"Boxing Clever": Practical Techniques for Gaining Insights into Training Data and Monitoring Distribution Shift

Rob Ashmore(✉) and Matthew Hill

Defence Science and Technology Laboratory, Fareham, Hants PO17 6AD, UK
{rdashmore,mhill2}@dstl.gov.uk

Abstract. Training data has a significant influence on the behaviour of an artificial intelligence algorithm developed using machine learning techniques. Consequently, any argument that the trained algorithm is, in some way, fit for purpose ought to include consideration of data as an entity in its own right. We describe some simple techniques that can provide domain experts and algorithm developers with insights into training data and which can be implemented without specialist computer hardware. Specifically, we consider sampling density, test case generation and monitoring for distribution shift. The techniques are illustrated using example data sets from the University of California, Irvine, Machine Learning repository.

Keywords: Artificial intelligence · Machine learning
Training data · Distribution shift · Test cases

1 Introduction

1.1 Background

Over recent years, great progress has been made in Machine Learning (ML) and a vast amount of academic literature has been written, including: explaining behaviour, [12]; ways of producing adversarial examples, [10,14]; ways of defending against these examples, [6,7]; and new types of algorithm, [13,15].

Whilst much research focuses on ML algorithms, training data is also important, not least because it encodes requirements that the algorithm should satisfy. In addition, small changes to a data set can introduce a backdoor [5] and adversarial examples can transfer across networks trained on the same data [11]. Adversarial examples may also be a natural consequence of high-dimensional input domains [3]; that is, the input domain could itself be significant. For these reasons we expect training data to be explicitly considered in any safety, or assurance, argument related to an ML algorithm. This paper outlines simple techniques for gaining insight into training data sets, which could support such considerations. Our focus is deliberately on low complexity techniques that do not rely on specialist computational hardware.

B. Gallina et al. (Eds.): SAFECOMP 2018 Workshops, LNCS 11094, pp. 393–405, 2018.
https://doi.org/10.1007/978-3-319-99229-7_33

1.2 Terminology and Notation

The data used to train an ML algorithm is referred to as a *training data set*, or simply a *data set*, and is denoted T. We have no need to separately distinguish data that is withheld in order to test an algorithm after a training epoch.

A training data set is made up of a number of *samples*, denoted s^i. Equivalently, $T = \{s^i : i = 1, \ldots, n\}$, where n is the number of samples.

Each sample comprises a number of *features*, with the jth feature of sample i being denoted by s^i_j. The collection of features define the algorithm's *input domain*. To simplify the discussion, we focus on quantified features; that is, features that take numerical values from a real-valued, continuously-defined scale. Equivalently, $s^i \in \mathbb{R}^d$, where d is the number of features.

For convenience, and without loss of generality, we assume samples have been scaled so that they are contained in the relevant d-dimensional unit hypercube. We have adopted a linear scaling: using o to denote the original, pre-scaling samples, scaling is achieved as follows:

$$max_j = \max_i o^i_j, \qquad min_j = \min_i o^i_j, \qquad s^i_j = \frac{o^i_j - min_j}{max_j - min_j}$$

1.3 Example Data Sets

The techniques are illustrated using three example data sets obtained from the University of California, Irvine (UCI) Machine Learning repository. Key properties are summarised in Table 1.

Table 1. Summary properties of example data sets

Name	Samples	Features	Reference http://archive.ics.uci.edu/ml/datasets/
Iris	150	4	Iris
Pen Digits	10,992	16	Pen-Based+Recognition+of+Handwritten+Digits
Cover Type	581,012	10	Covertype

The UCI repository's description of Cover Type states there are 54 features. We do not consider the 44 features that are binary flags, so our version contains 10 quantified features. Note that the original data set is © Jock A. Blackard and Colorado State University.

Since all samples have been scaled to the d-dimensional unit hypercube, for our purposes a sample from the Iris data set comprises four real numbers, each in the range $[0, 1]$. Similarly, the Pen Digits and Cover Type samples comprise sixteen and ten such numbers, respectively.

In terms of size, the data sets described in Table 1 are comparable with other data sets we are using in our work. They are, however, notably smaller than

many used in modern applications. More particularly, they are smaller both in terms of the number of features and, also, the number of samples.

Since they are easy to parallelise, the techniques described in this paper are easy to extend to data sets that contain more samples, albeit at the possible expense of having to use specialist hardware. Extending to data sets with a larger number of features is more complicated. Firstly, the techniques implicitly assume there is no connection between the different features; this is not the case, for example, when each sample is an image and each feature is a pixel. Secondly, the geometric nature of the techniques means they may be less informative in very high dimensional spaces.

Whilst application of the techniques to larger data sets is future work, we note the easiest way to apply them to a large, image-based data set may be to create a representation of that data set on a reduced-dimension manifold, the results of which may also be valuable for other reasons.

1.4 Structure

The remainder of this paper is structured as follows: Sect. 2 introduces the concept of an axis-aligned box, which underpins our techniques; Sects. 3 and 4 describe techniques for understanding sampling density; Sect. 5 discusses a way of creating novel test sets; Sect. 6 outlines a way of monitoring for distribution shift; Sect. 7 provides conclusions.

2 Axis-Aligned Boxes

The techniques developed in this paper make extensive use of axis-aligned boxes. A box is defined by providing lower and upper bounds for each feature, denoted lo_j and hi_j. A sample, s^i is within this box if $lo_j < s^i_j < hi_j$ for $j = 1, \ldots, d$. Note, the sample is in the box if it is strictly within the bounds.

Scaling the data set so all samples fall strictly within the d-dimensional unit hypercube implicitly introduces a distance metric. Using axis-aligned boxes to decide whether two samples are, in some sense, close to one another is consistent with the notion of Manhattan Distance [2]. Axis-aligned boxes are ideal for our purposes: determining whether a particular sample lies within a given box needs at most $2d$ comparisons, which can be computed very efficiently.

3 Sample Density

3.1 Approach

Generally speaking, samples in training data sets are observations from some external process. It is typically not possible to gather samples against a specific type of experimental design. This means that the distribution of samples across the input domain is often uncontrolled and may not be well understood.

One way of understanding this distribution involves calculating the distance between each sample and its nearest neighbour and looking at the ratios of these distances [1]. However, the associated distance calculations can be inefficient for large data sets. An alternative approach simply involves counting the number of points in a suitably-sized and located axis-aligned box.

In terms of box size, this is typically expressed in terms of the number of samples that would be expected to be found in the box if samples were uniformly distributed. More specifically, suppose we have n samples uniformly distributed within a d-dimensional unit hypercube. In this case, an axis-aligned box with sides of length l would be expected to contain x samples, where:

$$l = \exp\left(\frac{\ln(x/n)}{d}\right) \quad \Rightarrow \quad l^d = x/n$$

Table 2 shows values of l for our illustrative data sets. It is apparent that in all cases these values cover a reasonable fraction of the $[0, 1]$ range that each (scaled) feature covers. Table 2 also shows that as d increases, so does l; this is one challenge that arises when applying the techniques to high dimensional data sets.

Table 2. Length of sides so hypercube would be expected to contain 1 or 10 samples

Name	Expected (x)	Samples (n)	Features (d)	Side (l)
Iris	1	150	4	0.286
Pen Digits	1	10,992	16	0.559
Cover Type	1	581,012	10	0.265
Iris	10	150	4	0.508
Pen Digits	10	10,992	16	0.646
Cover Type	10	581,012	10	0.334

Intuitively, it would be preferable to count the total number of samples in an axis-aligned box centred on each sample[1]. However, given the sizes indicated by Table 2, a significant portion of these boxes could fall outside the unit hypercube (where our scaling means there are no sample points). Hence, each box is located by first centering it on the sample point of interest and then, if necessary, translating so it is entirely within the unit hypercube.

The number of samples in each of these boxes is an approximate measure of sampling density: boxes that contain large numbers of samples represent densely sampled regions; boxes with a low number of samples represent sparsely sampled regions. Consequently, the distribution of the total number of samples in each of these boxes is informative. It is also helpful to consider two other factors.

The first factor is the ratio between the 1st and 99th percentiles of this distribution. This indicates the ratio of sampling densities between the most densely and most sparsely sampled regions. There are often good reasons why

[1] For very large data sets, it is sufficient to consider a suitable number of randomly chosen samples.

sampling density should vary across the input domain; indeed, some sequential experimental design techniques are designed to target "more interesting" parts of the input domain [4]. As a minimum, extreme differences in sampling densities should be explained and justified. Whilst there are no hard and fast rules, our empirical work to date suggests that ratios of less than 100 may be judged reasonable, whilst those greater than 1000 may need special justification.

The second factor is whether any samples are outliers. In this context a sample would be an outlier if the associated suitably-sized, axis-aligned box contained no other samples. Outliers are of interest for several reasons: our experience shows they provide domain experts with insights into the training data set; they can have a disproportionate effect on the behaviour of the trained algorithm; and they can reveal errors in data collection and preparation.

3.2 Example Data Sets

Figure 1 illustrates the distribution of the number of data samples in axis-aligned boxes for cases where the box is sized so it would contain either 1 or 10 samples, if the samples were uniformly distributed. For the Iris and Pen Digits data sets, this distribution is based on all samples; due to the large number of samples in the Cover Type data set, this case is based on 100,000 randomly chosen samples. Table 3 summarises some characteristics of these distributions (for brevity, this only considers the case where the box is scaled for 10 samples).

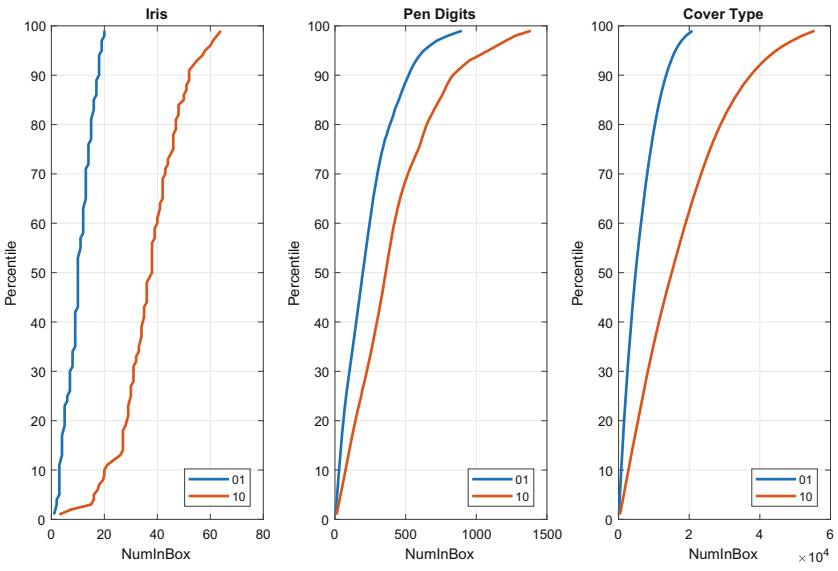

Fig. 1. Distribution of number of samples in axis-aligned boxes (scaled for 1 and 10 samples, assuming samples are uniformly distributed)

Table 3. Characteristics of the distribution of number of samples in an axis-aligned box (scaled for 10 samples, assuming samples are uniformly distributed)

Characteristic	Iris	Pen Digits	Cover Type
99th/1st %ile Ratio	21.3	115.5	148.7
Number of single-sample boxes	0	1	5
Maximum number of samples in a box	66	1,624	65,939

Figure 1 shows there are many more samples in each box than would be expected if the samples were uniformly distributed. For these three data sets, this effect becomes more pronounced as the number of samples in a data set increases. The same trend is shown in Table 3.

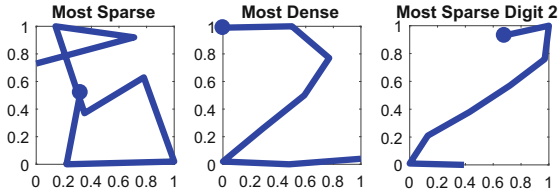

Fig. 2. Pen Digits samples from the most sparse (left), most dense (centre) and most sparse digit 2 (right) regions

The Pen Digits training data is in the form of eight pairs of (x, y) coordinates. Plotting these gives a graphical illustration of the sample. Figure 2 illustrates the samples associated with the smallest (left-hand) and largest (central) number of samples in the associated axis-aligned box (scaled for 10 samples); the circle marks the starting point, that is, (X1, Y1). Equivalently, these are samples from the most sparsely (left) and most densely (centre) sampled parts of the input domain.

These samples make intuitive sense: although the left-hand plot could be illustrative of a 6, or perhaps an 8, it does not follow a traditional pattern for writing either digit; conversely, the central plot clearly illustrates a typical way of writing 2. More generally, we have found that samples from the most sparse and most dense regions are informative for domain specialists.

If the data set is labelled with class information then repeating this analysis solely for samples within a single class can provide additional insights. For example, it can identify "modal" (most dense) and outlier (most sparse) samples for each specific class. As an illustration, the right-hand plot in Fig. 2 shows the sample that is in the most sparse region of digit 2s in the training data.

4 Empty Hyper-Rectangles

4.1 Approach

The previous section considered parts of the input domain where the training data set contains samples. Another perspective is provided by considering those parts of the input domain where there are no samples[2]. Equivalently, finding Empty Hyper-Rectangles (EHRs) within the training data set, is interesting.

Our approach to finding large, axis-aligned, EHRs is taken from Lemley, *et al.* [8]. The method starts from a randomly chosen point and expands from there. Observe that a random start point is more likely to be in a large empty space than it is to be in a small one. This observation provides some confidence that, despite its random nature, the method should provide reasonable performance; that is, it should identify large empty regions in the data set (should any exist).

Having started from a point, the method expands in each dimension until it hits any sample projected onto that single dimension. This is illustrated in the left-hand plot in Fig. 3: in this figure the randomly chosen start point is illustrated with a red X; samples in the data set are illustrated with red circles; and the EHR is shown as a blue rectangle. After this initial expansion, the empty hyper-rectangle is expanded in all dimensions simultaneously. Expansion in a given dimension stops when a sample point is reached. For example, the central plot in Fig. 3 shows the situation when no further expansion is possible in either the negative x direction or the positive y direction. The right-hand plot shows the final situation, when no further expansion is possible in any direction.

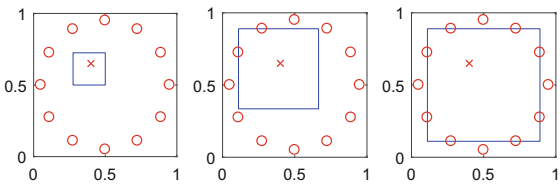

Fig. 3. Steps in finding an empty hyper-rectangle (Color figure online)

This process is repeated for different randomly chosen starting points. A variety of approaches may be taken to decide how many repetitions should be conducted. The simplest involves choosing a fixed value. An alternative involves considering a minimum of repetitions and then stopping if the last r repetitions has not increased the size of the largest EHR found so far, for a suitable value of r.

[2] The previous section showed that areas of the input domain contain more samples than would be the case if samples were uniformly distributed. A corollary is that there must be relatively large regions of the input domain that contain no samples.

4.2 Example Data Sets

Figure 4 shows EHRs for the Iris (top plot) and Cover Type (bottom plot) data sets. Each feature in the data set is illustrated by a separate column. For example, the top plot shows a region that covers the range $(0.0, 1.0)$ for the first two features, the range $(0.15, 1.0)$ for the third feature and the range $(0.0, 0.38)$ for the fourth feature. More loosely, this data set contains no samples where the third feature is large and the fourth feature is small. Our experience shows insights like this are valuable to domain experts.

Care needs to be taken when interpreting plots like Fig. 4. The outlined regions provide a useful illustration of the *shape* of the empty region, but it does not provide a useful guide to its *size*. To help address this difficulty, the plots include a final column, drawn in red and labelled as (Size). This shows the fraction of the unit hypercube that is covered by the empty region; this information is also shown in the plots' titles.

A full discussion of the regions shown in Fig. 4 is beyond the scope of this paper. However, note that both regions cover more than 25% of the respective unit hypercube and, as such, should provide domain experts with valuable insights into the respective data sets.

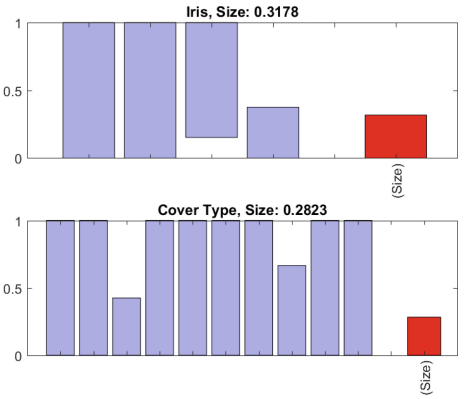

Fig. 4. Largest EHRs found in the Iris and Cover Type data sets

The Pen Digits data set has been excluded from Fig. 4 as the identified EHR covers the entire interior of the unit hypercube. Recall that a sample is only inside a region if it is *strictly* within the bounds established for each feature. The scaling used when building the Pen Digits data means each sample has at least one x coordinate that is 0 and one x coordinate that is 100; the same also holds for the y coordinates. Consequently, every sample falls on a boundary of the unit hypercube and hence the EHR covers the whole interior of the hypercube.

It is reassuring that the algorithm correctly identifies the EHR that covers the entire unit hypercube, but it is not especially interesting. Despite this, the

notion of EHRs can still be informative. As an example, we consider the first and last pairs of features, that is, (X1, Y1), which marks the start of the digit, and (X8, Y8), which marks the end.

The left-hand plot in Fig. 5 shows the EHR for these four dimensions. In the right-hand plot, the blue dashed rectangle shows the (X1, Y1) part of the region; the red dotted rectangle shows the (X8, Y8) part of the region. The EHR is such that there are no digits that start in the blue rectangle and finish in the red one. Our experience indicates these types of observation are informative to domain specialists.

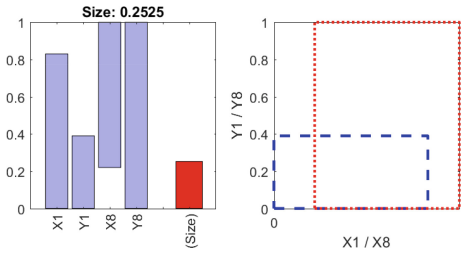

Fig. 5. Largest EHR found in the Pen Digits (X1, Y1, X8, Y8) data set (Color figure online)

5 Single-Class Regions

5.1 Approach

The techniques discussed in the preceding sections are applicable to all types of ML application (including, for example, classification and regression) and also to both supervised and unsupervised learning. In the specific case of supervised learning for classification applications, they can be extended to provide additional insights.

Consider the algorithm outlined below:

1. Select a class, C, from the training data, T.
2. Randomly pick a sample from this class, s^i.
3. Create a (temporary) set of training data, T', by removing all samples of class C from T. So, $T' = T \setminus \{s^i : s^i \in C\}$.
4. Find an EHR within T', using starting points drawn from $\{s^i : s^i \in C\}$; that is the starting points are the class C points within T.

Any region found by this algorithm will satisfy two properties: firstly, given the possible starting points, the region will contain at least one sample from C; secondly, since the region contains no samples from T', any samples it contains (from T) must be of class C.

Hence, this algorithm finds axis-aligned boxes that only contain samples from a single class. These regions can provide a domain expert with additional insights

into the training data. They can also inform testing of the model created from the training data: for example, if the model predicted many non-class C results from test inputs drawn from a "class C only" region then this should raise concerns.

For practical applications, it is helpful to add another step to the algorithm. Note, firstly, that the algorithm is based on finding *empty* regions and, secondly (as shown earlier) that training data sets may have relatively large regions where there are *no* samples. This means that the identified regions may contain relatively large spaces where there are no samples. More particularly, if the above algorithm is used to create single-class regions for more than one class then there may be a significant degree of overlap between these regions. Given the intended uses of the single-class regions, it is helpful to try and remove some of this overlap.

This can be achieved by increasing the lower bound of each dimension of the region so that it is equal to the lowest value of any sample (from T) in the empty (T') region. Similarly, the upper bound is reduced so that it is equal to the highest value of any sample. This is simple to implement and, consequently, consistent with our aims, but it is not guaranteed to remove all overlap between single-class regions for different classes.

5.2 Example Data Sets

For brevity, only results from the Pen Digits data set are shown. Figure 6 shows single-class regions for Class 5 (top) and Class 6 (bottom); for these plots the (Size) column reflects the fraction of Class 5 (or Class 6) samples that are in the region. The Class 5 plot contains over 34% of the 5s that are in the data set; the region in the Class 6 plot contains over 57% of the 6s. Hence, these regions are significant in terms of understanding the behaviour of any algorithm trained on the Pen Digits data set.

Despite their importance, each region covers less than 0.5% of the unit hypercube (reported as 0.00% in Fig. 6). Hence, without the type of analysis discussed here, it is unlikely that either would be explored in any detail.

6 Monitoring Distribution Shift

6.1 Approach

Strictly speaking, distribution shift is not a property of training data; it represents the relationship between training data and data observed during operational use. The topic is considered in this paper because the techniques discussed above can be easily extended to provide a perspective on distribution shift and, furthermore, this perspective is consistent with our general theme of limited computational resources.

Specifically, we are interested in monitoring for covariate shift [9], which occurs when the distribution of inputs during operational use is different from the distribution exhibited by the training data set. Our method considers two perspectives: that of individual points; and that of collections of points.

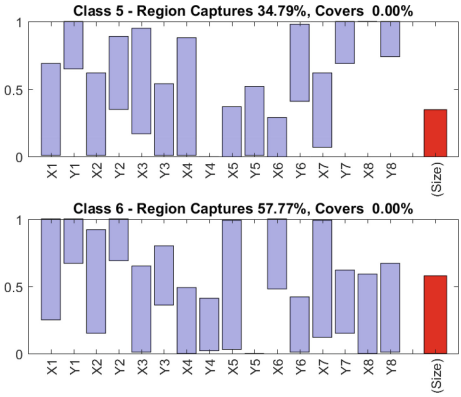

Fig. 6. Single-class regions for Class 5 and Class 6 in the Pen Digits data set

The individual point considerations are straightforward. Any operational inputs that (scaled using the same factors as for the training data set) fall outside the d-dimensional unit hypercube also fall outside the bounds of the training data set and, as such, may be indicative of distribution shift. Likewise, any (scaled) operational inputs that fall inside an EHR may also be indicative of a distribution shift. In both of these cases, some leeway is advisable: a single point that is only just outside the unit hypercube, or only just inside an EHR, is unlikely to represent a significant distribution shift.

In order to consider collections of points, some further analysis is needed. In particular, we find a series of expanding, axis-aligned boxes that contain given fractions of the training data. That is, we want to find $A_0 \subset A_1 \subset \ldots \subset A_m$ where $|\{s^i : s^i \in A_i\}|/|T| \approx f_i$, with $f_0 < f_1 < \ldots < f_m$.

A suitable collection of boxes can be identified by choosing a starting point and expanding equally in all dimensions until either the edge of the d-dimensional unit hypercube is reached, or the desired fraction of samples is contained in the box. Essentially, the choice of starting point is arbitrary. We have achieved good results by picking starting points in densely sampled parts of the input domain.

Note that, due to the restriction that we use axis-aligned boxes, it might not be possible to precisely achieve the desired fractions. We typically use 0.2, 0.4, 0.6 and 0.8 as our desired values.

Once we have the bounding boxes, we count the number of operational inputs that fall into each one. By considering, respectively, the number of operational inputs and the number of training data samples that are in A_m but not in A_{m-1}, and so on, we can create a contingency table for a Chi-squared test. This can be used to determine the probability that the spatial distributions of the training data and operational inputs (as summarised by the series of bounding boxes) are different. By using a relatively small number of axis-aligned, bounding boxes this approach discards a large amount of potentially useful information. Nevertheless, it is simple to implement, even on limited computational hardware.

Hence, it may be useful either in its own right, or as a trigger to prompt a more detailed, and more time consuming, calculation.

6.2 Example Data Sets

For reasons of brevity, only limited information is provided on applying the above process to the example data sets. In each case, results were as expected. For example, test sets developed by randomly choosing samples from the training data set did not routinely exhibit distribution shift. Conversely, test sets developed by choosing individual feature values from the training data set, or by choosing feature values at random, routinely exhibited distribution shift.

We acknowledge that these types of test are extremely simple. Nevertheless, they illustrate there is merit in the approach discussed above.

7 Conclusions

Although they inevitably introduce some inaccuracy, axis-aligned boxes are a helpful concept for understanding training data sets, especially in situations where computational resources are limited. They can be used to understand sampling density, including the balance between densely and sparsely sampled regions, and to identify regions that contain no samples. They can also inform test generation, for example by identifying regions that contain samples from a single class, and support simple ways of monitoring for distribution shift between training and operational use.

Acknowledgements. The authors would like to acknowledge the helpful feedback of the anonymous reviewers, which significantly improved this paper.

References

1. Bishnu, A., Desai, S., Ghosh, A., Goswami, M., Subhabrata, P.: Uniformity of point samples in metric spaces using gap ratio. In: Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation, pp. 347–358 (2015)
2. Black, P.E.: Manhattan distance. Dict. Algorithm. Data Struct. **18**, 2012 (2006)
3. Gilmer, J., Metz, L., Faghri, F., Schoenholz, S.S., Raghu, M., Wattenberg, M., Goodfellow, I.: Adversarial spheres (2018). arXiv:1801.02774v2
4. Gramacy, R.B., Lee, H.K.H.: Bayesian treed gaussian process models with an application to computer modeling. J. Am. Stat. Assoc. **103**(483), 1119–1130 (2008)
5. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: identifying vulnerabilities in the machine learning model supply chain (2017). arXiv:1708.06733
6. Guo, C., Rana, M., Cisse, M., van der Maaten, L.: Countering adversarial images using input transformations (2017). arXiv:1711.00117
7. Kolter, J.Z., Wong, E.: Provable defenses against adversarial examples via the convex outer adversarial polytope (2017). arXiv:1711.00851
8. Lemley, J., Jagodzinski, F., Andonie, R.: Big holes in big data: a Monte Carlo algorithm for detecting large hyper-rectangles in high dimensional data. In: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 563–571. IEEE (2016)
9. Moreno-Torres, J.G., Raeder, T., Alaiz-RodriGuez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recogn. **45**(1), 521–530 (2012)
10. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition, pp. 427–436 (2015)
11. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519. ACM (2017)
12. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should I trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)
13. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems, pp. 3857–3867 (2017)
14. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: Proceedings of the 2nd International Conference on Learning Representations, pp. 1–10 (2014)
15. Zhou, Z.H., Feng, J.: Deep forest: towards an alternative to deep neural networks (2017). arXiv:1702.08835