A CROC Stronger than ROC: Measuring, Visualizing, and Optimizing Early Retrieval

S. Joshua Swamidass\textsuperscript{1,2,3}, Chloé-Agathe Azencott\textsuperscript{1,2}, Kenny Daily\textsuperscript{1,2}, and Pierre Baldi\textsuperscript{1,2,4}\textsuperscript{*}

\textsuperscript{1}School of Information and Computer Sciences, University of California, Irvine, CA 92697, USA
\textsuperscript{2} Institute for Genomics and Bioinformatics, University of California, Irvine, CA 92697, USA
\textsuperscript{3} Department of Immunology and Pathology, Washington University, Saint Louis, MO 63110, USA
\textsuperscript{4} Department of Biological Chemistry, University of California, Irvine, CA 92697, USA

ABSTRACT

Motivation: The performance of classifiers is often assessed using ROC (or AC) curves and the corresponding areas under the curves (AUCs). However, in many fundamental problems ranging from information retrieval to drug discovery, only the very top of the ranked list of predictions is of any interest and ROCs and AUCs are not very useful. New metrics, visualizations, and optimization tools are needed to address this “early retrieval” problem.

Results: To address the early retrieval problem, we develop the general CROC (Concentrated ROC) framework. In this framework, any relevant portion of the ROC (or AC) curve is magnified smoothly by an appropriate continuous transformation of the coordinates with a corresponding magnification factor. Appropriate families of magnification functions confined to the unit square are derived and their properties are analyzed together with the resulting CROC curves. The area under the CROC curve (\text{AUC(CROC)}) can be used to assess early retrieval. The general framework is demonstrated on a drug discovery problem and used to discriminate more accurately the early retrieval performance of five different predictors. From this framework, we propose a novel metric and visualization—the CROC(exp), an exponential transform of the ROC curve—as an alternative to other methods. The CROC(exp) provides a principled, flexible, and effective way for measuring and visualizing early retrieval performance with excellent statistical power. Corresponding methods for optimizing early retrieval are also described in the Appendix.

Availability: Datasets are publicly available. Python code and command-line utilities implementing CROC curves and metrics is available at http://pypi.python.org/pypi/CROC/.

Contact: pbaldi@ics.uci.edu

1 INTRODUCTION

One of the most widely used tools to assess the performance of a classification or ranking algorithm in statistics and machine learning is the ROC (Receiver Operating Characteristic) curve, plotting true positive rate versus false positive rate, together with the corresponding area under the ROC curve (AUC) metric. However, in many applications, ranging from information retrieval to drug discovery, the ROC curve and the AUC metric are not very useful. This is because the total number of objects to be classified or ranked, such as web pages or chemical molecules, tends to be very large relative to the number of objects towards the top of the list that are practically useful or testable due to, for instance, financial constraints. Specifically, consider a typical drug discovery situation where one is interested in discovering molecules that may be “active” among a library of 1,000,000 molecules by computational virtual screening methods, such as docking or similarity search. Depending on the financial conditions and the details of the corresponding experimental setup, experimentalists may be able to test in the laboratory, for instance, only the top 1,000 hits on the list. In such conditions, the majority of the ROC curve is without much relevance and the AUC is useless. Only the early portion of the ROC curve is relevant. Furthermore, the precise ranking of the molecules, particularly for the bottom 999,000 molecules, is also of very minor interest. Similar observations can be made in many other areas, ranging from fraud detection to web page retrieval. What one is really interested in across all these cases is the notion of early enrichment/recognition/retrieval, having as many true positives as possible within the list of top hits.

To further drive this point, consider the following three cases from Truchon and Bayly (2007) corresponding to an algorithm which either: (1) ranks half the positive candidates at the top of the list and half at the bottom; (2) distributes the positive candidates uniformly throughout the list; or (3) ranks all the positive candidates exactly in the middle of the list. All three cases yield an AUC of 0.5 although, if only the top few hits can be experimentally tested, case 1 is clearly better than case 2 which, in turn, is better than case 3. Good early recognition metrics and visualization tools ought to easily discriminate between these cases and rank them appropriately.

Several metrics have been suggested in the literature to address the early retrieval problem, but none seems entirely satisfactory. Some metrics lack smoothness and require setting arbitrary thresholds, including looking at the derivative of the smoothed ROC curve at or near the origin, or looking at the ROC curve and its area over the interval \([0, t]\) for some threshold \(t\) (e.g. \(t = 0.05\)). However, as we shall see, methods with hard threshold cutoffs tend to be less statistically powerful than methods without threshold cutoffs, because they discard meaningful differences in the performance of the classifiers just beyond the cutoff threshold. Other interesting metrics which explicitly try to avoid setting an arbitrary threshold (Sheridan \textit{et al.}, 2001; Truchon and Bayly, 2007; Clark

*to whom correspondence should be addressed
and Webster-Clark, 2008) have other limitations, such as being untunable, lacking good visualization or generality, or requiring the non-principled choice of a transformation (see Discussion).

In this paper, the early retrieval problem is addressed by introducing and studying a general framework—the CROC framework—for both measuring and visualizing early retrieval performance in a principled way that is more suitable than the ROC curve and the AUC measure. The overall approach is demonstrated on a drug discovery problem. Finally, CROC is discussed in relation to other metrics in the literature and shown to provide a general unifying framework. Methods for maximizing early retrieval performance are described in the Appendix.

2 THE CROC FRAMEWORK

2.1 The ROC

Classification (or ranking) performance can be assessed using single-threshold metrics. For a given threshold, the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) can be used to compute quantities such as sensitivity, specificity, precision, recall, and accuracy (e.g. Baldi et al. 2000). Measuring performance at a single threshold, however, is somewhat arbitrary and unsatisfactory. To obviate this problem and capture performance across multiple thresholds, it is standard to use ROC or AC curves and the area under these curves. The ROC (Receiver Operating Characteristic) curve plots the TP rate (TPR) as a function of the FP rate (FPR). The AC (Accumulation Curve or Enrichment Curve) plots the TP rate as a function of the FDP, the fraction of the data classified as positive at a given threshold, and is occasionally used to assess virtual screening methods (Seifert, 2006). The ROC and AC curves are related by a linear transformation of the x-axis. The area under the ROC (AUC[ROC]) or under the AC (AUC[AC]) can be used to quantify global performance. The AUC[AC] is a linear transform of the AUC[ROC] and the two metrics are approximately equivalent as the size of the dataset increases (Truchon and Bayly, 2007).

Within the ROC or AC framework, a classifier can be assessed by comparing its performance to the performance of a random classifier, or to the best possible and worst possible classifiers. The random classifier corresponds to an algorithm that randomly and uniformly distributes all the positive candidates throughout its prediction-sorted list. This is exactly equivalent to using a random number generator, uniform in [0, 1], to produce class membership probabilities. For both the ROC and AC plots, averaging the performance curves of a large number of random trials of a random classifier constructed in this manner yields a straight line from the origin to the point (1,1), with an area of 0.5. Classifiers worse than random, therefore, yield an AUC[ROC] less than 0.5. Furthermore, the variance of the random AUC[ROC] or AUC[AC] can be estimated analytically or by Monte Carlo simulations and used, for instance, to derive a Z-score for the area. The best and worst classifiers assume that all the positive candidates are ranked at the top or bottom, respectively, of the prediction sorted list. These also provide a useful baseline, especially for visualization of the AC curve by providing extremal curve boundaries. Masking out the portion of the plot above the best possible curve and below the worst possible curve highlights the region within which the AC curve is confined (Figure 1).

However, despite their usefulness, the ROC and AC curves and their AUCs measure and visualize classification performance uniformly across the entire data and therefore are poorly suited to measure and visualize early retrieval performance. Thus, to address this shortcoming, we next introduce the CROC framework using ROC curves as the starting point, but similar ideas apply immediately to AC curves.

2.2 The CROC

Here we propose a different approach whereby any portion of the ROC curve of interest is magnified smoothly using an appropriate continuous and monotone (hence 1 to 1) function f from [0,1] to [0,1], satisfying f(0) = 0 and f(1) = 1. Since this general approach tends to concentrate resources on a particular portion of the ROC curve, we call it the Concentrated ROC (CROC) approach (and CAC for Concentrated-AC). The approach can be used to magnify the x-axis, or the y-axis, or both. Here we focus primarily on magnification along the x-axis, since this is the most critical in early retrieval, but the same ideas can be applied to the y-axis if necessary. Because we are interested in the early portion of the ROC curve, we need a function that expands the early part of the [0,1] interval, and contracts the latter part. This is easily achieved using a function f which is concave down (f′(x) < 0), i.e. with a negative second derivative in the differentiable case. Many such functions are possible and it is reasonable to further require that such functions have a simple expression, with at most one parameter to control the overall level of magnification. Examples of natural choices for the function f, using exponential, power, or logarithmic representations include:

\[
 f(x) = \frac{1 - e^{-\alpha x}}{1 - e^{-\alpha}} , \quad f(x) = x^{1/(\alpha+1)}, \quad f(x) = \frac{\log(1 + \alpha x)}{\log(1 + \alpha)} \quad (1)
\]

where \( \alpha > 0 \) is the global magnification factor. The logarithmic transformation \( f(x) = \log(1 + x)^{1/(\alpha+1)} \) is also possible but yields curves that are fairly similar to the power transformation above, as can be seen with a Taylor expansion. Of course, the common technique of displaying the initial portion of the ROC curve is equivalent to using the transform function:

\[
 f(x) = \min \{ x(1 + \alpha), 1 \} = \min \{ x/t, 1 \} , \quad (2)
\]

where \( t = 1/(1 + \alpha) \) is the hard threshold cutoff after which the ranks of positive instances are ignored. For all these transformations, at a given point \( x \), the derivative \( f′(x) \) measures the local degree of magnification: a small interval of length \( dx \) near \( x \) is transformed into an interval of length \( f′(x)dx \). If \( f′(x) > 1 \), the region around \( x \) is being stretched. If \( f′(x) < 1 \), the region around \( x \) is being contracted. Thus the functions under consideration magnify the early part of the x-axis until the point where \( f′(x) = 1 \) where there is no magnification. Past this point, the functions contract the x-axis. In contrast, the threshold cutoff function of Equation 2 corresponds to a constant magnification factor of 1/t over \([0, t]\).

2.3 Choice of Magnification Curve and Parameters

The choice of magnification curve \( f \) and magnification parameter \( \alpha \) can be done using additional considerations, such as “user behavior”. For instance, in a web page retrieval application, one may know on average how often the first, second, third, . . . records are clicked. The corresponding decreasing curve of how relevant each rank is provides valuable information for the range and magnification levels required. While it is possible to use the empirical relevance levels recorded in a particular application, for a more principled approach here we can imagine that this relevance \( g(r/N) \) decays exponentially or as a power law, as a function of the rank \( r \) (the scaling factor \( N \) is the total number of examples). Thus \( g(r/N) = Ce^{-\beta r/N} \) or \( g(r/N) = C/(r/N)\beta+1, \) with \( \beta > 0 \). It is reasonable to require the local magnification factor to be proportional to the corresponding relevance so that

\[
 f′(x) = g(Dg(x)) \quad (3)
\]

for some proportionality constant \( D \), so that \( f(x) = Dg(x) + K \), where \( G \) is a primitive of \( g \) and \( K \) is another constant. In the exponential and power-law cases, Equation 3 can be solved exactly taking into account the boundary values \( f(0) = 0 \) and \( f(1) = 1 \). When \( g \) decays exponentially, a simple calculation yields precisely the magnification function \( f(x) = \frac{1 - e^{-\beta x}}{1 - e^{-\beta}} \) thus \( \alpha = \beta \). When \( g \) decays like a power law, one obtains precisely \( f(x) = x^{-\beta} \) thus \( \alpha = \beta - 1 \). Thus, in short, the magnification function can be derived precisely from the relevance function.

In the application to be considered, an exponentially decaying relevance fits well with the drug discovery setting. Thus, in the rest of the paper, for both theoretical and practical reasons we use the exponential transformation \( f = (1 - e^{-βx})/(1 - e^{-β}) \) with magnification factor \( \beta \). The same ideas, however, can be applied immediately with any other transformation.
A CROC Stronger than ROC

\[
x_1 = -\frac{1}{\alpha} \log\left(\frac{1 - e^{-\alpha}}{\alpha}\right) \approx \frac{\log(\alpha)}{\alpha}. \tag{4}
\]

the approximation being valid for large values of \(\alpha\). Alternatively, one could choose \(\alpha\) to sensibly map a particular value of \(x\) to another one (e.g. \(f(0.1) = 0.5\), or to match other particular conventions (e.g \(\alpha = 20\) in the exponential case corresponds roughly to 8\% enrichment).

2.4 AUC[CROC] and Random Classifiers

The early recognition performance of a classifier can be measured by the area under the CROC (AUC[CROC]). This area depends on the magnification factor \(\alpha\) and therefore both the area and \(\alpha\) must be reported. Under smoothness assumptions, it can be shown by Taylor expanding the ROC function, that in the limit of very large \(\alpha\) the AUC[CROC] goes to 0 like ROC'(0)/\(\alpha\), where ROC'(0) represents the derivative of the ROC curve at the origin. Note that if both the \(x\) and \(y\) axis are magnified with the same transform using the same \(\alpha\), then the limit is ROC'(0). Thus asymptotically the AUC[CROC] depends on the tangent of the ROC curve at the origin, although in practice CROC curves are interesting primarily for finite values of \(\alpha\) (see Results).

As for the AUC[ROC], the AUC[CROC] can also be compared to the same area computed for a random classifier. The ROC curve of a completely random classifier is the straight diagonal line \(y = x\) between the points \((0,0)\) and \((1,1)\). The CROC curve of a random classifier is easily obtained by applying the magnification function \(f\) to this line. In the case of the exponential transform, the equation of the corresponding curve is given by

\[
y = -\frac{\log[1-x(1-e^{-\alpha})]}{\alpha}. \tag{5}
\]

This curve is approximately a line \(y \approx x(1-e^{-\alpha})/\alpha\) when \(\alpha\) is small. But otherwise the curve is far from being a line (see Figures 1 and 2). The AUC[ROC] of a random classifier is 0.5. The AUC[CROC] of a random classifier is given by

\[
\int_0^1 \frac{\log[1-x(1-e^{-\alpha})]}{\alpha} \approx \frac{-\alpha e^{-\alpha} - e^{-\alpha} + 1}{\alpha(1-e^{-\alpha})} = \frac{1}{\alpha} - \frac{e^{-\alpha}}{1-e^{-\alpha}} \tag{6}
\]

and so it behaves like \(1/\alpha\) for large values of the magnification factor \(\alpha\).

2.5 Assessing Significance

Assessing the statistical significance of the difference in performance between two classifiers is a fundamentally more important and general task than assessing the significance between a classifier’s performance and a random classifier. In most interesting settings, several classifiers are available which clearly perform better than random. The differences in performance between these classifiers, however, can be more subtle and require statistical assessments. In simple terms, how significant is the difference in AUC[ROC] for classification or AUC[CROC] for early enrichment between two different algorithms? We propose and compare six ways of assessing the significance of the difference between two early retrieval performances: (1) an unpaired permutation test; (2) a paired permutation test; (3) an unpaired t-test; (4) a paired t-test; (5) an unpaired Wilcoxon test; and (6) a paired Wilcoxon test.

The unpaired permutation test follows from the observation that the performance of two algorithms being compared is exactly defined by the size \(N\) of the entire dataset and the exact ranks assigned by each algorithm to the positive instances in this dataset (Zhao et al., 2009). Next, the ranks of the positive instances from both algorithms are pooled together and randomly partitioned into equally sized sets of ranks. Once the ranks are sampled in this way, the difference in performance between these two randomly derived methods is computed using, for instance, AUC[ROC] or AUC[CROC]. Several samples of this difference in performance can be drawn by repeatedly partitioning the ranks and compared to the observed value of interest. Significance can be assessed by estimating a p-value.
corresponding to the percentage of times that the sampled differences in performance are greater than the observed difference. Enough samples must be drawn to establish the desired level of confidence in the p-value. For example, if the performance difference between algorithms A and B is 0.13, then this difference would be considered significant at a \( p = 0.05 \) level if at most 50 out of 1,000 sampled performance differences are greater than 0.13.

The paired permutation test is novel in this context and computed in nearly identical manner. The only difference is the way the ranks are partitioned. For the paired permutation test, the ranks are partitioned so as to ensure that each instance’s rank is included exactly one time in each partition. This is equivalent to flipping an unbiased coin once for each instance in the dataset. If the coin comes up heads, the rank of this instance in the first algorithm’s list is put in the first partition and the corresponding rank produced by the second algorithm is put in the second partition. If the coin comes up tails, these ranks are placed in the opposite partitions. Once the ranks have been partitioned, the Monte Carlo procedure is repeated a sufficient number of times to determine a corresponding p-value.

The permutation tests we have described make few assumptions about the data but do require substantial amounts of computations, especially for the reliable estimation of small p-values. A faster approach is obtained by performing the appropriate unpaired and paired t-tests. These t-tests rely on normality assumptions or approximations, which may be more or less accurate, and therefore the t-tests can be expected to be less precise than extensive permutation testing.

The t-tests are derived by observing that computing the AUC[CROC] is equivalent to computing the mean of the transformed FPRs of the positive instances. For example, if a method ranks five positive instances at ranks \( \{1, 2, 4, 5, 7\} \) in a list of 10 total instances, this corresponds to FPRs of \( \{0, 0, 0.2, 0.2, 0.4\} \) for each positive instance, and to transformed FPRs of \( \{f(0), f(0), f(0.2), f(0.2), f(0.4)\} \). It can easily be seen that the AUC[CROC] is exactly the mean of \( (1 - f(0)), (1 - f(0)), (1 - f(0.2)), (1 - f(0.2)), (1 - f(0.4)) \). From this observation, one can compute a t-test between the population of transformed FPRs produced by one classifier against the similar population produced by another classifier. This approximates the results of the unpaired permutation test. Similarly, a paired t-test between the two populations approximates the results of the paired permutation test. An even faster approximation for the unpaired case is obtained by applying the t-test using the mean and standard deviation of each classifier’s performance derived by cross validation methods.

A parallel derivation shows that computing the AUC[CAC] is equivalent to computing the mean of the transformed ranks. Using the same example, this would correspond to computing the mean of \( \{f(0.1), f(0.2), f(0.4), f(0.5), f(0.7)\} \). From this observation, it is easy to see how a t-test approximation of the permutation test can be constructed for the AUC[CAC] as well.

The accuracy of the t-test approximations depend on the normality of the distribution of the transformed ranks. Clearly, the transformed ranks can be very skewed, and therefore violate the t-tests’ normality assumptions. This motivates the last two significance tests implemented here, which use either a paired or unpaired Wilcoxon test on the same population of transformed ranks used for the t-tests. The Wilcoxon tests are more robust to outliers and do not make normality assumptions; however, they are not designed to analytically approximate the permutation tests.

3 DRUG DISCOVERY DATA, REPRESENTATIONS, AND SIMILARITY MEASURES

Drug Discovery Data. We use the benchmark dataset associated with the Drug Therapeutics Program (DTP) AIDS Antiviral Screen made available by the National Cancer Institute (NCI) (http://dtp.nci.nih.gov/docs/aids/aids_data.html). The set contains assay results for 42,678 chemicals experimentally tested for their ability to inhibit the replication of the Human Immunodeficiency Virus in vitro. Initially, the data is divided into three classes: inactive, active, and moderately active.

In what follows, we combine the active and moderately active classes into a single “active” class containing about 3.5% of the data. Thus the final dataset contains 1,503 active molecules and 41,175 inactive molecules. All performance metrics are computed using 10-fold cross-validation experiments.

Chemical Representations. Molecules are described by their labeled molecular graph, where the vertices correspond to the atoms of the molecule, labeled by the atom type (e.g. Carbon (C), Nitrogen (N), Oxygen (O)), and the edges correspond to the bonds connecting these atoms together, labeled by the bond type (e.g. single, double). In chemoinformatics, these variable-size molecular graphs are routinely converted into long, fixed-size, binary fingerprint vectors. Each bit in a fingerprint representation corresponds to the presence or absence of a particular feature in the molecular graph. Typical sets of features used in the literature (Leach and Gillet, 2005; Hassan et al., 2006; Swamidass and Baldi, 2007) are labeled paths up to a certain length, or labeled trees up to a certain depth. While the methods to be presented can be applied with any set of features, here we use circular substructures (Hassan et al., 2006), corresponding to labeled trees of depth up to 2. Each vertex is labeled by the atomic symbol of the corresponding atom together with the number of non-hydrogen atoms to which it is bonded (e.g., C3 for a carbon with three non-hydrogen atoms attached), and each bond is labeled according to its bond type (single, double, triple, or aromatic). We choose this particular labeling scheme and set of features because the resulting representations seem to yield reasonable performance based on experiments performed using other datasets (not reported).

Chemical Similarity Measures. Although there are several possible ways of defining similarity between fingerprint vectors (Holliday et al., 2002), by far the most widely used approach is the well known Jaccard-Tanimoto similarity measure (Leach and Gillet, 2005). If \( \tilde{A} \) and \( \tilde{B} \) are the fingerprint vectors representing molecules \( A \) and \( B \), the Jaccard-Tanimoto similarity between \( A \) and \( B \) is defined by:

\[
S(\tilde{A}, \tilde{B}) = \frac{A \cap B}{A \cup B}.
\]

where \( \tilde{A} \cap \tilde{B} \) is the number of 1-bits in the intersection \( \tilde{A} \) AND \( \tilde{B} \), and \( \tilde{A} \cup \tilde{B} \) is the number of 1-bits that appear in the union \( \tilde{A} OR \tilde{B} \). It is well known that the Jaccard-Tanimoto similarity measure satisfies the Mercer’s conditions and, hence, yields a Mercer kernel (Swamidass et al., 2005a).

4 CLASSIFIERS

In the experiments, we use the CROC framework to compare the early retrieval performance of five different classifiers: RANDOM, MAX-SIM, kNN, SVM, and IRV. Although the details of the algorithms are not relevant to this study, we describe them briefly for completeness. RANDOM corresponds to a random classifier which assigns a random probability vector to each instance. The score of a new molecule \( X \) is given by:

\[
z(X) = \frac{1}{P_{i=1}^{P}} S(X, X_i).
\]

where \( S(X, X_i) \) is the similarity between molecule \( X \) and each of the \( P \) reference molecules. MAX-SIM has been well studied and is straightforward to implement. However, it is a rather unrefined method, which takes very little information into account ignoring, for instance, all information about known inactive molecules. A slightly more refined method is the k-Nearest-Neighbor algorithm, where one scores a molecule using the fraction of its k-nearest-neighbors that are active. For the SVM algorithm, it is well known that the Tanimoto similarity satisfies Mercer’s condition (Swamidass et al., 2005a) and therefore it defines a kernel which can be used in an SVM for classification (see Swamidass et al. (2005b); Mahé et al. (2006); Azencott et al. (2007) for other related kernels). Finally, we use the IRV (Influence Relevance Voter) algorithm (Swamidass et al., 2009), which can...
be viewed as a refinement of kNN. At a high-level, the IRV is defined by a preprocessing step, during which all the neighbors— as defined by the Tanimoto similarity metric— of a test molecule are identified, and a processing step during which information from each neighbor is fed into a neural network to produce a prediction. The prediction is computed from the influence of the training example contributes to the prediction and can be used to interpret These influences indicate exactly how much, and in which direction, each neighbor’s “relevance,” encoding how much each neighbor should “influence” of each neighbor, which in turn is computed as the product of each neighbor’s “relevance,” encoding how much each neighbor should affect the prediction, and “vote,” encoding the direction towards which the prediction should be pushed. The output probability of membership in the active class is computed as

\[ z(X) = \sigma \left( w_x + \sum_{i} I_i(X) \right) \quad (9) \]

where \( X \) is the test molecule, \( i \) ranges from 1 to \( k \) over all the \( k \) nearest neighbors, \( I_i \) is the “influence” of the \( i \)th neighbor on the output, \( w_x \) is the bias of the output node, and \( \sigma(z) \) is the logistic function \( 1/(1 + e^{-z}) \).

These influences indicate exactly how much, and in which direction, each training example contributes to the prediction and can be used to interpret each final prediction. The influence of the \( i \)th node is defined multiplicatively by \( I_i = R_i(X) V_i(X), \) where \( R_i(X) \) is the relevance and \( V_i(X) \) is the vote of the \( i \)th neighbor. Here the relevance is defined by

\[ R_i(X) = \sigma \left( w_{x_i} + w_{s_i}(X) + w_{r_i}(X) \right), \quad (10) \]

where \( s_i \) is the similarity \( S(X, N_i) \) of the \( i \)th closest neighbor to the test compound, \( r_i \) is the rank of the \( i \)th neighbor in the similarity-sorted list of neighbors, \( w_x \) and \( w_{\beta} \) are parameters tuning the importance of different inputs, and \( w_{\alpha} \) is the bias of this logistic function. For this study, we define the vote by: \( V_i = w_0 \) if \( c_i = 0 \) and \( V_i = w_1 \) if \( c_i = 1 \), where \( w_0 \) is the weight associated with inactive neighbors, \( w_1 \) is the weight associated with active neighbors, and \( c_i \in \{0, 1\} \) is the class of the \( i \)th neighbor. The IRV is trained by gradient descent on the relative entropy (or negative log-likelihood). Thus this version of the IRV has only 6 tunable parameters: three parameters \( w_x, w_{s_i}, \) and \( w_{r_i} \) shared across all \( k \) neighbors, and three additional parameters, \( w_1, w_0, \) and \( w_{\alpha} \)—making overfitting unlikely. In the experiments, the neighborhood parameter \( k \) is set to 20 for both the kNN and IRV algorithms.

5 RESULTS

5.1 CROC Metric and Visualization

First, the ROC and CROC frameworks are compared by examining how they assess the early retrieval performance of the classifiers. Figure 2a shows the ROC curves of the various classifiers on the HIV data. It is virtually impossible to disentangle their performances in terms of early recognition using the ROC curves. In contrast, Figures 2b, 2c, and 2d obtained by using exponential CROC curves with different global magnification factors \( \alpha \), show immediately that the CROC framework disambiguates the early retrieval performance of the classifiers. In these plots, increasing magnification factors of \( \alpha = 7, 14, \) and 80 are used, corresponding...
and second-best performances are in italics.

The results are consistent at all magnification scales, thus the CROC approach appears to be robust with respect to the choice of $\alpha$.

Table 2. Estimated $p$-values for the difference in performance for SVM vs IRV (upper panel) and kNN vs IRV (lower panel). Permutation tests were sampled 10,000 times. $p$-values less than 0.01 are in bold and $p$-values between 0.01 and 0.05 are in italics.

<table>
<thead>
<tr>
<th></th>
<th>AUC[ROC]</th>
<th>AUC[CROC]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha = 7$</td>
<td>$\alpha = 14$</td>
</tr>
<tr>
<td>RANDOM</td>
<td>0.500</td>
<td>0.142</td>
</tr>
<tr>
<td>MAXSIM</td>
<td>0.806 (± 0.002)</td>
<td>0.592 (± 0.002)</td>
</tr>
<tr>
<td>kNN</td>
<td>0.742 (± 0.003)</td>
<td>0.638 (± 0.004)</td>
</tr>
<tr>
<td>SVM</td>
<td>0.852 (± 0.004)</td>
<td>0.644 (± 0.003)</td>
</tr>
<tr>
<td>IRV</td>
<td>0.845 (± 0.002)</td>
<td>0.656 (± 0.003)</td>
</tr>
</tbody>
</table>

Table 1. AUC[ROC] and AUC[CROC] achieved by the 5 classifiers obtained using 10-fold cross-validations on the HIV data. Best performances are in bold and second-best performances are in italics. Standard deviations are in parentheses.
between the best methods—IRV, SVM, and kNN—are small, they are significant, as demonstrated in the next section.

5.2 Assessing Significance

We have applied the six proposed methods for assessing significance (paired and unpaired permutation tests, paired and unpaired t-tests, paired and unpaired Wilcoxon tests) to all the known metrics for assessing early retrieval (e.g. AUC[ROC], AUC[ROC] with hard threshold cutoff \( t \), AUC[CROC], AUC[pROC], BEDROC) and to all five classifiers. For the methods with hard threshold cutoffs, the thresholds were chosen to correspond to general practice (\( FP = 50 \) corresponds to ROC50) and to the values of the magnification factor \( \alpha \) we tested, thus \( t = 0.5 \) corresponds to \( \alpha = 7 \), \( t = 0.1 \) corresponds to \( \alpha = 14 \), and \( t = 0.0086 \) corresponds to \( \alpha = 80 \). In total, we obtained well over 100 p-value measurements. Only the most salient results are reported here.

Using the permutation test of significance several variants of the CROC yield p-values less than 0.05 and 0.01 (Table 2) confirming that the difference are statistically significant. Several additional patterns are discernible in the significance tests across different performance metrics. First, as expected, the methods with hard threshold cutoffs perform better than AUC[ROC] but are substantially weaker than several of the metrics devoid of any hard threshold cutoff. Second, consistent with prior studies on statistical power, the AUC[pAC] has better power than the AUC[pROC], which has better power than the AUC[ROC] (Zhao et al., 2009). Third, CROC variants have more power than CAC variants, pROC, and ROC. Other metrics reviewed in the Discussion, such as RIE (Robust Initial Enhancement), BEDROC (Boltzman-Enhanced Discrimination of the ROC), and SLR (Sum of Log Ranks) correspond to affine transforms of AUC[CAC] or AUC[pAC] and therefore yield the same p-values as AUC[CAC] or AUC[pAC].

Most importantly, the significance of the early retrieval performance differences between the classification algorithms can be established with high confidence using the AUC[CROC](\( exp \)) on a dataset where the AUC[ROC] cannot establish significance. Furthermore, the paired permutation test appears to be the most powerful method for assessing significance. While the t-test and Wilcoxon methods produce reasonable approximations, their results are somewhat different from the results produced by the permutation tests and also less powerful. For the t-test, this is probably as a result of the violations of the underlying normality assumptions. If a t-test is to be used to save computational time, the paired t-test is preferable over the unpaired t-test.

6 DISCUSSION AND CONCLUSION

There have been other attempts to try to address the early enrichment problem beyond using a low threshold cutoff for the ROC or AC curves. Clark and Webster-Clark (2008) propose to apply a different logarithmic transformation to the ROC curve by transforming its \( x \)-axis using

\[
p(x) = \log_{10}(\max[x, 0.5/N]).
\]

In this formula, the \( \max \) function is used to avoid a 0 argument in the logarithm when \( x = 0 \). The transformed curve is called a pROC curve by these authors, and obviously the same transformation could be used to derive a PAC curve which does not require the zero-point correction. ROC and AC curves transformed by this equation extend from \( \log_{10}(0.5/N) \) to \( (0, 1) \) (Figure 1). This logarithmic transformation, however, has two deficiencies. First, the width of the curve varies based on the number of examples. This means that performance curves from datasets with different sizes cannot be meaningfully compared on the same plot. Furthermore, there is no way to tune the importance of the early portion of the curve. In some applications, only the top ten percent may be interesting or experimentally testable, while in other applications only the top one percent may be useful. Ideally, a metric should be tunable to these differences.

Although derived from different standpoints, additional metrics proposed in the literature are linear affine transforms of our metrics and can be written in the present notation. Specifically, Sheridan et al. (2001), Truchon and Bayly (2007), and Zhao et al. (2009) have respectively proposed to use the Robust Initial Enhancement (RIE), the Boltzman-Enhanced Discrimination of the ROC (BEDROC), and the Sum of Log Ranks (SLR) metrics. The RIE metric corresponds to

\[
RIE \approx \frac{AUC[\text{CAC}(exp)] - 1 + \frac{1}{1 - \exp(-\alpha)}}{AUC[\text{CAC}(exp)]_{\text{rand}} - 1 + \frac{1}{1 - \exp(-\alpha)}}.
\]

Fig. 3. The geometric relationship between the CAC(\( exp \)) curve and two early recognition metrics. Each letter, from A to F, corresponds to an area delimited by curves and the boundary of the plot. Both the Robust Initial Enhancement (RIE) and Boltzman-Enhanced Discriminate Receiver Operating Characteristic (BEDROC) metrics can be expressed as ratios of particular areas of the plot.

where AUC[\( CAC(\text{exp}) \)]_{\text{rand}} is the area under the Concentrated-AC curve obtained by exponential transformation (CAC(\( exp \))) of the random classifier. The BEDROC metric corresponds to

\[
\frac{AUC[\text{CAC}(exp)] - AUC[\text{CAC}(\text{exp})]_{\text{min}}}{AUC[\text{CAC}(\text{exp})]_{\text{max}} - AUC[\text{CAC}(\text{exp})]_{\text{min}}}.
\]

where AUC[\( CAC(\text{exp}) \)]_{\text{max}} and AUC[\( CAC(\text{exp}) \)]_{\text{min}} are, respectively, the area under the best and worst possible CAC(\( exp \)) curves. The SLR metric can be approximated as

\[
SLR \approx N_{+} \cdot (AUC[pAC] + \log N),
\]

where \( N \) is the total number of test candidates being scored and \( N_{+} \) is the number of positives. Thus RIE and BEDROC are affine transforms of AUC[CAC] and thus have the same statistical power as AUC[CAC]. SLR is an affine transform of AUC[pAC] and thus has the same statistical power as AUC[pAC]. Note that the approximation error in Equations 12 and 14 is negligible since it
is less than the fluctuations introduced by the ambiguity in the
definition of the AC curve and its area, depending on whether
one uses the rectangular or trapezoidal rule, corresponding to
whether consecutive points are connected by discrete jumps or linear
interpolation. These fluctuations become also irrelevant on large
datasets. Ignoring these fluctuations, which only subtly alter the
computations, REI and BEDROC have a geometric interpretation
depicted in Figure 3. The particular standpoints originally used to
derive these other metrics did not lend themselves to deriving a more
general magnification framework that can be extended, for instance,
from AC curves to ROC curves, or to different magnification
functions, or to derive corresponding visualizations. Furthermore,
both the RIE and SLR metrics are larger than one and their ranges
vary with the dataset to which they are applied, which is somewhat
at odds with most other commonly used metrics for classification.
The unifying framework presented here addresses all these issues.

Our data also suggests that the CROC(exp), the variant we
propose for general use in early recognition, has better statistical
power than other commonly used metrics (ROC, pROC, and SLR, REI,
BEDROC). It is tempting to perform more exhaustive benchmarks
on the HIV data set. Best performances are in bold and second-best performances are in italics.

<table>
<thead>
<tr>
<th>AUC-CROC</th>
<th>Iterations</th>
<th>No weight</th>
<th>$w_r = e^{-r/N}$</th>
<th>$w_r = \frac{e^{-\gamma r/N}}{\gamma}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>After 3 iterations</td>
<td></td>
<td>0.368</td>
<td>0.383</td>
<td>0.364</td>
</tr>
<tr>
<td>After convergence</td>
<td></td>
<td>0.404</td>
<td>0.388</td>
<td>0.381</td>
</tr>
<tr>
<td>101</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3. Comparison of the area under the CROC curves ($\alpha = 80$) of several weight update schemes for the IRV, after 3 training iterations and after convergence of the algorithm, together with the total number of iterations for convergence. Results are 10-fold cross-validated on the HIV data set. Best performances are in bold and second-best performances are in italics.

**ACKNOWLEDGEMENT**

**Funding:** Work supported by a Laurel Wilkening Faculty Innovation award, an NIH Biomedical Informatics Training grant (LM-07443-01), and NSF grants EIA-0321390, CCF-0725370, and IIS-0513376 to P.B., and an IBM PhD Fellowship to C.A.

**APPENDIX A: MAXIMIZING EARLY RETRIEVAL**

An important question is whether a learning algorithm can be
developed to emphasize early recognition, as opposed to
classification or ranking over the entire data. A possible approach
to address this issue is to try to optimize CROC behavior, rather
than ROC behavior by, for instance, maximizing the AUC[CROC]
rather than the AUC[ROC]. To this end, we propose an iterative
algorithm where the training examples are reweighted after each
training epoch giving, everything else being equal, higher weights
to higher ranked examples. We describe several variations on this
basic idea and explore them in the simulations. We assume that there
are $N$ training examples. All the examples are initialized uniformly,
with a weight equal to $1/N$. After each training epoch $t$, the weights
are recomputed according to a polynomial or exponential decaying
scheme of the form

$$w_r(t + 1) = Ce^{-\gamma t/r}$$

where $w_r(t + 1)$ denotes the weight of the example with ranking $r = r(t)$ after the first $t$ epochs. The constant $C$ is used to normalize
the weights so that their sum remain constant.

A first variation on this idea is to multiply the weight update rule
by the density of the scores that are in the neighborhood of the score
of the example ranked $r$ so that

$$w_r(t + 1) = Ce^{-\gamma g(r)}$$

Here $g(r)$ represents the empirically measured or continuously
smoothed density of the classifier scores around the score of the
This is why the AUC[CROC] are slightly different for \( \alpha \). AUC[ROC] in the weighted and unweighted cases seem identical in the unweighted case. Although after convergence the 33 epochs if multiplied or not by the output density \( g(r) \), presented after 1 and 3 iterations and after convergence of the algorithm. The IRV converges after 101 iterations if the weights are not updated, 42 if they are updated by \( w_r = 1/r \), and 33 if they are updated by \( w_r = g(r)/r \). Results are 10-fold cross-validated on the HIV data set. Best performances are in **bold** and second-best performances are in *italics*.

\[ \begin{array}{cccccc}
\text{After iteration} & w_r = 1 & w_r = 1/r & w_r = g(r)/r & w_r = 1 & w_r = 1/r & w_r = g(r)/r \\
\hline
\text{After 1 iteration} & 0.569 & 0.598 & 0.611 & 0.348 & 0.365 & 0.366 \\
\text{After 3 iterations} & 0.641 & 0.646 & 0.652 & 0.368 & 0.380 & 0.386 \\
\text{After convergence} & 0.660 & 0.660 & 0.661 & 0.404 & 0.403 & 0.404 \\
\end{array} \]

Table 4. Comparison of the AUC[CROC] (\( \alpha = 7 \) and \( \alpha = 80 \)) of the regular IRV and the IRV with weights updated by \( w_r = 1/r \) multiplied or not by the output density \( g(r) \), presented after 1 and 3 iterations and after convergence of the algorithm. The IRV converges after 101 iterations if the weights are not updated, 42 if they are updated by \( w_r = 1/r \), and 33 if they are updated by \( w_r = g(r)/r \).

A second variation on this idea is to change the parameter \( \gamma \) during training, increasing it monotonically at each epoch, according to some schedule, linear or other. As in simulated annealing, the goal is to start with low values of \( \gamma \) to avoid getting stuck in early local minima.

In the corresponding series of experiments, for clarity we focus on the IRV exclusively, but in principle the same ideas can be applied to any learning method that allows for weighting the importance of the training instances. Table 3 shows the results for exponential weighting schemes, which greatly speed up the convergence of the algorithm, by a factor of 10 or so. Four to eight epochs of training with exponential weighting of the examples is sufficient to reach convergence. In contrast, the unweighted IRV requires 101 epochs.

Although the final results obtained with exponential weighting are slightly inferior to those obtained without any weighting, the weighted IRV still outperforms the SVM and kNN algorithms. When a multiplicative factor associated with the density of the classifier scores around a given example is introduced into the weighting scheme, mixed results are observed.

As shown in Table 4 the 1/r power-law weighting scheme performs the best in this case, converging 3 times faster than the unweighted IRV, and reaching a set of weights that gives the same level of performance, in terms of early enrichment or AUC[CROC], as in the unweighted case. Convergence takes 42 epochs, and only 33 epochs if multiplied by the density \( g(r) \), as opposed to 101 epochs in the unweighted case. Although after convergence the AUC[ROC] in the weighted and unweighted cases seem identical (0.396), the compounds are ranked similarly but not identically. This is why the AUC[CROC] are slightly different for \( \alpha = 7 \) and, with more decimal precision, the AUC[CROC] with \( \alpha = 80 \) can be differentiated. In any case, on this dataset, we see that for the power \( (1/r) \) weighting scheme, multiplication by the density \( g(r) \) slightly improves both the speed of convergence and the quality of the final classifiers. In summary, this learning scheme can sometimes slightly improve the performance of a classifier while quite often dramatically speeding up the time it takes to train a model by gradient descent. In our study, this speed-up can be close to two-fold.

---

**REFERENCES**


