 different potential sources of radiation, which is obviously not a sample representative of all anomalies that may surface. For this reason, the anomalies can be used for testing purposes, but using them for training would be of little help.

Another issue that comes up as a result of the small number of anomalies is that the resolution of our test results is limited.

There are only 23 different levels of sensitivity we can reach, and the distance between two subsequent such levels is already a significant gap in performance. This makes evaluation more difficult.

We limited our experiments to this domain only due to the applied nature of this research: our goal was to show that smoothing works for the specific domain of gamma-ray data.

C. Evaluation Methodology

The goal of the experiments was to determine whether or not the smoothing pre-processing procedure had an impact on the performance of the classifiers built upon the data, and, if it did, whether or not that impact was positive.

From the *Original* dataset, we generated two additional datasets: *Smoothed*, i.e. the result of a first pass of cubic spline smoothing on the original data, and *MBC*, i.e. the result of the multiplicative bias correction applied to the smoothed data.

For both of the two passes of smoothing, the parameter `p` of the `csaps` smoother in MATLAB was set to 1×10^{-6}. This value was determined to yield the best results when used before training a classifier on a set of negative instances held out of the later experiments. Given the small number of available anomalous instances for testing, those could unfortunately not be held out. We also noticed that using different parameters for the two passes of smoothing typically resulted in weaker classifier performance.

On these datasets, the two classifiers, Mahalanobis distance and ϵSVM, were used, though their results will be presented separately as the goal was not to compare the classifiers but instead to study the impact of the smoothing on each one of them. For ϵSVM, due its the extremely long training time, we sample one tenth of the dataset for each iteration in the experiment, reducing the training time to feasible levels. This may lead to higher variance in the results.

We then used 10x10-fold cross-validation to collect performance measurements for each of the two classifiers on each of the three datasets. The instances in each fold of each iteration were controlled to be the same across all of the conditions. This is important because it allows us to use a paired statistical test, which often has greater statistical power than non-paired tests.

The folds were defined over the background (negative) data, as the anomalies (positive data) would never be used for training purposes anyway. These anomalies were incorporated into the test set for each fold. In essence, the classifier's goal then becomes to, using the training background data, rank the testing background data so that it falls below this fixed set of anomalies in terms of anomaly scores. Each fold is used as the testing set once, and is part of the training set the nine other times out of ten.

After applying the trained classifier to the test set and obtaining an anomaly score for each of the testing instances, background and anomalous, we used the scores to plot an ROC curve. To do so, we use the scores given to each of the testing background instances as thresholds, and find the number of true and false positives each of these thresholds would give us. We then plot these true and false positive rate pairs as points in the ROC space.

It	*Original*	*Smoothed*	*MBC*
1	0.9572	0.9574	0.9762
2	0.9573	0.9573	0.9762
3	0.9573	0.9573	0.9763
4	0.9573	0.9574	0.9762
5	0.9573	0.9574	0.9762
6	0.9573	0.9574	0.9762
7	0.9573	0.9574	0.9762
8	0.9573	0.9574	0.9763
9	0.9573	0.9574	0.9762
10	0.9572	0.9574	0.9763
Avg.	0.9573	0.9574	0.9762
S.D.	3.5E-5	3.0E-5	3.0E-5

(a) Mahalanobis distance classifier

It	*Original*	*Smoothed*	*MBC*
1	0.8851	0.9854	0.9931
2	0.8940	0.9856	0.9912
3	0.8915	0.9755	0.9882
4	0.8326	0.9820	0.9912
5	0.8902	0.9791	0.9891
6	0.8949	0.9864	0.9934
7	0.8819	0.9783	0.9927
8	0.8967	0.9802	0.9894
9	0.8865	0.9699	0.9918
10	0.8918	0.9885	0.9895
Avg.	0.8845	0.9811	0.9910
S.D.	0.0188	0.0057	0.0018

(b) ϵSVM one-class classifier

TABLE I: AUROC over 10 iterations

The ROC curve is useful to gain insight into the levels of selectivity at which each method dominates, or whether a single method dominates no matter the selectivity of the threshold. However, since it is based on a single iteration of 10-fold cross-validation, we can't use its results to come to a strong conclusion on the relative strengths of the methods.

As the next step in our evaluation methodology, we compute the area under the ROC curve as a single metric of a classifier's performance over one of the 10 iterations of 10-fold cross-validation. It suffices to take the average of the true positive rates of the points of the ROC curve because these points are equally distant in terms of their coordinates on the false positive rate axis, resulting in bands of equal width when using the rectangle method to compute the area under the curve.

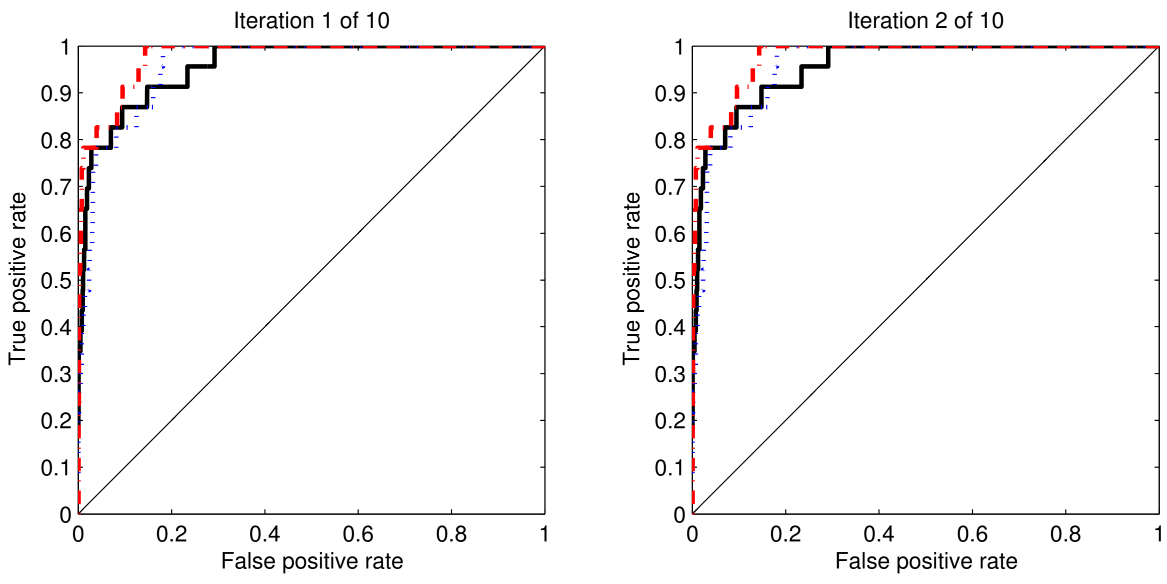
After the 10 iterations of 10-fold cross-validation, we therefore end up with 10 AUROC measurements for each condition (dataset and classifier). We can then use these measurements in a paired t-test to compare the results of the different methods. We will consider the results to be statistically significant if the p-value is lower than 0.01.

V. Results

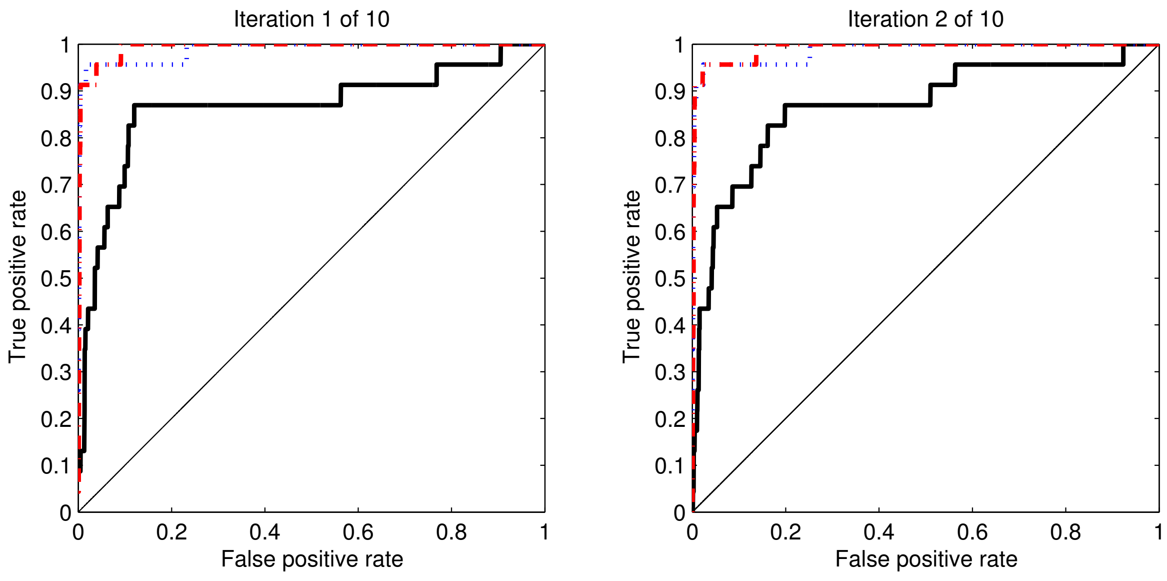
The results of the 10 iterations of 10-fold cross-validation, for the 3 versions of the dataset, are shown in Tables Ia and Ib respectively for the Mahalanobis distance classifier and ϵSVM one-class classifier.

In addition, figures 2a and 2b show the ROC curves for 2 of these iterations. The curves for the performance on the *Original*, *Smoothed* and *MBC* datasets are respectively in black/full, blue/dotted and red/dashed.

Given that the *MBC* dataset lead to the best results in all cases, we now want to support the hypothesis that the



(a) Mahalanobis distance classifier



(b) ϵSVM one-class classifier

Fig. 2: ROC Curves for 2 of the 10 iterations of 10-fold cross-validation.
Black/Full: Original, **Blue/Dotted:** Smoothed and **Red/Dashed:** MBC

performance on this dataset is better than the performance on the *Original* dataset with statistical testing results. We use Student's *t*-test with paired samples to compare the results on both. The t statistics for the differences between the two paired samples are $t_{Maha} = 1.2214 \times 10^3$ and $t_{SVM} = 1.7563 \times 10^1$.

Both of these test statistics are larger than the critical value for $p = 0.01$, i.e. 2.821. In fact, these test statistics are so extreme that we could reject the null hypothesis that there is no difference between the two sets of results at p-values in the order of 3×10^{-8}.

Another useful statistic is Cohen's d Effect Size [23], the difference between the means divided by the pooled standard deviation. According to Cohen [23], anything above 0.8 is considered a "large" effect size. In this case, the effect sizes are 575 and 7.97 respectively for the Mahalanobis distance and ϵSVM classifiers, which are both clearly "large" under that definition.

VI. Discussion

We notice two interesting results, which are discussed in the next subsections. First, smoothing had a quantitative impact on the performance of the classifiers as measured by the AUROC. Second, it had a qualitative impact on the kind of instance the models judge to be anomalous.

A. *Quantitative Impact of Smoothing*

As a first element of discussion, it is quite clear that the full smoothing with multiplicative bias correction pre-processing procedure brings performance benefits with both the ϵSVM and Mahalanobis distance classifier. This improvement is statistically significant in both cases, and the effect sizes are beyond what Cohen would have called "large". There is no denying that for this particular domain, the smoothing procedure has great benefits.

It is interesting, however, to look at the differences in how the different smoothing steps affect both classifiers.

First, in the case of the Mahalanobis distance classifier, it seems like the first smoothing step, the simple cubic smoothing splines, has extremely little impact on performance. It really is the multiplicative bias correction that makes all the difference. Perhaps this is because the multiplicative bias correction, by emphasizing the peaks and valleys of the spectra, forces the gaussian-by-nature classifier to use the "*right*" information.

Another important observation for the Mahalanobis distance classifier results is that the results all have very low variances. This is not particularly surprising, as with datasets of the size of the one used, we can expect the parameters of the gaussian distribution used by the classifier to be very stable from one fold to the other.

One might think that it is because it used the entire dataset, whereas the ϵSVM classifier only used a tenth, that the Mahalanobis distance classifier had such a lower variance. In a later experiment, the results of which are not documented here, we attempted to replicate the Mahalanobis distance classifier results using only one tenth of the dataset, as with ϵSVM. The variance of the AUROC did not change much, suggesting that the Mahalanobis distance simply produced more stable results naturally. This is not particularly surprising, as a single instance can make a significant difference for the non-parametric SVM classifier, if that instance is chosen to become a support vector, whereas all instances are given equal weight in the parametric Mahalanobis distance classifier.

Second, we see that the results of the ϵSVM classifier tell a very different story. In their case, the ROC curve plots clearly show that both smoothing passes, the initial cubic smoothing spline and the multiplicative bias correction, had a large impact on the classification performance.

More specifically, the first pass of smoothing seems to bring about a large portion of the improvements, but with varying levels of success on different folds. In some cases, the first pass alone brings us very close to the end, optimal result. In other cases, however, the first pass does not take us as far, and the multiplicative bias correction is what takes us all the way. In other words, the improvement effected by the first smoothing pass are, on average, large, but with a large variance. The improvements brought about by both smoothing passes combined, however, are more stable and bring us consistently to an AUROC close to 0.9910.

B. *Qualitative Impact of Smoothing Using PCA*

Principal component analysis can be a useful tool to study classifier performance. Placing the instances in a bidimensional plot based on their first two principal components has previously helped us identify incorrectly labeled instances, and it often helps us see how anomalies differ from normal data.

To further our analysis of the results described above, we plot the instances of the dataset along their first 2 principal components and color code them to identify different types of instances. The results are shown in Figure 3, where instances in green/crosses are true anomalies, and instances in red/circles are the top 20 most anomalous instances of the background class according to the different classifiers trained and tested on the different datasets. The remainder of the background class instances are shown as black dots.

As is obvious in the Mahalanobis distance plots, the multiplicative bias correction causes the type of instances the classifier judges to be anomalous to change qualitatively. The classifier used to select its anomalies from the top of the main cloud of background instances, and kept doing so despite the first pass of smoothing. When the multiplicative bias correction was introduced, however, it started picking its anomalies from the bottom of the cloud, close to a cluster of true anomalies.

A similar change is observed with the ϵSVM results. In this case, the first pass of smoothing helps to kickstart this transition, as we see the qualitative changes in the type of instances judged anomalous begins in the second graph.

The PCA plots give us some insight, but it is still not clear what the real impact is, as we don't know what the principal components actually represent. We hypothesize that the smoothing, by reducing the variance caused by the Poisson nature of the photon counts, forces the classifiers to consider another source of variance, one that actually gives us meaningful information as to the anomaly of an instance.

VII. Conclusion

In this project, we studied the impact of a smoothing pre-processing procedure on the performance of one-class classifiers in the context of a gamma ray spectral anomaly detection problem.

Overall, the results are very encouraging and show that on one particular domain the preprocessing procedure effects significant performance improvements as measured by the area under the ROC curve.

Future work may investigate whether this effect is as strong on data that is less noisy, such as gamma-ray detector samples collected over longer periods. It would also be interesting to look into the profile of instances that are better classified when smoothed, in the hope that it may give us further insight into how the smoothing helps.

Finally, gamma-ray spectra are not the only type of data where peaks and valleys are of vital importance to classification. Perhaps this smoothing method could be used to improve the performance of classifiers in other similar domains.

Acknowledgment

The authors would like to thank Health Canada, particularly Dr. Kurt Ungar and Dr. Rodney Berg, for their gracious support of this research.

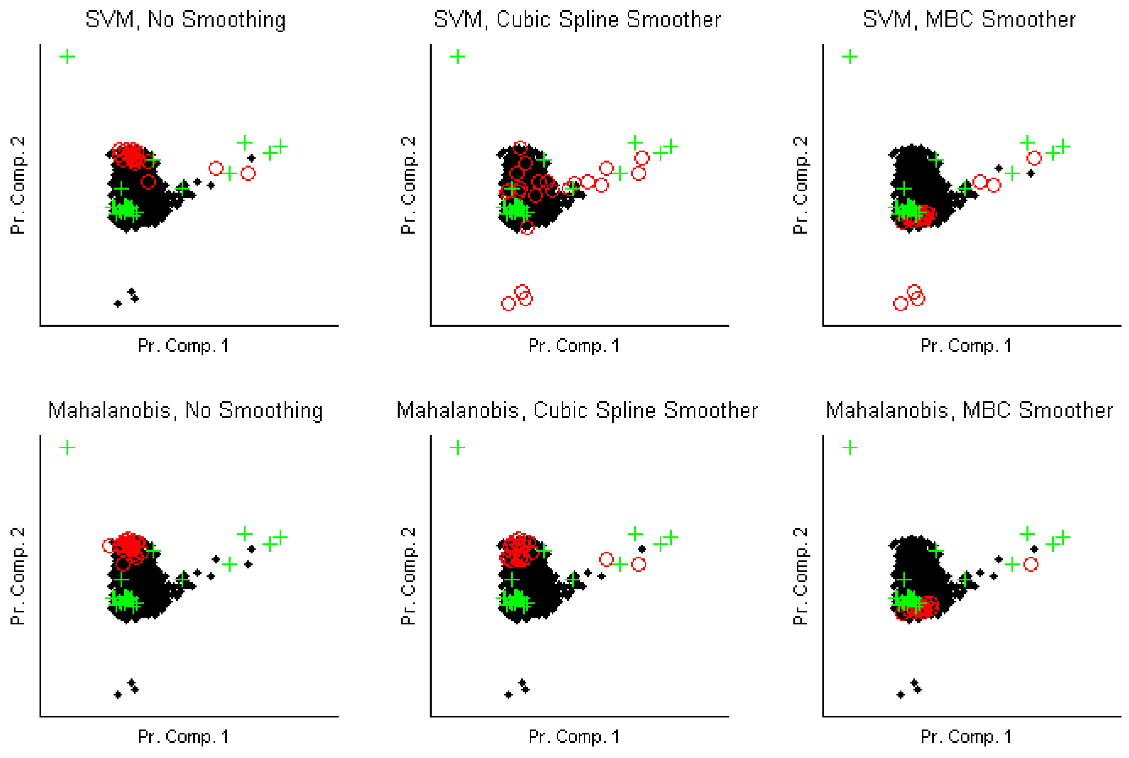


Fig. 3: PCA plots of the anomalies identified by each classifier.
Black/Dots: Background, **Green/Crosses:** True Anomalies, **Red/Circles:** Top False Anomalies

References

[1] S. Sharma, C. Bellinger, N. Japkowicz, R. Berg, and K. Ungar, "Anomaly detection in gamma ray spectra: A machine learning perspective," in *Computational Intelligence for Security and Defence Applications (CISDA), 2012 IEEE Symposium on.* IEEE, 2012, pp. 1–8.

[2] T. Burr, N. Hengartner, E. Matzner-Lober, S. Myers, and L. Rouviere, "Smoothing low resolution gamma spectra," *Nuclear Science, IEEE Transactions on*, vol. 57, no. 5, pp. 2831–2840, 2010.

[3] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[4] A. Arning, R. Agrawal, and P. Raghavan, "A linear method for deviation detection in large databases." in *KDD*, 1996, pp. 164–169.

[5] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[6] E. M. Knorr and R. T. Ng, "A unified approach for mining outliers," in *Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 1997, p. 11.

[7] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[8] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski, "Clustering approaches for anomaly based intrusion detection," *Proceedings of intelligent engineering systems through artificial neural networks*, pp. 579–584, 2002.

[9] M. Li and P. M. Vitányi, *An introduction to Kolmogorov complexity and its applications*. Springer, 2009.

[10] G. M. Weiss, "Mining with rarity: a unifying framework," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7–19, 2004.

[11] G. M. Weiss and H. Hirsh, "The problem with noise and small disjuncts," in *ICML*, 1998, p. 574.

[12] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[13] N. Japkowicz, C. Myers, M. Gluck *et al.*, "A novelty detection approach to classification," in *IJCAI*, 1995, pp. 518–523.

[14] M. Denil and T. Trappenberg, "Overlap versus imbalance," in *Advances in Artificial Intelligence*. Springer, 2010, pp. 220–231.

[15] M. P. Wand and M. C. Jones, *Kernel smoothing*. Crc Press, 1994, vol. 60.

[16] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[17] G. S. Watson, "Smooth regression analysis," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

[18] C. Sullivan, M. Martinez, and S. Garner, "Wavelet analysis of sodium iodide spectra," in *Nuclear Science Symposium Conference Record, 2005 IEEE*, vol. 1. IEEE, 2005, pp. 302–306.

[19] C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[20] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[23] J. Cohen, *Statistical power analysis for the behavioral sciences (rev.* Lawrence Erlbaum Associates, Inc, 1977.