SAND2017-12370C

PERFORMANCE BENCHMARKS FOR DETECTION PROBLEMS

K. Larson and M. Boutin

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN, 47907

ABSTRACT

A benchmark curve that measures the inherent complexity of a detection problem is proposed. It is built using a sequence of simple detection methods based upon random projection. The benchmark curve is parameterized by the area above the receiver-operating characteristic curve of the detection method and its computational cost. It divides the plane into regions that can be used to characterize the computational and structural advantages of a given detection method. Numerical illustrations are provided.

***Index Terms*—** Benchmark, classification, detection, random projection, receiver-operating characteristic

1. INTRODUCTION

Some detection problems are easier to solve than others. For example, it has been observed that some image classification and detection problems have such a simple and obvious structure that a solution involving mere random projections has near-optimal results [1]. Information has also been shown to be preserved well under random projections [2, 3]. We propose random projections to develop simple detection methods - ones whose performances are near-optimal when the structure of the problem is as simple as possible - to serve as references and help judge the performance of other detection methods. The performance of our and other detection methods is qualified by the receiver-operating characteristic (ROC) curve along with the respective computational cost.

Our proposed detection methods are ordered in sequence of increasing computational cost. There is generally a trade-off between detection ability and computational cost. To account for this trade-off, we build a benchmark curve in a 2-D plane defined by the area above the ROC curve (AAC) for a given method and its associated compuational cost. Our benchmark curve in this plane is the piecewise linear interpolation of the points corresponding to each detection method in our sequence; moving from one method to the next in sequence, the AAC (the complement of the more common statistic AUC) decreases as detection accuracy improves and computational cost increases (i.e. increasingly costly ROC references have decreasing AACs). The resulting monotonic curve is used as a benchmark against which any detection method capable of producing a ROC curve can be compared.

We developed the first detection method in our sequence by modifying the TARP approach [4] to form one ROC reference curve; the computational cost and the AAC of that ROC curve define the first point of our proposed benchmark curve. We obtain the next points by using the modified TARP approach r times, where $r > 1$, and choosing the best resulting ROC curve as the reference curve. The AAC and the computational cost (CC) of the "best among r" ROC curves defines point number r in the benchmark plane. The benchmark curve is defined as the linear interpolation of points $1, 2, 3, ...$

Our proposed benchmark curve divides the plane into several regions; other detection methods are represented by points in the AAC-CC plane and can be characterized by the region in which they lie. The positive gain region contains methods which outperform the benchmark and the negative gain region contains methods which underperform the benchmark. Methods which lie on or near to the benchmark curve have zero gain and perform comparably to the benchmark. The positive gain region is further divided into a structural gain region, which contains methods that have a significant modeling advantage over the benchmark, and a computational gain region, which contains methods that have only a computational advantage over the benchmark. Because the benchmark is constructed from a series of random projections, a method which has negative gain is considered ill-suited for the given problem; the framework a negative gain method relies on yields a worse performance than no framework (random projections) at all.

After introducing the benchmark curve in Section 2, we develop a procedure for estimating it from labeled data in Section 3. In Section 4, we present numerical experiments with real and synthetic data that illustrate cases where SVM detection algorithms fall into each of our proposed benchmark regions. Finally, in Section 5, we summarize our findings.

2. PROPOSED BENCHMARK

We propose the use of random projections to generate a benchmark from the ROC curves of detection problems. Our benchmark builds on the concept of "Thresholding After Random Projection," or "TARP," first introduced in [4] from the perspective of classification problems. We extend this concept to the more general problem of detection.

TARP originally involved a random projection of a signal (e.g. image features) $x \in \mathbb{R}^p$, into one dimension, followed by a simple threshold classification. In our proposed detection framework, however, the threshold is varied and instead of classifying the projection, the projected data is used to build a ROC curve. The ROC curve itself is random because it is constructed from a random projection. We use the expected ROC curve as our first reference; successful detection methods will have ROC curves which lie above this reference and unsuccessful methods will have ROC curves which lie below this reference. The detection capability of just one random projection is not necessarily high, in general, so use of this ROC reference curve is limited.

Thanks to Sandia National Laboratories and the National Physical Science Consortium for support.

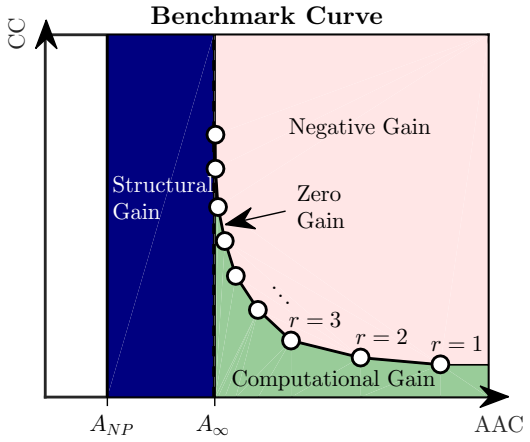

Fig. 1: **Proposed benchmark curve in AAC-CC space.** The benchmark points for $r = 1, 2, 3, ...$ are interpolated to form a curve. The benchmark curve converges to an asymptote (A_∞) at or above the optimal, N-P AAC (A_{NP}). The curve carves regions into the AAC-CC space; the outputs of other detection algorithms can be represented by points that lie either on the benchmark curve (Zero Gain) or in one of the labeled regions (Negative, Computational, or Structural Gain).

To improve TARP detection ability, we project the features, x, onto r random vectors, $v_1, ..., v_r \in \mathbb{R}^p$, where $r > 1$. We produce ROC curves for each of the r random projections, then select the "best" ROC curve: the ROC curve with the lowest AAC. The AAC is the area above the ROC curve and below true positive rate (TPR) $= 1$. This best ROC curve is random, so its expectation is another (improved) reference.

The AAC of the expected best ROC curve among r, denoted A_r, tends to decrease as we increase r. The best possible performance of a detection algorithm is that of the Neyman-Pearson (N-P) test performed with full knowledge of the underlying class probability densities. The AAC of the N-P test, denoted A_{NP}, is the minimum possible AAC for the given problem and is a lower bound on the benchmark curve AAC. We thus expect the $\{A_r\}_{r=1}^{\infty}$ to converge to some limit A_∞, with $A_\infty \geq A_{NP}$.

There is also an increase in the computational cost (CC) required to find the best ROC curve as we increase the value of r; each additional random projection adds the computational cost of building an additional ROC curve. By varying r, we produce a series of points in the AAC-CC plane. The linear interpolation of these points makes up our proposed benchmark curve, as illustrated in Figure 1.

The location of the N-P AAC and the location and shape of the benchmark curve will vary depending upon the problem. If the detection classes are perfectly linearly separable, then $A_{NP} = 0$ and the benchmark curve will converge to zero. If the classes are not perfectly linearly separable, the benchmark curve will converge to an AAC greater than zero. In any problem, the benchmark curve may converge to a value greater than or equal to A_{NP}.

Figure 1 also illustrates gain regions in AAC-CC space. If a detection algorithm allows for the creation of a ROC curve (i.e. a parameter may be altered to change the resulting TPR and false positive rate, or FPR), then its output may be converted to a point in AAC-CC space. Where the point falls on the AAC-CC plane with regard to the benchmark curve will identify the algorithm as having structural gain, computational gain, zero gain, or negative gain for a particular data set.

Structural gain is one type of positive gain. An algorithm has structural gain if its output in AAC-CC space lies in the vertical band between A_{NP} and the asymptote, A_∞, in Figure 1. The AAC of a structural gain method is lower than A_∞, implying that the structure imposed by the algorithm provides an advantage over a random projection method (which has no structure). Note that it is possible for the benchmark asymptote to coincide with the optimal, N-P AAC (i.e. $A_\infty = A_{NP}$). In this case, the structural gain region is empty and no algorithm can have a structural advantage over the benchmark.

Computational gain is another type of positive gain. An algorithm has computational gain if its output in AAC-CC space lies to the right of A_∞ and below the benchmark curve. An algorithm with this type of gain has an AAC that is also achievable by the benchmark curve, but it achieves that AAC at a lower cost than the benchmark. This implies that the algorithm is computationally superior (but not structurally superior) to a random projection method for the considered problem.

An algorithm has zero gain if its output in AAC-CC space is on the benchmark curve. It achieves a similar AAC to the benchmark with a similar computational cost.

An algorithm with negative gain for a particular data set has a higher AAC than the benchmark curve and has a higher computational cost. Its output on the AAC-CC plane is above and to the right of the benchmark curve in Figure 1. Negative gain suggests that the algorithm is ill-suited for the problem under consideration.

3. BENCHMARK CURVE ESTIMATION USING LABELED DATA SAMPLES

In practice, finding one point on the benchmark curve involves a training step and a testing step: the training step determines the "best" random vector out of r and the testing step applies the "best" random vector to construct the benchmark point. The training and testing steps are then repeated a number of times in order to approximate expected values for the ROC curve. The steps are detailed in the pseudocode for Algorithm 1 (and our code is available at [5]). Repeating the process in Algorithm 1 with varying values of r yields multiple benchmark points in AAC-CC space. The points are interpolated to form the benchmark curve.

To find the asymptote, the training and testing steps are repeated for increasing values of r until the resulting benchmark points no longer improve in AAC. The process can be halted earlier if the computational cost becomes prohibitive. If the computations are halted earlier, one obtains merely an upper bound on the asymptote value.

3.1. Training

Training for the construction of a benchmark point includes the steps in the "benchmark_train" procedure of Algorithm 1. Let $\{x_i^{train}\}_{i=1,...,n} \in \mathbb{R}^p$ be the feature vectors representing a labeled training data set of n samples. We generate r random vectors, $\{v_j\}_{j=1,...,r} \in \mathbb{R}^p$, by randomly selecting each component from a uniform distribution on $[-1, 1]$. The $\{x_i^{train}\}$ are projected onto each random vector, creating 1-D signatures y_j^{train} for $j = 1, ..., r$, where

$$y_j^{train} = \frac{1}{|v_j|}(x_1^{train} \cdot v_j, ..., x_n^{train} \cdot v_j).$$

Each signature y_j^{train} is then used to produce a ROC curve. We find the AAC for each of the r ROC curves, then select the curve with the lowest AAC and record its corresponding random vector. This

Algorithm 1 Training and testing procedures for benchmark points

procedure BENCHMARK_TRAIN(r, K, x^{train}, n)
  $t\_start \leftarrow$ CLOCK_CHECK
  for $i = 1$ to K **do**
    $A\_best[i] \leftarrow 1$
    for $j = 1$ to r **do**
      $v_j \leftarrow$ GENERATE_RANDOM_VECTOR
      for $\ell = 1$ to n **do**
        $y_j^{train}[\ell] \leftarrow x_\ell^{train} \cdot v_j$
      end for
      $ROC\_tmp \leftarrow$ CALCULATE_ROC(y_j^{train})
      $A\_tmp \leftarrow$ CALCULATE_AAC(ROC_tmp)
      if $A\_tmp < A\_best[i]$ **then**
        $A\_best[i] \leftarrow A\_tmp$
        $v\_best_i \leftarrow v_j$
      end if
    end for
  end for
  $t\_train \leftarrow \frac{\text{CLOCK\_CHECK} - t\_start}{K}$
  return v_best, t_train
end procedure
procedure BENCHMARK_TEST(v_best, K, x^{test}, m)
  $t\_start \leftarrow$ CLOCK_CHECK
  for $i = 1$ to K **do**
    for $\ell = 1$ to m **do**
      $y\_best[\ell] \leftarrow x_\ell^{test} \cdot v\_best_i$
    end for
    $ROC_i \leftarrow$ CALCULATE_ROC(y_best)
  end for
  $t\_test \leftarrow \frac{\text{CLOCK\_CHECK} - t\_start}{K}$
  $E\_ROC \leftarrow$ THRESHOLD_AVERAGE($ROC_1, ..., ROC_K$)
  $E\_A \leftarrow$ CALCULATE_AAC(E_ROC)
  return E_A, t_test
end procedure

vector is referred to as the "best" of r random vectors, or v_best. The elapsed computational time (CT) during training is recorded for use in the benchmark curve; it is the experimental representation of the computational cost (CC) of training for the r random vectors.

3.2. Testing

Testing for the construction of a benchmark point includes the steps shown in the "benchmark_test" procedure of Algorithm 1. Let $\{x_i^{test}\}_{i=1,...,m} \in \mathbb{R}^p$ be the feature vectors representing a labeled testing set of m samples. To find a benchmark point, each feature vector x_i^{test} is projected onto v_best. The resulting 1-D signature is y_best, where

$$y\_best = \frac{1}{|v\_best|}(x_1^{test} \cdot v\_best, ..., x_m^{test} \cdot v\_best).$$

The signature y_best is used to create a ROC curve for which the AAC is measured. The elapsed testing time is recorded; it is the experimental representation of the CC of testing for the r random vectors.

3.3. Building the Curve

The training and testing procedure is repeated K times for r random vectors in order to approximate the expected ROC curve (E_ROC), training time (t_train), and testing time (t_test). The expected training and testing times are estimated by finding their means over the K repetitions. The expected ROC curve is estimated by threshold averaging [6], which finds the mean FPR and mean TPR at each threshold. We empirically choose K to produce a good representation of the expected ROC curve; larger values of K tend to produce averages which are more representative of the expected curve. The AAC of the approximate expected ROC curve (labeled E_A) is calculated, then paired with the expected training or testing time to form a point on the AAC-CT plane.

The training and testing procedures for r random vectors (each repeated K times) provide one AAC-CT point. Changing the value of r provides another AAC-CT point. We continue this process with various values of r to produce a series of AAC-CT points, then use linear interpolation to approximate the benchmark curve.

4. NUMERICAL TEST CASES

A detection method may have structural gain, computational gain, zero gain, or negative gain for a given problem. We will demonstrate each of these cases using results from four different data sets.

We found the benchmark curve in each experiment by interpolating the benchmark points for $r = 1, 3, 10, 30, 100, ...$, increasing r until the benchmark reached a clear asymptote or the training time became unreasonably long. Calculations for each point were repeated $K = 30$ times. The AAC of the expected best ROC curve was paired with the corresponding expected training time for each point.

A linear SVM and Gaussian[1] SVM was also used in each experiment to demonstrate the use of the benchmark curve. ROC curves were generated for the SVMs by varying a threshold across the SVM scores and calculating the TPR and FPR. The SVM classifications were repeated 30 times to approximate expected results. The AAC of the expected SVM ROC curve was paired with the expected training time to produce a single point in AAC-CC space.

The code used to construct the benchmark curves for these experiments is at [5]. The SVM implementations are from the SVMlight library [7]. All experiments were performed on a 4-core, 2.33GHz Intel Xeon CPU.

We present four of our experimental results, selected to demonstrate the use of the benchmark curve and the different possible outcomes for comparison to other methods. All experimental results are shown in [8].

The first selected experiment (Fig. 2a) was on two synthetic normal data classes, each with covariance $\mathbb{I}$. The classes had means $(0, 0)$ and $(1, 0)$, such that they had high overlap, and each class contained 2,000 samples which were divided into equal-sized training and testing groups.

We were able to estimate the optimal AAC for the synthetic normal data experiments because the actual probability density function for each class is known. The Neyman-Pearson results were approximated numerically by generating up to 200,000 normally-distributed samples and calculating the ratio of the log-likelihoods for each sample. The log-likelihood ratios were thresholded to produce the approximate optimal ROC curve with the optimal AAC.

The N-P AAC for this data is fairly large at 0.24. The benchmark points decrease in AAC as computational time increases, but they appear to approach an asymptote at about 0.24; that is, they approach the optimal AAC. Notice that in this case, the structural gain region

[1] In our experiments, the (gamma) parameter was roughly tuned by adjusting it through powers of 10 and computing the Xi-Alpha bound.

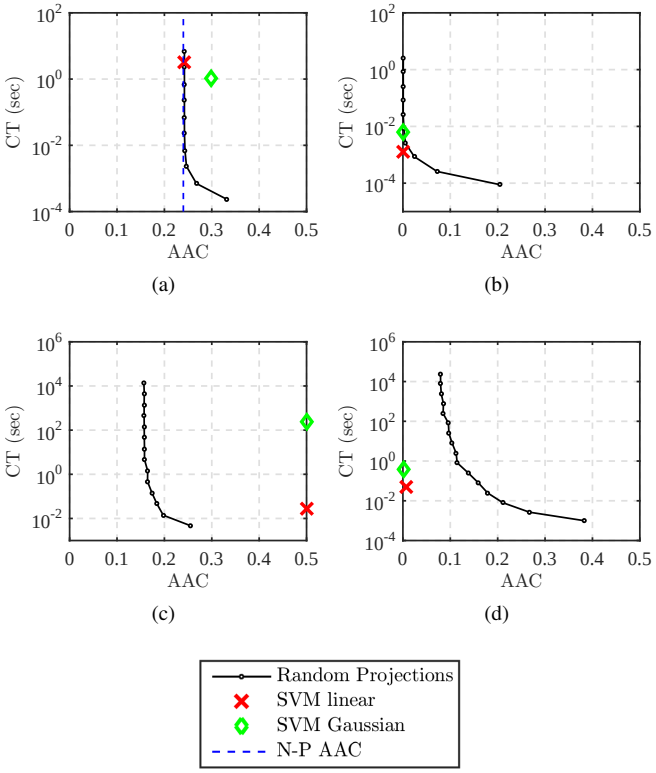

Fig. 2: **Benchmark curves for various data sets.** The linear SVM ($\times$) and Gaussian SVM ($\Diamond$) results are plotted with the benchmark curve ($-\circ$) and the N-P optimal AAC ($--$). (a) Normal Gaussian classes with large amount of overlap. The asymptote to the benchmark curve (A_∞) coincides with the optimal N-P AAC, or A_{NP}. (b) MFEAT Fourier coefficients feature for the 0-1 problem. (c) Daimler-pedestrian Pascal triangle coefficients feature. (d) MSTAR PCA coefficients feature for the BTR70 vs. T72 vehicles.

(between the N-P AAC and the benchmark asymptote) is empty. The linear SVM result has zero gain as it lies on the benchmark curve, indicating that although it produces the optimal result, the linear SVM performs no better than a random projection approach. The Gaussian SVM result surprisingly falls in the negative gain region above the benchmark curve, implying that the model underlying the Gaussian SVM is ill-suited for this problem. Neither the linear SVM nor the Gaussian SVM had positive gain, i.e. they did not outperform the TARP benchmark, a criterion built from random projections. This suggests that the structural and computational advantages provided by the SVM framework are not useful for this type of data; for example, a similar situation may arise with data from two well-separated classes whose distributions are approximately normal.

The second selected experiment (Fig. 2b) was performed on the Fourier coefficients feature of the Multiple Features, or MFEAT, data set [9, 10], for the handwritten digits 0 vs. 1. Each class contained 200 samples, which were divided 70/30 into training and testing groups. The data had 76 dimensions.

The linear SVM result lies at AAC ≈ 0 in the computational gain region below the benchmark curve, indicating that the linear SVM is slightly better-suited for the 0-1 problem with this feature, but only by a small computational margin. The Gaussian SVM result has zero gain as it lies at AAC ≈ 0 on the benchmark curve, indicating that while the Gaussian SVM reaches an optimal AAC, it may be unnecessarily computationally costly for the problem under consideration.

The third selected experiment (Fig. 2c) was performed on the Pascal triangles feature of the Daimler-Pedestrian (PED) data set [11, 12, 13]. Each of our training and testing sets contained 5,000 non-pedestrian images and 4,800 pedestrian images. The data had 130 dimensions.

In this experiment, the benchmark curve reaches a clear asymptote at AAC ≈ 0.16, but the linear and Gaussian SVM results lie at AAC $= 0.5$, corresponding to a scenario where a classifier does not discriminate between classes. Both the linear SVM and Gaussian SVM classified all samples for this feature into a single class. This experiment is significant; the benchmark clearly shows that there is structure to this data which random projections can uncover, yet the linear and Gaussian SVMs failed to find a separation.

The final selected experiment (Fig. 2d) was performed on the dimension-reduced PCA (Principal Component Analysis) coefficients feature of the MSTAR data set [14] for the BTR70 vehicle vs. the T72 vehicle. The PCA dimension reduction was performed such that only the first few components representing approximately 90% of the variance were retained. The training and testing set each included 233 BTR70 samples and 691 T72 samples.

The benchmark for this case approaches an asymptote at AAC ≈ 0.07. The linear and Gaussian SVM results lie near zero AAC in the structural gain region, suggesting that the SVMs have a clear advantage over the random projection methods in this case.

5. CONCLUSION

Real data (e.g. images) often have underlying structure which can be uncovered by simple random projections. Building on this, we proposed a benchmark curve based on a sequence of detection methods using random projection followed by a thresholding (TARP) as a detection method. The curve carves regions into the AAC-CC space; the outputs of other detection algorithms can be represented by points that lie either on the benchmark curve (Zero Gain) or in one of the regions (Negative, Computational, or Structural Gain).

We conducted experiments with both synthetic and real data to demonstrate the use of the benchmark curve and illustrate occurrences of detection methods in each of the benchmark plane regions. Our numerical experiments using various SVM detection methods illustrated the relevance of the benchmarks and show how to interpret the curve to gain insight in a given detection problem.

6. REFERENCES

[1] Sangchun Han and Mireille Boutin, "The Hidden Structure of Image Datasets," in *ICIP, 2015 IEEE International Conference on Image Processing*, September 2015.

[2] Samuel Kaski, "Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering," in *Proceedings of ICJNN'98, 1998 IEEE International Joint Conference on Neural Networks*, May 1998.

[3] Ella Bingham and Heikki Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, August 2001, pp. 245–250, ACM.

[4] Tarun Yellamraju, Jonas Hepp, and Mireille Boutin, "Benchmarks for Image Classification and Other High-Dimensional Pattern Recognition Problems," Submitted, 2016.

[5] Kelsie Larson and Mireille Boutin, "Code and Dataset for TARP Detection Benchmarks," Purdue University Research Repository, 2017.

[6] Tom Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 2006.

[7] Thorsten Joachims, "Making Large-Scale SVM Learning Practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf, C. Burges, and A. Smola, Eds. MIT-Press, 1999.

[8] Kelsie Larson, "Tarp benchmarks for detection problems," M.S. thesis, Purdue University, West Lafayette, Indiana, USA, 2017.

[9] Robert Duin, "Multiple Features Data Set," Delft University of Technology, Department of Applied Physics.

[10] M. Lichman, "UCI Machine Learning Repository," University of California, Irvine, School of Information and Computer Sciences, 2013.

[11] S. Munder and D. M. Gavrila, "An Experimental Study on Pedestrian Classification," *EEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, November 2006.

[12] Mireille Boutin and Shanshan Huang, "The Pascal Triangle of a Discrete Image: Definition, Properties and Application to Shape Analysis," *SIGMA 9*, vol. 31, 2013.

[13] Jonas Hepp, Tarun Yellamraju, and Mireille Boutin, "Code and Dataset for Pattern Recognition Benchmarks," Purdue University Research Repository, 2016.

[14] "MSTAR Public Targets," SDMS, 1995.