Efficient Multiclass ROC Approximation by Decomposition via Confusion Matrix Perturbation Analysis

Thomas C.W. Landgrebe, *Member*, *IEEE*, and Robert P.W. Duin, *Senior Member*, *IEEE*

Abstract—Receiver operator characteristic (ROC) analysis has become a standard tool in the design and evaluation of two-class classification problems. It allows for an analysis that incorporates all possible priors, costs, and operating points, which is important in many real problems, where conditions are often nonideal. Extending this to the multiclass case is attractive, conferring the benefits of ROC analysis to a multitude of new problems. Even though the ROC analysis extends theoretically to the multiclass case, the exponential computational complexity as a function of the number of classes is restrictive. In this paper, we show that the multiclass ROC can often be simplified considerably because some ROC dimensions are independent of each other. We present an algorithm that analyzes interactions between various ROC dimensions, identifying independent classes, and groups of interacting classes, allowing the ROC to be decomposed. The resulting decomposed ROC hypersurface can be interrogated in a similar fashion to the ideal case, allowing for approaches such as cost-sensitive and Neyman-Pearson optimization, as well as the volume under the ROC. An extensive bouquet of examples and experiments demonstrates the potential of this methodology.

Index Terms—Pattern recognition, machine learning, design methodology, classifier design and evaluation, receiver operator characteristic, multiclass analysis, cost sensitive, volume under the ROC.

1 Introduction

Receiver operator characteristics (ROCs) [1], [2] have become a standard tool for the design, optimization, and evaluation of two-class classifiers. In cost-sensitive problems, the ROC can be used directly to select the best operating point based on given priors and costs [3]. Similarly, the Neyman-Pearson-type optimization can be carried out simply by selecting the operating point corresponding to a specified error [4]. In imprecise environments, ROC analysis is particularly useful, since it provides the means for comparing competing models over a range of operating conditions [3]. The Area under the ROC (AUC) [5] has become an important performance measure in this regard, since it is invariant to operating conditions. Fluctuations in performance due to variations in class abundances can also be analyzed, since they are constrained to vary along the ROC [6].

However, the ROC has only been studied primarily in the two-class case. Extension to the multiclass case is attractive, since it would confer the benefits of ROC analysis to more problems in pattern recognition. Recently, a number of studies in this area have been performed. The three-class case has been studied in [7] and [8]. In [9], we generalized the multiclass ROC by using a framework involving weighting of classifier outputs, which are analogous to the two-class "classifier threshold." The limitation of the extension was exposed by showing that the computational complexity is exponential with an increasing number of classes C, restricting the analysis to problems with low C. In [10], the ROC convex hull method for comparing classifiers in [11] was shown to extend theoretically to the C-class case. The Volume under the ROC hyperSurface (VUS), which is a generalization of the AUC, has been studied in [12], presenting calculations/estimations of the performance bounds of the VUS as a function of an increasing number of classes C. In [13], a theoretical study of the VUS argued that since the VUS of a random classifier approaches that of a perfect classifier as C increases, the VUS may not, in fact, be a very useful performance measure. In [14], we presented a simplified VUS measure that extends with C, but these approaches are all limited to low C. The work in [15] and [16] propose simplified VUS measures that are efficient for high C problems. The former averages AUC scores between each class and all remaining classes, and the latter averages AUC scores between all class pairs.

The area of multiclass, cost-sensitive, and Neyman-Pearson optimization is also related to the ROC analysis. In the cost-sensitive case, the classifier weight/threshold optimization problem has been posed in an optimization framework. In [17] and [18], greedy search approaches were developed to optimize a classifier to given costs/priors but are prone to local minima. In [9], a naive and greedy approach was presented, as well as a data-driven approach involving the construction of two-class ROCs between all class pairs. Classifier weights are assembled from the various ROC pairs based on the given class priors and

• *The authors are with the Information and Communication Theory Group, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands. E-mail: LandgrebeTCW@gmail.com, r.duin@ieee.org.*

Manuscript received 5 Feb. 2007; revised 2 June 2007; accepted 8 June 2007; published online 6 July 2007.
Recommended for acceptance by N.K. Ratha.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0070-0207.
Digital Object Identifier no. 10.1109/TPAMI.2007.70740.

costs. An evolutionary approach was proposed in [19], attempting to find a global solution to the optimization problem. Other related approaches have been proposed in [20] and [21]. In [22], the theoretical extension of the Neyman-Pearson optimization to a multiclass optimization was discussed, allowing for the specification of any element in the confusion matrix. In [23], we presented an algorithm that allows multiple elements in the confusion matrix to be specified, but a solution is not always guaranteed.

Even though recent research has tackled several areas involved with a multiclass ROC analysis, an efficient approach to constructing the ROC hypersurface that scales to large numbers of classes does not exist. This is desirable, since it would provide a unified tool to perform the various ROC tasks. In this paper, we present such an approach based primarily on observations of many pattern recognition problems. These have shown that it is often the case that many ROC dimensions are independent of each other, which is based on a perturbation analysis. Exploitation of this allows for the decomposition of the ROC problem into a number of independent (or approximately independent) groups of classes that can be optimized independently. These groups are often much smaller than the original problem, reducing the computational requirements drastically. An algorithm is presented, which identifies a potential decomposition, involving perturbing the various classifier weights and inspecting sensitivities in the confusion matrix. The approach also takes a practical stance for problems that cannot be decomposed sufficiently. The perturbation analysis provides information on the most interacting dimensions, guiding the best compromise between ROC construction accuracy and computational burden.

This paper is constructed as follows: In Section 2, a multiclass analysis framework is formalized, as well as the construction of the multiclass ROC. Next, Section 3 discusses the potential and consequences of ROC decomposition, showing how a perturbation analysis can be used to study interactions between ROC dimensions. A case study shows just how effective a decomposition can be at reducing unnecessary complexity. The topic of approximate decomposition as a function of class overlap is studied in Section 4 via a controlled experiment, showing results on cost-sensitive experiments and VUS estimations as a function of class overlap. Section 5 presents the perturbation analysis algorithm that inspects sensitivity to perturbations via the confusion matrix in an efficient manner. A number of experiments are presented in Section 6, which investigate cost-sensitive optimizations by using the decomposition in a number of synthetic and real scenarios. Finally, conclusions are presented in Section 7.

2 Notation and Multiclass ROC Analysis

2.1 Multiclass Analysis Framework

The output $p(\mathbf{x})$ of a multiclass classifier consists of C values, corresponding to classes $\omega_1, \omega_2, \dots \omega_C$, with d-dimensional measurement vector $\mathbf{x}$. The prior probability corresponding to class ω_i is denoted $P(\omega_i)$, with class-conditional density distribution $p(\mathbf{x}|\omega_i)$. The posterior distribution $p(\omega_i|\mathbf{x})$ can then be written according to Bayes rule as $p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x}|\omega_1)P(\omega_1)+p(\mathbf{x}|\omega_2)P(\omega_2)+\ldots+p(\mathbf{x}|\omega_C)P(\omega_C)}$. New objects are assigned by the classifier to the class with the highest output as follows:

$$arg \max_{i=1}^{C} p(\omega_i|\mathbf{x}). \tag{1}$$

TABLE 1
The Multiclass Confusion Rate Matrix Ξ Defined

		estimated			
	$\xi_{i,j}$	$\xi_{i,1}$	$\xi_{i,2}$	$\cdots$	$\xi_{i,C}$
	$\xi_{1,j}$	$\xi_{1,1}$	$\xi_{1,2}$	$\cdots$	$\xi_{1,C}$
true	$\xi_{2,j}$	$\xi_{2,1}$	$\xi_{2,2}$	$\cdots$	$\xi_{2,C}$
	$\vdots$	$\vdots$		$\ddots$	
	$\xi_{C,j}$	$\xi_{C,1}$	$\xi_{C,2}$	$\cdots$	$\xi_{C,C}$

In the case of class overlap, erroneous classifications occur occasionally. The multiclass classifier is evaluated via a $(C \times C)$-dimensional confusion rate matrix Ξ showing the respective classification errors between classes (off diagonal) and correct classifications (diagonal elements), as defined in Table 1. The interaction between classes ω_i and ω_j is denoted $\xi_{i,j}$. Diagonal elements are superfluous, since they are equivalent to the complement of the sum of the off-diagonal elements in the respective row, that is, $\xi_{i,i} = 1 - \sum_{j=1}^{C} \xi_{i,j},\ i \neq j$. In the practical case, where distributions are unknown, and only representative examples per class are available, $p(\omega_i|\mathbf{x})$ is approximated or replaced by other types of "confidence-like" measures such as distances to decision boundaries/support vectors [24]. In this case, a confusion matrix CM is generated via an application of a representative independent test set. These CM outputs are normalized by the absolute number of objects N_i per class ω_i, $N = [N_1, N_2, \dots N_C]^T$, resulting in the confusion rate matrix Ξ, with $\xi_{i,j} = \frac{cm_{i,j}}{N(i)}$. In this paper, we consider both theoretical cases with known distributions and practical problems where only examples are available.

In order to compute each confusion element $\xi_{i,j}$ in the ideal case, the following integration is performed:

$$\xi_{i,j} = p(\omega_i) \int p(\mathbf{x}|\omega_i) I_j(\mathbf{x}) dx. \tag{2}$$

The indicator function $I_j(\mathbf{x})$ specifies the relevant domain

$$I_j(\mathbf{x}) = \begin{cases} 1 & \text{if } p(\omega_j|\mathbf{x}) > p(\omega_k|\mathbf{x})\ \forall k, k \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Equation (2) allows any confusion matrix output to be computed, which is generalized for both diagonal and off-diagonal elements.

2.2 Multiclass ROC

The confusion matrix only defines the performance at a single *operating point*, which is valid for a single prior probability situation, and a single position of the classifier *thresholds/weights*. The classifier thresholds/weights can be manipulated by weighting the classifier output $p(\mathbf{x})$ by

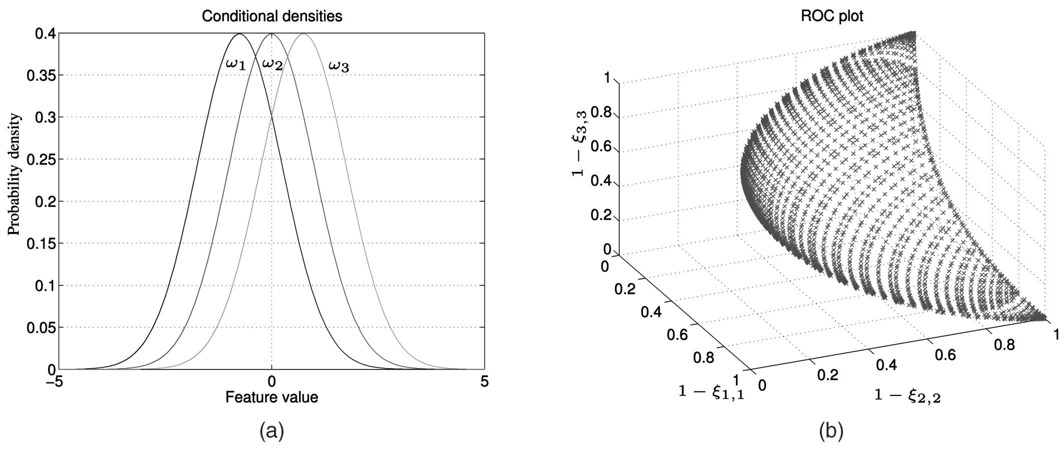


Fig. 1. Example with three Gaussian classes. (a) The distributions. (b) ROC dimensions $1-\xi_{1,1}$, $1-\xi_{2,2}$, and $1-\xi_{3,3}$.

$\Phi = [\phi_1, \phi_2, \dots \phi_C]$, $\phi_i \geq 0\ \forall i$. Equation (1) can then be generalized as (4). Since the class assignment decisions are relative, this implies that there are $C-1$ degrees of freedom, and one weight can be held constant:[1]

$$arg \max_{i=1}^{C} \phi_i p(\omega_i|\mathbf{x}). \tag{4}$$

The full operating characteristic or multiclass ROC can be generated by considering all possible values of Φ. The new Ξ resulting from a new classifier weighting can be calculated by modifying (2), which results to the following:

$$\xi_{i,j}(\Phi) = \phi_i p(\omega_i) \int p(\mathbf{x}|\omega_i) I_j(\mathbf{x}|\Phi) dx. \tag{5}$$

The indicator function $I_j(\mathbf{x}|\Phi)$ is as in (3), except that each posterior is multiplied by the corresponding class weight, that is,

$$I_j(\mathbf{x}|\Phi) = \begin{cases} 1 \text{ if } \phi_j p(\omega_j|\mathbf{x}) > \phi_k p(\omega_k|\mathbf{x})\ \forall k \\ k = 1, 2, \dots C,\ k \neq j \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

In the two-class case, the ROC is monotonically increasing, so efficient generation of thresholds is typically achieved by using ordering of data samples [2]. This is not the general case, so the approach taken here is to generate a combinatorial $(C-1)$-dimensional grid of weightings/thresholds, which considers all possible combinations of interclass weightings. A total of r different arbitrary weightings are used, and thus, the Φ matrix is $r^{C-1} \times C$ in size, which clearly demonstrates the exponential computational complexity of the generalized ROC analysis $O(r^{C-1})$. The resolution must be fine enough, and the scale of each weight that is adequately chosen to ensure the operating characteristic is well sampled. In this paper, r is typically between 80 and 100, a logarithmic scale is used across the range $\{10^{-3}, 10^3\}$, and one arbitrary weight is set to 1. For example, this leads to 1×10^{18} weightings in the 10-class case, with $r = 100$. Fig. 1 is an example of a three-class problem (univariate Gaussians with unit variances, with means at -0.75, 0.00, and 0.75, respectively), showing the distribution (Fig. 1a), three ROC dimensions (Fig. 1b), with $r = 100$, and linear resampling.

3 The Potential and Consequences of Decomposition

The exponential computational requirements (as a function of C) rule out the practical construction of the multiclass ROC for high C problems. However, it may still be possible to generate an equivalent[2] representation in a more efficient manner if a problem lends itself to this. In some cases, an approximate representation may also be useful if the resulting operating characteristic is suitably accurate for the given problem.

Consider the problem in Fig. 2, which depicts a four-class problem between $\omega_1, \omega_2, \omega_3$, and ω_4, with known distributions (Gaussian distributions with unit variances, and means μ, given as follows: $\mu_{\omega_1} = -8$, $\mu_{\omega_2} = -5$, $\mu_{\omega_3} = 5$, and $\mu_{\omega_4} = 8$).

The normalized confusion matrix for the equal prior case and $\Phi = [1\,1\,1\,1]$ is given as follows:

$\xi_{i,j}$	$\xi_{i,1}$	$\xi_{i,2}$	$\xi_{i,3}$	$\xi_{i,4}$
$\xi_{1,j}$	0.9332	0.0668	0.0000	0.0000
$\xi_{2,j}$	0.0668	0.9332	0.0000	0.0000
$\xi_{3,j}$	0.0000	0.0000	0.9332	0.0668
$\xi_{4,j}$	0.0000	0.0000	0.0668	0.9332.

(7)

In this problem, the multiclass ROC consists of $4^2 - 4 = 12$ dimensions. However, the confusion matrix (and, of course, the distribution, which is typically unknown in practical scenarios) suggests that all possible ROC dimensions do not *interact*. For example, the output $\xi_{1,3}$ has a zero value, suggesting no interaction with ω_1.

Theoretically, the multiclass ROC requires an analysis of all possible interactions, that is, the impact of the variation

1. It is important to note that there are several manifestations of multiclass classifiers such as one-versus-all classifiers and error-correcting codes, each having architectural parameters that can be tuned. In this paper, we consider only the final classifier, that is, all parameters have been set, and define the operating characteristic by the surface resulting from all combinations of classifier weightings only. Any variation of classifier internals would result in a new classifier and, thus, a new operating characteristic.

2. "Equivalent" in this sense implies that any analysis/measurements based on the new ROC would return equivalent results to the theoretical case.

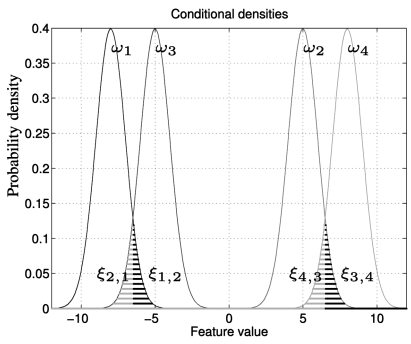


Fig. 2. Example with known distribution, showing significant interactions.

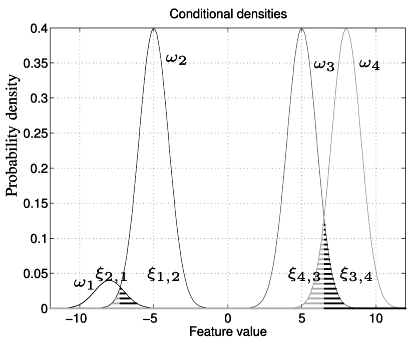


Fig. 3. Example with known distribution, with a perturbation of ϕ_1 from 1.0 to 0.1.

of classifier weight ϕ_k, $1 \leq k \leq C$, on the output $\xi_{i,j}$, $1 \leq i, j \leq C$. However, the example in Fig. 2 makes it apparent that in some cases, varying some weights will have no or little impact on some outputs of Ξ. Thus, if we perturb ϕ_k by $\Delta\phi_k$, it is of interest to understand the resulting variation (sensitivity) in $\xi_{i,j}$, denoted $\Delta\xi_{i,j}(\Delta\phi_k)$. If $\Delta\xi_{i,j}(\Delta\phi_k) = 0$, then $\xi_{i,j}$ is independent of ϕ_k. If $\Delta\xi_{i,j}(\Delta\phi_k) = 0 \; \forall i$, this implies that weight ϕ_k is independent of ω_j; that is ω_j is separable from ω_k.

For classes that are completely separable, for example, ω_j, it holds that

$$\Delta\xi_{i,j}(\Delta\phi_k) = 0 \; \forall i, k. \tag{8}$$

What this implies is that no $\phi_k \; \forall k$ perturbations affect outputs corresponding to ω_j, and thus, ω_j can be excluded from the ROC calculation,[3] thus reducing the computational requirements from $O(r^{C-1})$ to $O(r^{C-2})$.

Another type of conceivable decomposition situation is illustrated in Fig. 2, in which groups of classes interact independently of other classes/groups. For example, ω_k and ω_l are interdependent but independent of $\omega_j \; \forall j, \; j \neq k, \; l$. In this case, $\Delta\xi_{i,k}(\Delta\phi_l) > 0 \; \forall i$, and $\Delta\xi_{i,l}(\Delta\phi_k) > 0 \; \forall i$, but $\Delta\xi_{i,k}(\Delta\phi_m) = 0 \; \forall i, \; m, \; m \neq k, \; l$, and $\Delta\xi_{i,l}(\Delta\phi_m) = 0 \; \forall i, \; m, \; m \neq k, \; l$. The implication of this is that any variations in classifier weights/thresholds $\phi_m \; \forall m, \; m \neq k, \; l$ will have no impact on outputs $\xi_{i,k} \; \forall i$ and $\xi_{i,l} \; \forall i$; that is, all operating points involving different ϕ_m values do not affect ROC dimensions corresponding to the independent groups. This, in turn, implies that the ROC could be decomposed into its constituent groups. Thus, in this example, the original four-class ROC required an $O(r^3)$ calculation, which can now be broken down into two $O(r)$ calculations (note that in this example, the two groups $[\omega_1, \omega_2]$ and $[\omega_3, \omega_4]$ are not theoretically independent because $p(x|\omega_i) > 0, \; -\infty < x < \infty \; \forall i$, but $p(x|\omega_i)$ becomes negligibly small, allowing for the decomposition). To illustrate the group independence, in Fig. 3, it can be seen that a perturbation of ϕ_1 impacts $\xi_{2,1}$ and $\xi_{1,2}$, but no significant impact can be seen with respect to $\xi_{3,4}$ and $\xi_{4,3}$.

This type of analysis could be applied to any trained classifier (given a representative test set) in an attempt to decompose the ROC as far as possible, with the objective of obtaining a tractable calculation.

3. The classifier weight ϕ_j can simply be set to an arbitrary nonzero value, for example, 1, for all operating points.

3.1 Case Study on a 10-Class Problem

Consider, for example, the *Digits-Zernike* data set consisting of 2,000 examples of 10 handwritten digits (from "0" to "9"), originating in Dutch utility maps (available from [25]). In this data set, Zernike moments have been extracted from the original images, resulting in a 47-dimensional representation of each digit. This constitutes a 10-class problem, with an ROC complexity of $O(r^9)$. A classifier is trained on half the data by using a principal component mapping (20 components retained) followed by a Bayes quadratic classifier, which is then evaluated on the remainder of the data. The resulting normalized confusion matrix is graphically shown in Fig. 4a.

It can be seen that most classes appear approximately separable, except for the seventh and 10th, which correspond to digits "6" and "9" (which is intuitive, since the representation does not account for orientation). Outputs $\xi_{7,10}$ and $\xi_{10,7}$ indicate a large degree of overlap. Perturbing ϕ_7 from 1.0 to 10.0 results in the normalized confusion matrix shown in Fig. 4b. The result shows that even though the classifier weighting has varied considerably, the perturbation has only affected $\xi_{7,10}$ and $\xi_{10,7}$ significantly. Similarly, perturbing ϕ_{10} only impacts ω_7 and ω_{10} outputs significantly. If a thorough perturbation analysis is applied to the problem (using, for example, the algorithm presented later), it would follow that the ROC would be found possible to simplify (approximately) via decomposition from $O(r^9)$ to $O(8 + r)$. This reduces the intractable calculation by 9 orders of magnitude.

4 Approximate Decomposition

In some problems, it may not be possible to decompose the ROC sufficiently for a tractable calculation due to larger groups/more interactions. In these situations, it may be possible to construct an approximate ROC that is similar to the ideal case. In the severe case, in which there are still many significant interactions, ROC decomposition may still be useful in accounting for the most prominent interactions, resulting in a suboptimal but nevertheless useful solution. In the cost-sensitive optimization case, the weights obtained from the decomposed ROC could be treated as a good starting point for a postprocessing search approach (for example, see the evolutionary search in [19], the naive and greedy search in [9], and [18]). This could be performed to account for the smaller interactions that were ignored.

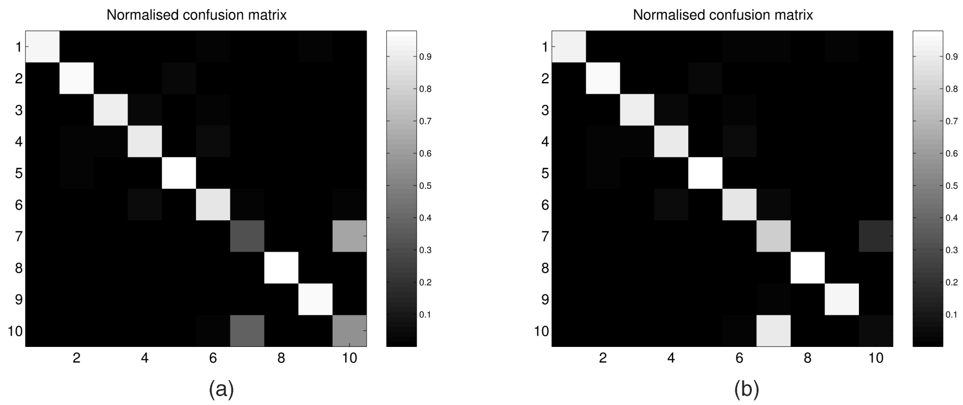


Fig. 4. Normalized confusion matrix for the digits data set example, with unit weighting in (a) and (b) a perturbation of ϕ_7.

In this suboptimal scenario, the perturbation analysis could be used to find the most interacting groups, with the least interacting classes excluded from a group. Consider the example in Fig. 5, which illustrates a three-class problem between normal distributions with unit variance, and means at -3.0, 0.0, and 6.0, corresponding to classes ω_1, ω_2, and ω_3, respectively. There is a large degree of interaction between ω_1 and ω_2, but only a little between ω_2 and ω_3. If the problem is thus decomposed into groups $[\omega_1, \omega_2]$ and $[\omega_3]$, the ROC is reduced from an $O(r^2)$ calculation to an $O(r+1)$ calculation.

It is of interest to understand the impact of the simplification in this case. To assess this, we investigate the difference in performance between the true ROC and the decomposed version in two different scenarios: the first is a cost-sensitive optimization scenario, and the second inspects the impact on the volume under the ROC measure.

4.1 Cost-Sensitive Optimization Validation

Referring to the previous example, in this experiment, 50 different random cost matrices are generated (from a uniform distribution between 0 and 1), and priors are set equal (that is, performance is compared for 50 different operating points). The $(C \times C)$-dimensional cost matrix S consists of profits on the diagonal and of costs off diagonal, denoted $s_{i,j}$ for the cost incurred for a misclassification between ω_i and ω_j. In the example, $C = 3$. The experiment is compared (using the same cost matrices) for several different degrees of interaction between ω_2 and ω_3 by varying the mean of ω_3, denoted μ_3, between 1.0 and 9.0. Thus, the degree of overlap is varied from an extreme to an insignificant degree, allowing for an analysis of the decomposition consequences for higher and lower degrees of interaction. In Fig. 6a, the difference between the loss (9) obtained via the true ROC L_{gt} is compared to the decomposed ROC loss L_d as a function of μ_3. The plot shows the median $\frac{L_d - L_{gt}}{L_{gt}}$ and the upper and lower quartiles (the distribution is heavily skewed). It can be seen that for a high degree of interaction (low μ_3 values), the decomposed ROC performance is generally worse than in the case of lower interaction; that is, the loss has not been reduced as far as possible. For low interaction, there is little difference between the true ROC and decomposed ROC performance, showing that decomposition has little impact on performance:

$$L = \sum_{i=1}^{C} P(\omega_i) \left(\sum_{j=1, i \neq j}^{C} \xi_{i,j} s_{ij} \right) - \sum_{i=1}^{C} P(\omega_i) \xi_{i,i} s_{ii}. \qquad (9)$$

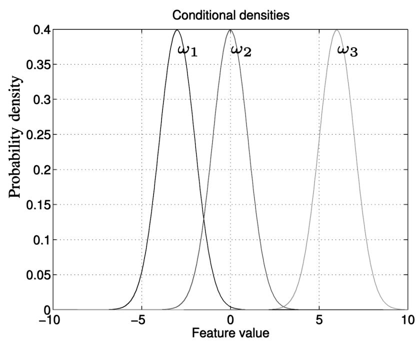
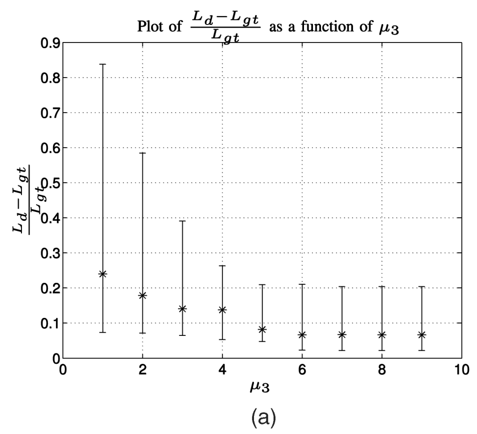


Fig. 5. A three-class problem with a low degree of interaction between ω_2 and ω_3.

4.2 Volume under the ROC Validation

The cost-sensitive experiments validated the decomposition assumption for low degrees of interaction over 50 different operating points. Another type of ROC analysis that is important is the VUS, which extends the popular two-class AUC measure [5] to the multiclass case. The VUS measure allows for an evaluation that encompasses all operating points. The simplified VUS measure proposed in [14] is used for this study, which considers only ROC dimensions corresponding to diagonal elements of the ROC. These performances are equivalent to the complement of a summation of off-diagonal errors in the corresponding confusion matrix row, that is, $\xi_{i,i} = 1 - \sum_{j=1}^{C} \xi_{i,j}$. The upper bound is 1.0, and the lower bound was conjectured to be $\frac{1}{C!}$. The VUS measure can be formalized as $VUS = \int \dots \int \int \xi_{C,C} d\xi_{C-1,C-1} d\xi_{C-2,C-2} \dots d\xi_{1,1}$. This formulation results in a measure that extends with C (at the expense of the simplification) and does not suffer with the limitation, as highlighted in [13]. It is also simple to extend the simplified VUS measure to the decomposed case. Separable groups imply that VUS scores could be computed per group and multiplied.[4] In the three-class example with one separable class, the VUS can be approximated as the VUS between the $[\omega_1, \omega_2]$ group (AUC in

4. The generalization and proof of this is the topic of further research.



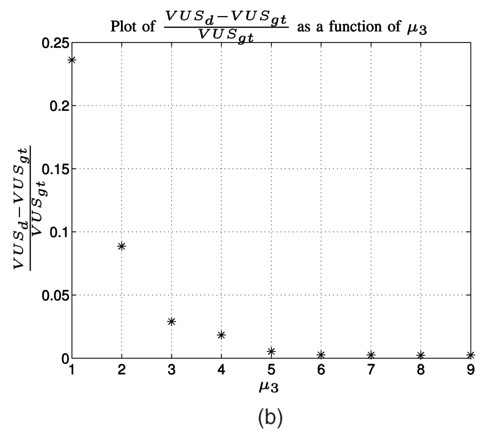


Fig. 6. (a) compares the cost-sensitive optimization performance between the ideal and decomposed ROC techniques, as the degree of interaction between ω_3 and other classes is varied. The median, and upper and lower quartiles are shown for each μ_3 value. (b) compares the VUS performance as a function of ω_3, with $\frac{VUS_d - VUS_{gt}}{VUS_{gt}}$ representing the difference between the ideal and the decomposed calculation.

this case) multiplied by the VUS for ω_3, which is 1. Thus, the VUS is simply the AUC between ω_1 and ω_2.

The decomposition is compared to the ideal case in terms of VUS as a function of the degree of interaction in Fig. 6b. In these experiments, the full three-class ROC surface has been generated for each μ_3 (following the same procedure as in the cost-sensitive case), as well as the decomposed case, consisting of a single two-class ROC (between ω_1 and ω_2), and an independent dimension (corresponding to ω_3). The VUS is then computed for the ideal case VUS_{gt}, as per [14], by using 80 steps and linear resampling. This is compared to the decomposition VUS approximation VUS_d, in which the two-class VUS is multiplied by 1.0 to account for the separable dimension. Fig. 6b shows $\frac{VUS_d - VUS_{gt}}{VUS_{gt}}$ as a function of the separability of ω_3, which indicates the accuracy of the VUS estimation. Note that this VUS is a performance measure; that is, good classifiers result in higher scores. As in the previous experiments, the results clearly show that for higher degrees of interaction, the decomposition results in a poorer estimation. When the interaction is small, the performance difference becomes negligible, and the VUS estimate approaches the true value.

5 Confusion Matrix Perturbation Analysis

The previous sections showed that perturbing the weights and analyzing the subsequent confusion matrix dynamics are useful for identifying independent ROC dimensions and independent groups of dimensions. This implies that the ROC can be decomposed into a number of approximately independent groups. In this section, an algorithm is presented as a general and efficient approach for recovering this decomposition, with a computational complexity that is linear with C. The algorithm should also be capable of ignoring smaller interactions, since it was shown that this has little impact on performance. This may be important to maintain a reasonable computational complexity.

The decomposition algorithm basically involves inspection of the resulting confusion rate matrices Ξ, as each classifier weight is independently perturbed. As a starting point, the algorithm considers a "default" unoptimized classifier (which is often trained by using population or balanced priors), that is, a single operating point. It is important that the default classifier is not, at an extreme, an operating point, that is, where one or more classes have no discriminability, by ensuring that $\phi_i > 0\ \forall i$.

5.1 Algorithm

At the first step of the decomposition, each weight ϕ_i is perturbed independently of other weights $\phi_j\ \forall j, j \neq i$, with ϕ_j weights held constant at fixed values c_j, $1 \leq j \leq c$, $j \neq i$, $c_j \neq 0$ (for example, $c_j = 1\ \forall j$). The weight ϕ_i is successively varied across the range $\{10^{\alpha_1}; 10^{\alpha_2}\}$, ranging from very small to large values. In practice, a log_{10} scale is used between the extrema $\alpha_1 = -3$ and $\alpha_2 = 3$, with r steps (the same scale as used to generate the multiclass ROC). Each of the r perturbations result in a new Ξ, denoted Λ_i, as formulated in (10), with dimensionality $C \times C \times r$. The weight perturbation procedure is repeated for each weight, resulting in C of these Λ structures

$$\Lambda_i = \xi_{j,k}(\Phi_i(\Theta))\ \forall j, k. \tag{10}$$

The perturbation matrix $\Phi_i(\Theta)$ is constructed as follows (the subscript i specifies the column at which we can apply the perturbation):

$$\Phi_i(\Theta) = \begin{bmatrix} c_1 & c_2 & \dots & c_{i-1} & \theta_1 & c_{i+1} & \dots & c_C \\ c_1 & c_2 & \dots & c_{i-1} & \theta_2 & c_{i+1} & \dots & c_C \\ \vdots & & & & & & & \\ c_1 & c_2 & \dots & c_{i-1} & \theta_r & c_{i+1} & \dots & c_C \end{bmatrix}. \tag{11}$$

The logarithmic perturbation vector $\Theta = [\theta_1, \theta_2, \dots, \theta_r]$ is calculated, given the number of steps r, across the range $\{10^{\alpha_1}, 10^{\alpha_2}\}$ as follows:

$$\Theta = \left[10^{\alpha_1}, 10^{(\alpha_1 + \frac{\alpha_2 - \alpha_1}{r-1})}, 10^{(\alpha_1 + 2\frac{\alpha_2 - \alpha_1}{r-1})}, \dots 10^{(\alpha_1 + (r-2)\frac{\alpha_2 - \alpha_1}{r-1})}, 10^{\alpha_2}\right]. \tag{12}$$

The Λ data cubes contain information pertaining to dependencies between ROC dimensions and a particular weight. The dependencies are revealed by simplifying the $(C \times C \times r)$-dimensional Λ_i matrices into $C \times C$ matrices, denoted Λ_i^s. These depict the sensitivity range of each Λ_i output. Each element of Λ_i^s corresponding to the jth row and kth column can be derived from Λ_i as follows:

$$\begin{aligned}\Lambda^s_{i(j,k)} &= \Delta\Lambda_{i(j,k)} \\ &= \max(\Lambda_{i(j,k,l)}) - \min(\Lambda_{i(j,k,l)}) \forall l, 1 \leq l \leq r.\end{aligned} \tag{13}$$

The $(C \times C)$-dimensional Λ_i^s matrix can then be written as

$$\Lambda_i^s = \begin{bmatrix} \Delta\Lambda_{i(1,1)} & \Delta\Lambda_{i(1,2)} & \dots & \Delta\Lambda_{i(1,C)} \\ \Delta\Lambda_{i(2,1)} & \Delta\Lambda_{i(2,2)} & \dots & \Delta\Lambda_{i(2,C)} \\ \vdots & & & \\ \Delta\Lambda_{i(C,1)} & \Delta\Lambda_{i(C,2)} & \dots & \Delta\Lambda_{i(C,C)} \end{bmatrix}. \tag{14}$$

This representation now summarizes the degree of interaction that occurred due to the perturbation analysis for each ROC dimension. It is important to note that these "sensitivities" are the key to an algorithm that can recover the decomposition, independent of the default classifier operating point (provided that it is not extreme). For example, a particular operating point may result in some finite Ξ output on, say, $\xi_{i,j}$. A different operating point may result in a different $\xi_{i,j}$ output. If this output is (approximately) independent of a certain weight ϕ_k, even though the $\xi_{i,j}$ values differ in both cases, the sensitivity, as measured by Λ^s, would be negligible, irrespective of the operating point. A heavily interacting ROC dimension would result in large sensitivities, independent of operating point.

The next step of the algorithm involves simplifying the Λ_i^s matrices into $(1 \times C)$-dimensional vectors, denoted Λ_i^v. These vectors simply consider the most interacting ROC dimension per class with respect to ϕ_i. This simplification is justified because if *any* ROC dimensions interact with ϕ_i, we cannot simplify the ROC. These maximal sensitivity vectors are defined as follows by calculating the largest interaction per class:

$$\Lambda_i^v = \left[\max \Lambda^s_{i(k,1)}, \max \Lambda^s_{i(k,2)}, \dots \max \Lambda^s_{i(k,C)}\right] \forall k. \tag{15}$$

The Λ^v vectors can then be used to decompose the problem directly. This is achieved by comparing each $\Lambda_i^v \; \forall i$. To simplify this analysis, the Λ^v vectors can be binarized, with all "sensitive" entries set to 1, and the insensitive ones to 0. This is also the point at which interactions considered to be insignificant can be eliminated by introducing a sensitivity threshold t_s. The binarized sensitivity vectors, denoted $\Lambda_i^{v_b}$, for ω_i can be written as follows for each element k of Λ_i^v, denoted $\Lambda^v_{i(k)}$, $1 \leq k \leq C$:

$$\Lambda^{v_b}_{i(k)} = \begin{cases} 1 & \textit{if } \Lambda^v_{i(k)} > t_s \\ 0 & \textit{otherwise.} \end{cases} \tag{16}$$

Choosing an appropriate t_s is achieved by inspecting Λ_i^v. Interacting classes typically result in large values, whereas values corresponding to approximately independent groups are much smaller. Thus, t_s should be set just larger than the smallest "insignificant" interaction, as justified by the study in Section 4. Importantly, t_s can also be used to limit the computational burden by restricting the maximum group size. In this case, t_s is chosen to obtain at most M groups. In this scenario, the decomposition may not result in optimal performance but may nevertheless be a reasonable approximation, since the most interacting classes are accounted for.

The decomposition is now complete. Common values of 1 in the columns of the Λ^{v_b} vectors indicate which classes interact with which weights. A 0 in the $\Lambda_i^{v_b}$ columns indicates which classes are independent of which weights. Additionally, if only a single element in any $\Lambda_i^{v_b}$ is 1, this indicates that the corresponding class is independent and can be removed from the ROC calculation. In this case, a fixed nonzero weighting could be used for all operating points.

5.2 Illustration of Algorithm

The algorithm is presented via a running example for clarity, consisting of four Gaussian-distributed classes ($\omega_1, \omega_2, \omega_3$, and ω_4), with means occurring at $\mu_1 = -8$, $\mu_2 = 0$, $\mu_3 = -6$, and $\mu_4 = 5$ and with 0.5 variances (plotted in Fig. 7a).

It can be seen that classes ω_1 and ω_3 interact together significantly, approximately independently of the near-separable ω_2 and ω_4. For reference purposes, the normalized confusion matrix Ξ for an equal prior is shown in Fig. 7b, illustrating interclass errors and intraclass performances for a fixed operating point (this is referred to as the *default* confusion matrix).

Fig. 8 presents some Λ slices for the example, shown for three different weightings (the four columns correspond to Λ_1, Λ_2, Λ_3, and Λ_4, respectively). It can be seen that varying ϕ_1 or ϕ_3 affects both ω_1 and ω_3 outputs significantly in each case, whereas ω_2 and ω_4 perturbations have little effect on any outputs, suggesting independence. The Λ matrices thus provide a mechanism for assessing the dependence between a particular weight and the various ROC dimensions.

Next, the Λ matrices are simplified into $(C \times C)$-dimensional sensitivity matrices Λ^s, as depicted in Fig. 9. These show clearly that there is an interaction between ROC dimensions $\xi_{1,1}$, $\xi_{1,3}$, $\xi_{3,1}$, and $\xi_{3,3}$.

The most interacting ROC dimensions per class are then identified, resulting in the following Λ^v vectors:

$$\begin{aligned}\Lambda_1^v &= [0.8554 \quad 0.0000 \quad 0.8554 \quad 0.0000], \\ \Lambda_2^v &= [0.0000 \quad 0.0058 \quad 0.0003 \quad 0.0055], \\ \Lambda_3^v &= [0.8554 \quad 0.0003 \quad 0.8558 \quad 0.0000], \\ \Lambda_4^v &= [0.0000 \quad 0.0055 \quad 0.0000 \quad 0.0055].\end{aligned} \tag{17}$$

These are then binarized via (16), resulting in Λ^{v_b}, by using $t_s = 0.01$, resulting in

$$\begin{aligned}\Lambda_1^{v_b} &= [1 \quad 0 \quad 1 \quad 0], \\ \Lambda_2^{v_b} &= [0 \quad 0 \quad 0 \quad 0], \\ \Lambda_3^{v_b} &= [1 \quad 0 \quad 1 \quad 0], \\ \Lambda_4^{v_b} &= [0 \quad 0 \quad 0 \quad 0].\end{aligned} \tag{18}$$

The decomposition groupings for the example are thus $[\omega_1, \omega_3]$, $[\omega_2]$, and $[\omega_4]$, reducing the $O(r^3)$ calculation to $O(2 + r)$.

5.3 The Necessity of Sensitivity Analysis

At first glance, it could be reasoned that a confusion matrix at any operating point is useful in identifying dependent and independent ROC dimensions. Theoretically, this is

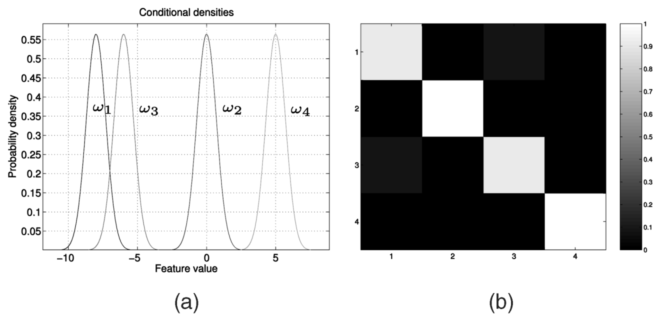


Fig. 7. Probability density functions for the four-class example with known distributions in (a) and with default Ξ for the example in (b) $(\Phi = [1\,1\,1\,1])$.

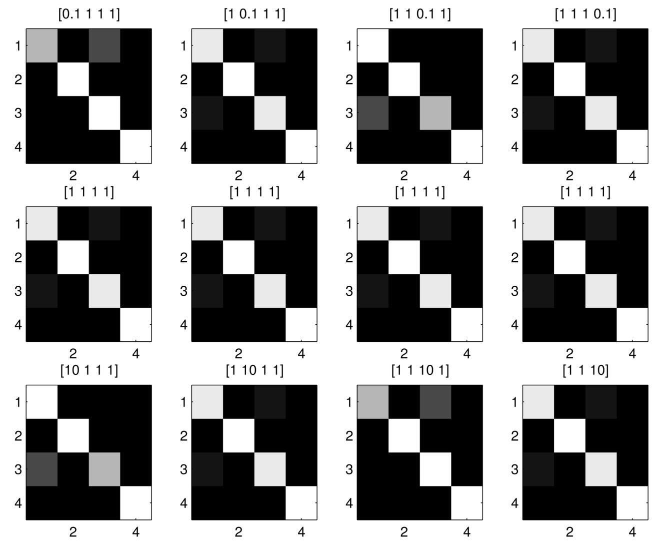


Fig. 8. Slices of Λ for the four-class example for a number of different Φ weightings shown above each plot.

true ($\xi_{i,j} = 0$ if ω_i is independent of ω_j, irrespective of the operating point), but from a practical standpoint, it is of interest to get some measure of how sensitive various dependencies are to variations in operating point. To emphasize this rather subtle point, consider the three-class problem in Fig. 10 between ω_1, ω_2, and ω_3. The first class consists of a bimodal Gaussian distribution with means at

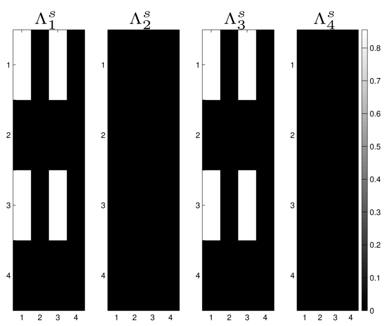


Fig. 9. Sensitivity matrices Λ^s for the four-class example, resulting from a sensitivity analysis of the Λ structures.

0.0 and 7.5, modal weightings of 0.95 and 0.05, respectively, and variances of 0.5. The second and third classes are unimodal Gaussians with variances of 0.5 and means at −3.0 and 8.0, respectively. Two different operating points are investigated, consisting of a near-balanced weighting in Fig. 10a and a more imbalanced setting in Fig. 10b.

The following Ξ results correspond to the operating points depicted in Fig. 10:

$$\begin{array}{c|ccc} \xi_{i,j} & \xi_{i,1} & \xi_{i,2} & \xi_{i,3} \\ \hline \xi_{1,j} & 0.9273 & 0.0227 & 0.0500 \\ \xi_{2,j} & 0.0142 & 0.9858 & 0.0000 \\ \xi_{3,j} & 0.0000 & 0.0000 & 1.0000, \end{array} \tag{19}$$

$$\begin{array}{c|ccc} \xi_{i,j} & \xi_{i,1} & \xi_{i,2} & \xi_{i,3} \\ \hline \xi_{1,j} & 0.9508 & 0.0018 & 0.0474 \\ \xi_{2,j} & 0.1016 & 0.8984 & 0.0000 \\ \xi_{3,j} & 0.0119 & 0.0000 & 0.9881. \end{array} \tag{20}$$

Inspection of the first Ξ may lead to the (false) conclusion that the problem is approximately separable, since all off-diagonal elements are relatively small. However, the second Ξ makes it apparent that there is a large interaction between ω_1 and ω_2. Thus, it can be reasoned that in attempting to find significantly interacting ROC

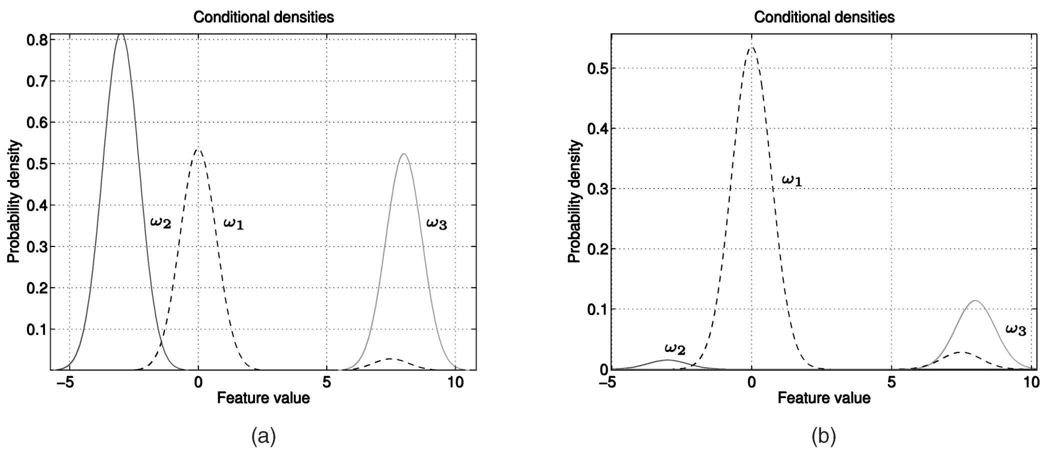


Fig. 10. Comparing two operating points for a three-class problem, with close to a balanced weighting in (a) and a more imbalanced weighting in (b).

dimensions, it is important to inspect some sensitivity to variations in operating point. Perturbation analysis can identify these "weakly" interacting dimensions, independent of the default operating point.

6 Experiments

6.1 Overview

The efficacy of the ROC decomposition approach is demonstrated via a number of experiments in cost-sensitive scenarios involving large numbers of classes. Two experimental sets are presented: The first consists of synthetic examples that demonstrate performance in ideal circumstances, and the second consists of realistic examples that are not necessarily ideal.

The experimental protocol is explained as follows. Each data set is analyzed separately, and a competitive classifier is chosen based on the minimization of the equal error rate $\xi_{eq} = \frac{1}{C}\sum_{i=1}^{C}\sum_{j=1}^{C}\xi_{i,j}$, $i \neq j$. Each classifier is evaluated via a 10-fold cross-validation procedure, with 80 percent of data used for training, and the remainder for testing. The multiclass decomposition algorithm (called *Decomp*) is applied to each test set, with a decomposition threshold t_s (applied via (16)) chosen to suit the problem or to keep the maximum decomposition size smaller than 5 (this constrains the computational complexity to $O(r^3)$). Subsequent to the decomposition analysis, the decomposed multiclass ROC is generated, with $r = 80$. The experiments involve computing the overall loss (9) for 50 different randomly generated cost matrices and balanced priors, effectively evaluating performance at 50 different operating points. Since the experiments involve problems with high C, it is not possible to compute the optimal classifier weightings (as done in the three-class case in Section 4).

The evaluation approach taken is to compare the loss obtained via the *Decomp* approach with three other (non-ROC-based) cost-sensitive optimization approaches. The optimization is with respect to a single operating point, and thus, the algorithms cannot be used for other ROC tasks (such as computing the VUS) but are nevertheless useful as a benchmark in this application. The first approach, denoted *Simple*, uses an extremely basic but fast method for choosing the classifier weights. Each weight is optimized by considering the overall cost per class multiplied by priors. Thus, for ω_i, the corresponding weight is $\phi_i = p(\omega_i)\sum_{j=1}^{C} s_{ij}$. Both intraclass costs and interactions are ignored, but the approach is nevertheless fast and occasionally yields good performance. The other two approaches benchmarked against are two search-based algorithms, called the *Naive* and *Greedy* algorithms, respectively [9], the first of which is a slight modification of the algorithm in [17]. These algorithms attempt to optimize classifier weights according to given priors and costs using a search paradigm. The *Naive* algorithm optimizes each classifier weight independently of others, ignoring possible interactions between weights. The *Greedy* algorithm attempts to account for some interaction by tuning successive classifier weights randomly, dependent on previously optimized weights. The algorithm is repeated three times in experiments with random initializations, in an attempt to avoid local minima. Note that other cost-sensitive optimization algorithms are possible, as discussed in Section 1, which is currently an active area of research. The final algorithm that is implemented involves a postprocessing step applied to the ROC decomposition algorithm, attempting to overcome the independence assumption of the decomposition. This postprocessing step involves a "constrained" *Greedy* algorithm, in which the output of the decomposition is used as an initialization. The algorithm is "constrained" in the way that relative weightings in groups (identified by the decomposition) are unaltered, but classifier weights between groups or independent classes may be varied. This final algorithm is called the *DecompG* algorithm, with three iterations of the postprocessing used in experiments.

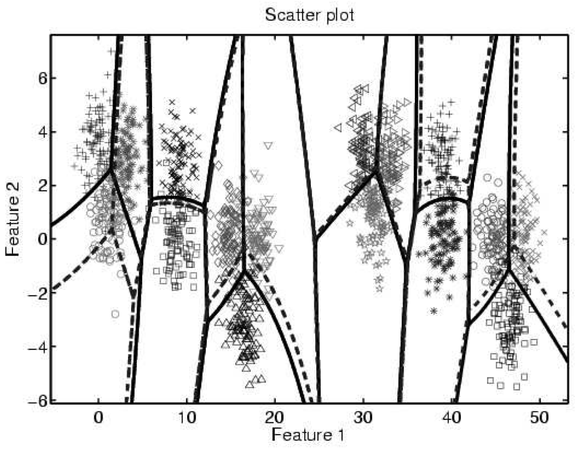
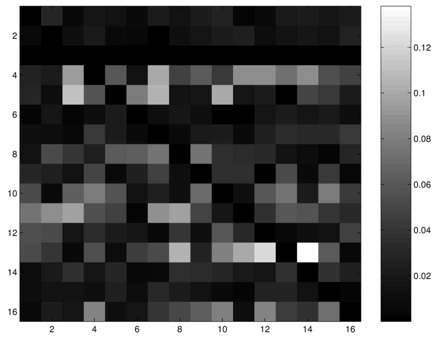


Fig. 11. Scatter plot for the 16-class synthetic data set. The solid line is the decision boundary of the default classifier, and the dashed line is that of the operating point provided by the *Decomp* algorithm for a particular cost.

6.2 Synthetic Experiments

Three experiments have been constructed in order to compare the performance of the various algorithms in an ideal scenario. Each experiment consists of a number of identical independent two-class and three-class clusters, in which data has been drawn from a unit-variance Gaussian distribution in two dimensions, with a repeating of these clusters by varying the means only. The means of the two-class clusters are a distance of 3.0 apart. The means of the three-class clusters are $\sqrt{\frac{29}{4}}$ apart between the first and second classes, $\sqrt{8}$ between the first and third, and $\sqrt{\frac{37}{4}}$ between the second and third. The first experiment is an eight-class problem with one two-class cluster and two three-class clusters. The second experiment is a 16-class problem, repeating the same structure as the eight-class case, staggered in feature space, as shown in Fig. 11. Finally, the third experiment consists of a 40-class problem, with five repetitions of the eight-class structure. In each experiment, a Bayesian quadratic classifier is used. To illustrate a typical experiment, a cost-sensitive example is presented for the 16-class case. The following cost matrix is passed to *Decomp* algorithm:



The default unoptimized classifier results in a loss of 0.3079 in this example, with the solid line in Fig. 11 representing the respective decision boundary. The *Decomp* algorithm attempts to minimize this by finding a new set of classifier weights, resulting in a loss of 0.1993, which is better than the default case. The following classifier weights result:

$$\Phi^* = [1.000\,0.323, 0.005, 1.216, 0.651, 0.377, 0.377, 5.356, 0.515, 0.476, 0.761, 0.019, 0.218, 0.276, 0.037, 1.124]. \quad (21)$$

The dashed line in Fig. 11 depicts the decision boundary resulting from the *Decomp* algorithm, deviating significantly from the default case.

In Table 2, the results of the synthetic experiments are shown. The table shows the number of times (out of the 50 cost-sensitive experiments) that the decomposition algorithm is better (Won) or worse (Lost) than other approaches. Significance is tested via analysis of variance (ANOVA), with the number in parentheses stating the number of experiments that are significantly better/worse based on a 95 percent confidence; that is, for each new cost situation, the difference between the *Decomp* and other algorithm performance is compared for each fold. The variability across algorithms and folds is incorporated into the ANOVA test. The approaches compared are the *Simple* approach and the two search-based algorithms, *Naive* and *Greedy*. The number of wins/losses is computed by comparing the mean performance for each experiment over the 10 cross-validation folds. A decomposition threshold t_s of 0.08 was used for all three data sets. The following computation reduction due to decomposition resulted for the 8-class, 16-class, and 40-class cases, respectively (denoted *Synth8C*, *Synth16C*, and *Synth40C*):

TABLE 2
Results of Synthetic Experiments, Comparing Results Obtained Using the Decomposition Algorithm to the *Simple* Case and to the *Naive* and *Greedy* Algorithms

Dataset	**Simple**		**Naive**		**Greedy**	
	Lost	Won	Lost	Won	Lost	Won
Synth8C	0 (0)	50 (50)	0 (0)	50 (50)	8 (1)	42 (16)
Synth16C	0 (0)	50 (50)	0 (0)	50 (50)	3 (0)	47 (34)
Synth40C	0 (0)	50 (50)	0 (0)	50 (50)	0 (0)	50 (49)

Each comparison shows the number of experiments (out of 50) that the Decomp algorithm was superior (won) and inferior (lost). The number exceeding the 95 percent statistical confidence is shown in parentheses.

- *Synth8C*: $O(r^7)$ reduces to $2O(r^2) + O(r)$,
- *Synth16C*: $O(r^{15})$ reduces to $4O(r^2) + 2O(r)$, and
- *Synth40C*: $O(r^{39})$ reduces to $10O(r^2) + 5O(r)$.

It can be seen that for all three data sets, the *Decomp* algorithm is consistently better than the *Simple* case, which is significant for all experiments. This shows that the *Decomp* algorithm was able to improve classification performance for all conditions. Similarly, the *Decomp* algorithm is consistently better than the *Naive* approach, which is expected, since the latter approach ignores interactions that are present between the various clusters. In the *Greedy* case, it can be seen that it occasionally competes in the eight-class case, showing that it is better able to account for interactions than the *Naive* algorithm. However, in the other two data sets with larger numbers of classes, the *Decomp* algorithm dominates. The *Decomp* algorithm scales easily with the number of classes, whereas the *Greedy* algorithm becomes more susceptible to local minima. These results demonstrate the efficacy of the decomposition approach up to large numbers of classes in ideal circumstances.

6.3 Experiments with Real Data Sets

The experiments undertaken consider a number of data sets with varying numbers of classes, as described in Table 3.

TABLE 3
Important Data Set Statistics Showing the Data Set Source, the Number of Objects, Classes, and Dimensions (d)

Dataset	Source	Objects	C	d	Classifier	ξ_{eq}	t_s	Decomposition
Delve	[27]	2310	7	16	fisher qdc	$13.10(1.27)\%$	0.20	$2O(r^2) + 3$
Soybean	[25]	562	15	35	ldc	$5.20(2.29)\%$	0.05	$O(r^3) + 11$
Dermatology	[25]	358	6	34	fisher ldc	$4.09(1.56)\%$	0.04	$O(r^2) + 3$
Satellite	[28]	6435	6	17	fisher mogc 2	$15.39(0.87)\%$	0.50	$O(r^2) + O(r) + 2$
Segmentation	[25]	2310	7	19	fisher qdc	$7.92(1.15)\%$	0.40	$2O(r^2) + 3$
Nist	[29]	2000	10	256	nlfisher mogc 2	$10.90(1.37)\%$	0.20	$O(r^2) + 2(r) + 4$

The classifier chosen is also shown, followed by the equal error rate ξ_{eq} (with standard deviation), the decomposition threshold used, and the resulting computational reduction due to decomposition.

TABLE 4
Results of Real Experiments for the *Decomp* Algorithm, Following the Same Format as in Table 2

Dataset	Simple		Naive		Greedy	
	Lost	Won	Lost	Won	Lost	Won
Delve	0 (0)	50 (46)	3 (0)	47 (40)	24 (2)	26 (17)
Soybean	0 (0)	50 (50)	0 (0)	50 (49)	19 (0)	31 (2)
Dermatology	0 (0)	50 (42)	0 (0)	50 (46)	46 (0)	4 (0)
Satimage	22 (16)	28 (15)	8 (4)	42 (33)	45 (29)	5 (2)
Segmentation	1 (0)	49 (48)	0 (0)	50 (47)	45 (4)	5 (0)
Nist	45 (1)	5 (0)	43 (8)	7 (0)	50 (50)	0 (0)

TABLE 5
Results of Real Experiments for the *DecompG* Algorithm, Following the Same Format as in Table 2

Dataset	Simple		Naive		Greedy	
	Lost	Won	Lost	Won	Lost	Won
Delve	0 (0)	50 (47)	0 (0)	50 (45)	0 (0)	50 (28)
Soybean	0 (0)	50 (50)	0 (0)	50 (49)	7 (0)	43 (7)
Dermatology	0 (0)	50 (49)	0 (0)	50 (50)	0 (0)	50 (0)
Satimage	0 (0)	50 (34)	0 (0)	50 (45)	7 (0)	43 (11)
Segmentation	0 (0)	50 (49)	0 (0)	50 (50)	4 (0)	46 (4)
Nist	0 (0)	50 (49)	0 (0)	50 (49)	21 (0)	29 (0)

Competing classifiers were chosen by minimization of the equal error rate ξ_{eq} over 10-folds. In the table, "fisher" is a Fisher projection, with the weighted pairwise Fisher mapping [26], denoted "nlfisher." Classifiers are denoted "ldc," "qdc," and "mogc," which are Bayes linear, quadratic, and mixture of Gaussians classifiers, respectively, with the latter followed by the number of mixtures used per class. The table also shows the resulting computational reduction due to decomposition for one fold (this is usually quite stable over folds but varies occasionally).

In many of the experiments, an ideal decomposition with a low enough computational complexity did not result, so an approximate decomposition was forced by using the decomposition threshold. This often resulted in poor performance because some significant interactions were ignored. The *DecompG* algorithm discussed earlier attempts to cope with these limitations.

Comparing the *Decomp* and *Naive* results, it can again be seen that the former algorithm is usually superior, but in the *Nist* data set, it is outperformed. An oversimplified decomposition is attributed to this, but this limitation is again overcome by the *DecompG* algorithm. Considering the *Greedy* results, the *Decomp* algorithm is often beaten, but the *DecompG* algorithm appears superior in most cases. In the *Nist* case, the *DecompG* and *Greedy* algorithms result in very similar performances. The postprocessing step is thus important to refine the results when an oversimplified decomposition is used.

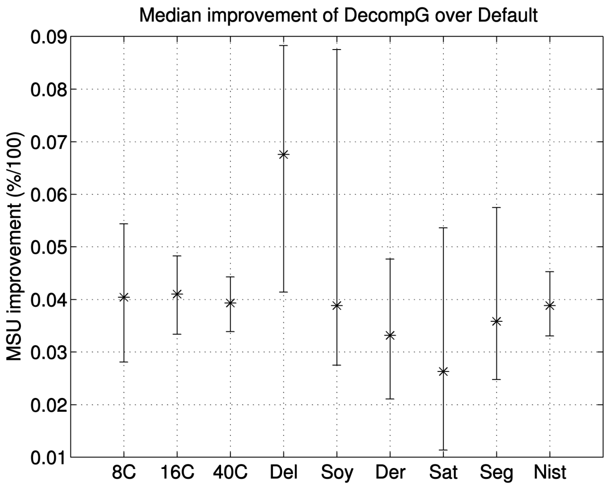


Fig. 12. Plot showing the median improvement (in percent) in performance that the *DecompG* cost-sensitive algorithm gives over the default case across the various data sets.

6.4 Summary

The experimental results (see Tables 4 and 5) show that in some data sets, a very efficient decomposition can be created, resulting in good performance. In these cases, there are only a few (significantly) interacting ROC dimensions. In other data sets, there are fewer independent groups/large groups, and thus, for computational reasons, the decomposition must be oversimplified. In these cases, the *DecompG* algorithm is important, helping to overcome these limitations.

A final result is presented in Fig. 12 in order to give some indication of how much improvement, on the average, the *DecompG* algorithm gives over the default case (that is, no optimization performed). A useful measure to consider here is the Mean Subjective Utility (MSU) score [30]. The MSU score scales the resulting loss L obtained between 0 and 1 (higher scores are better) based on the given priors and costs. This is more meaningful than L in assessing the absolute improvement, since L scales according to costs and prior probabilities. In Fig. 12, the default MSU scores are subtracted from the *DecompG* algorithm scores for each data set. The median of each cost-sensitive experiment (over the 10 folds) is considered, resulting in 50 MSU scores. The figure shows the median of these scores, together with the respective upper and lower quartiles. It can be seen that in some cases, the new operating points improve performance by over 8 percent, which illustrates how beneficial the optimization can be.

7 Conclusions

This paper has considered the construction of multiclass ROC curves, pointing out that even though this is theoretically possible, the exponential computational complexity with an increasing number of classes C is severely restrictive. It was argued, however, that many practical problems can be simplified because not all of the $C^2 - C$ ROC dimensions interact (significantly). In fact, in some cases, there are only a few interacting dimensions, with the consequence that the ROC can be decomposed into a number of lower dimensional groups. This reduces the computational complexity drastically. The consideration of approximate decomposition was discussed, showing that if there is only a small degree of interaction between two decomposed groups, the interaction can be ignored, since it does not have a significant impact on performance. This was shown on an example for both cost-sensitive optimization experiments and via a simplified VUS study. Approximate decomposition allows for a more efficient ROC construction, which is required for many real problems.

An algorithm has been proposed, which identifies independent classifier weights and groups of weights, given a trained classifier at an arbitrary operating point and a representative test set. The algorithm inspects the sensitivity of each ROC dimension to variations of each classifier weight and has a computational complexity that is linear with increasing C. A number of cost-sensitive optimization experiments were conducted, involving synthetic experiments in ideal circumstances and realistic ones with no restrictions. The synthetic experiments showed the efficacy of the proposed methodology in problems of up to 40 classes, outperforming the unoptimized classifier significantly in all experiments and outperforming two search-based techniques. Experiments with real data sets showed that the decomposition approach is generally significantly better than the unoptimized case. In many cases, it was found that the search-based approaches competed or dominated, which is attributed to oversimplified decomposition (required for computational reasons). A modified algorithm was also implemented, which uses a search approach as a postprocessing step. This proved to perform better, helping to overcome the limitation of the oversimplification.

The theoretical arguments and experiments make a strong case for this type of methodology for general multiclass problems in pattern recognition, scaling with large numbers of classes. This may form the basis for important future works, generalizing the various useful tools used in two-class ROC analysis to multiclass problems.

Acknowledgments

This research was supported by the Technology Foundation STW, the Applied Science Division, Netherlands Organization for Scientific Research (NWO), and the Technology Program of the Ministry of Economic Affairs, The Netherlands.

References

[1] C. Metz, "Basic Principles of ROC Analysis," *Seminars in Nuclear Medicine,* vol. 3, no. 4, 1978.

[2] T. Fawcett, "An Introduction to ROC analysis," *Pattern Recognition Letters,* special issue on ROC analysis, vol. 27, pp. 861-874, 2005.

[3] F. Provost and T. Fawcett, "Robust Classification for Imprecise Environments," *Machine Learning,* vol. 42, pp. 203-231, 2001.

[4] R. Duda, P. Hart, and D. Stork, *Pattern Classification,* second ed. Wiley-Interscience, 2001.

[5] A. Bradley, "The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition,* vol. 30, no. 7, pp. 1145-1159, 1997.

[6] T. Landgrebe and R. Duin, "Combining Accuracy and Prior Sensitivity for Classifier Design under Prior Uncertainty," *Proc. IAPR Int'l Workshops Structural and Syntactic Pattern Recognition,* pp. 512-521, Aug. 2006.

[7] D. Mossman, "Three-Way ROC's," *Medical Decision Making,* vol. 19, pp. 78-89, 1999.

[8] S. Dreisetl, S. Ohno-Machado, and M. Binder, "Comparing Trichotomous Tests by Three-Way ROC Analysis," *Medical Decision Making,* vol. 20, no. 3, pp. 323-331, 2000.

[9] T. Landgrebe and R. Duin, "Approximating the Multiclass ROC by Pairwise Analysis," *Pattern Recognition Letters,* 2007.

[10] A. Srinivasan, "Note on the Location of Optimal Classifiers in N-Dimensional ROC Space," Technical Report PRG-TR-2-99, Computing Laboratory, Oxford Univ., Oct. 1999.

[11] F. Provost and T. Fawcett, "Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining,* pp. 43-48, 2001.

[12] C. Ferri, J. Hernandez-Orallo, and M. Salido, "Volume Under the ROC Surface for Multi-Class Problems," *Proc. 14th European Conf. Machine Learning,* pp. 108-120, 2003.

[13] D. Edwards, C. Metz, and R. Nishikawa, "The Hypervolume under the ROC Hypersurface of 'Near-Guessing' and 'Near-Perfect' Observers in N-Class Classification Tasks," *IEEE Trans. Medical Imaging,* vol. 24, no. 3, pp. 293-299, Mar. 2005.

[14] T. Landgrebe and R. Duin, "A Simplified Extension of the Area under the ROC to the Multiclass Domain," *Proc. 17th Ann. Symp. Pattern Recognition Assoc. South Africa,* Nov. 2006.

[15] F. Provost and P. Domingos, *Well Trained PETs: Improving Probability Estimation Trees,* CeDER Working Paper IS-00-04, Stern School of Business, New York Univ., 2001.

[16] D. Hand and R. Till, "A Simple Generalisation of the Area under the ROC Curve for Multiple Class Classification Problems," *Machine Learning,* vol. 45, pp. 171-186, 2001.

[17] N. Lachiche and P. Flach, "Improving Accuracy and Cost of Two-Class and Multi-Class Probabilistic Classifiers Using ROC Curves," *Proc. 20th Int'l Conf. Machine Learning,* pp. 416-423, 2003.

[18] D. O'Brien and R. Gray, "Improving Classification Performance by Exploring the Role of Cost Matrices in Partitioning the Estimated Class Probability Space," *Proc. 22nd Int'l Conf. Machine Learning Workshop ROC Analysis in Machine Learning,* 2005.

[19] R. Everson and J. Fieldsend, "Multi-Class ROC Analysis from a Multi-Objective Optimisation Perspective," *Pattern Recognition Letters,* special issue on ROC analysis, vol. 27, 2005.

[20] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive," *Proc. Fifth ACM Int'l Conf. Knowledge Discovery and Data Mining,* pp. 155-164, 1999.

[21] N. Abe, B. Zadrozny, and J. Langford, "An Iterative Method for Multi-Class Cost-Sensitive Learning," *Proc. 10th ACM Int'l Conf. Knowledge Discovery and Data Mining,* pp. 3-11, Aug. 2004.

[22] D. Edwards, C. Metz, and M. Kupinski, "Ideal Observers and Optimal ROC Hypersurfaces in N-Class Classification," *IEEE Trans. Medical Imaging,* vol. 23, no. 7, pp. 891-895, July 2004.

[23] T. Landgrebe and R. Duin, "On Neyman-Pearson Optimisation for Multiclass Classifiers," *Proc. 16th Ann. Symp. Pattern Recognition Assoc. South Africa,* Nov. 2005.

[24] M. Li and I. Sethi, "Confidence-Based Classifier Design," *Pattern Recognition,* vol. 39, pp. 1230-1240, 2006.

[25] P. Murphy and D. Aha, "CI Repository of Machine Learning Databases," Dept. of Information and Computer Science, Univ. of California, ftp://ftp.ics.uci.edu/pub/machine-learning-databases, 1992.

[26] M. Loog, R. Duin, and H.-U.R., "Multiclass Linear Dimension Reduction by Weighted Pairwise Fisher Criteria," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 7, pp. 762-766, July 2001.

[27] "Delve Datasets, Image Segmentation," Univ. of Toronto, http://www.cs.toronto.edu, 2008.

[28] ELENA Project, "European ESPRIT 5516 Project," *Satimage Dataset,* ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases, 2004.

[29] C. Wilson and M.M.D. Garris, "Handprinted Character Database 3," Advanced Systems Division, Nat'l Inst. of Standards and Technology, 1992.

[30] R. McDonald, "The Mean Subjective Utility Score, a Novel Metric for Cost-Sensitive Classifier Evaluation," *Pattern Recognition Letters,* vol. 27, no. 13, pp. 1472-1477, Oct. 2006.

Thomas C.W. Landgrebe received the BSc (Eng) and MSc (Eng) degrees in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa. In 2001, he joined DebTech in the Research and Development Department of De Beers, working on video-based surveillance and tackling problems such as multiobject tracking, stereo vision, and segmentation. In 2003, he started the PhD with the pattern recognition group at Delft Technical University, focused on multiclass operating characteristics and ill-defined environments. He is now with DebTech, involved with mineral discrimination via hyperspectral imaging and real-time multicamera particle analysis. His research interests include pattern recognition, systems design, operating characteristics, computer vision, and hyperspectral imaging. He is a member of the IEEE.

Robert P.W. Duin received the bachelor's degree in applied physics from Delft University of Technology, The Netherlands, and the PhD degree in 1978 for a dissertation on the accuracy of statistical pattern recognizers. In his research, he included various aspects of sensors, image processing, parallel computing, the automatic interpretation of measurements, learning systems, and classifiers. He is currently an associate professor with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology. He was an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and is currently an advisory editor of *Pattern Recognition Letters*. His current research interests include the design, evaluation, and application of algorithms that learn from examples. Recently, he started investigating alternative object representations for classification and became thereby interested in dissimilarity-based pattern recognition and in the possibilities of learning domain descriptions. He is a senior member of the IEEE and a fellow of the International Association for Pattern Recognition (IAPR). He received the Pierre Devijver Award in August 2006 for his contributions to statistical pattern recognition.